Informatik III
Datenbanken

Prof. Dr. R. Laue

Sommersemester 1992

Erstellt von Walter Dörwald, Martin Dobmann und Helmut Hahn
Graphiken von Walter Dörwald, Helmut Hahn und Martin Dobmann

Inhalt:

1. Einführung 1
2. Formale Beschreibungsmittel 3
3. Datenmodelle 6
4. Die Sprache SQL 11
5. Die physikalische Ebene 24
6. Entwurfstheorie relationaler Datenbanken 36
A. Literaturverzeichnis 90

1. Einführung

Es wird darauf hingewiesen, daß die Vorlesung zu großen Teilen aus der im Anhang stehenden Literatur entnommen wurde.

Beispiel Versicherungsunternehmen

In einem Versicherungsunternehmen gebe es die Sparten

Lebens-, Kfz-, Gebäude-, Kranken-, Hausrat-, Transport- und Rechtsschutzversicherung.

Außerdem gibt es Abteilungen für die einzelnen Bereiche. Dort arbeiten Spezialisten für den jeweiligen Bereich. Die Daten fallen in großen Mengen an. Deshalb erfolgt die Speicherung und Verwaltung im Computer.

Für die einzelnen Sparten werden spezielle Auswertprogramme und Datensammlungen benötigt beziehungsweise vorgenommen. Daher muß man Daten, die mehrere Sparten betreffen, mehrfach speichern.

Vorteil

Jede Sparte ist für ihre eigenen Daten verantwortlich, daher ergeben sich keine Kompetenzschwierigkeiten und es sind keine zeitaufwendigen Abstimmungen über Datenfestlegung zwischen den verschiedenen Abteilungen nötig.

Nachteil

Ist eine Information mehrfach vorhanden, so besteht bei einer Änderung das Risiko, daß die Information nicht überall gleichzeitig geändert wird. Ursachen dafür können sein

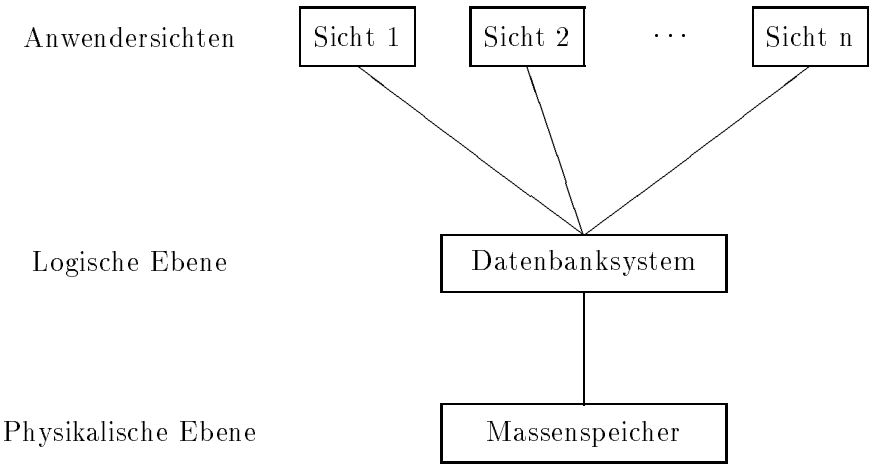
1. Schlechte Organisation
2. Informationen sind verschieden modelliert (Transformation schwierig)

Beispiel

Die Informationen vom Kunden kommen an eine bestimmte Sparte, zum Beispiel Kontoänderung oder Adressänderung. Bei Zahlungsunfähigkeit wird eine Sparte informiert. Für Werbezwecke muß jede Sparte auf die Kundendaten der anderen Sparten zugreifen. Auch die Rentabiltät und Reaktionen auf den Markt sind spartenübergreifend.

1.1 3-Ebenen-Modell

Man kann sich das 3-Ebenen-Modell wie folgt veranschaulichen



Die logische Ebene stellt ein Gesamtmodell aller Daten des Unternehmens dar. Der zu verwaltende Ausschnitt der realen Welt ist hier definiert. Das erfordert die Kenntnis der Bedeutung aller Daten des Unternehmens. Dies ist somit nicht nur eine datenverarbeitungsspezifische Beschreibung!

Die verschiedenen Sichten stellen Ausschnitte aus dieser Gesamtheit dar. Sie entsprechen speziellen Anwenderwünschen, etwa denen einer Abteilung.

Die physikalische Ebene beschreibt die Datenverarbeitungssicht der Daten. Dies beinhaltet die Aufteilung in Dateien, Zugriffswege, Zugriffsrechte und anderes.

Bemerkung

Es wird gefordert, daß die Ebenen unabhängig sein müssen.

- Änderung der physikalischen Speicherung darf Anwenderprogramme nicht berühren.
- Jederzeit müssen neue Sichten auf vorhandene Daten hinzufügbar sein.
- Auf der logischen Ebene sollte es möglich sein, den Ausschnitt der modellierten Welt zu vergrößern, ohne vorhandene Sichten zu ändern.

1.2 Datenbanksystem

Die Aufgabe des Datenbanksystems ist es, eine automatische Transformation der Anwenderwünsche zu Datenverarbeitungsanfragen durchzuführen. Weiterhin muß durch Programme die Rücktransformation der Ergebnisse durchgeführt werden. Außerdem müssen die eingegebenen Daten auf ihre Korrektheit überprüft werden.

2. Formale Beschreibungsmittel

2.1 Das Entity-Relationship-Model

Die Objekte der realen Welt heißen Entities. Objekte gleichen Typs werden zu Entity-sets zusammengefaßt. Objekte besitzen Eigenschaften oder Merkmale, diese heißen Attribute. Alle Entities eines Entity-sets haben die gleichen Attribute.

Beispiel

Hinz und Kunz sind Angestellte. `Hinz`, `Kunz` sind also Entities des Entity-sets `Angestellter`. Attribute sind etwa `Alter`, `Telefonnummer`, `Anschrift`, `Kinderzahl`, `Gehalt` usw.

Seien nun $A_1, \ldots, A_n$ die Attribute eines Entity-sets $\mathcal{E}$. Dann wird jedem A_i ein Wertebereich $D(A_i)$ zugeordnet. Jedes Entity E aus $\mathcal{E}$ wird beschrieben durch $(e_1, \ldots, e_n)$, wobei $e_i \in D(A_i)$. Es ist also e_i der Wert des Objekts E bei Eigenschaft A_i. Falls $A_i =$ Alter, dann ist beispielsweise $e_i = 45$ Jahre.

Die Attributwerte dienen dazu, Eigenschaften zu beschreiben. Damit können zu diesen Objekten Informationen ausgegeben werden. Außerdem dienen sie dazu, die Objekte (Entities) zu identifizieren.

Es werden häufig spezielle Attribute zur Identifikation eingeführt.

Definition

Eine minimale Menge von Attributen, deren Einträge ein Objekt bereits festlegen, heißt *Schlüssel*, mit anderen Worten ein minimales S mit $S \subseteq \mathcal{A} = \{A_1, \ldots, A_n\}$ und der Eigenschaft, daß für jedes $E = (e_1, \ldots, e_n) \in \mathcal{E}$ und $E' = (e'_1, \ldots, e'_n) \in \mathcal{E}$ mit $e_i = e'_i$ für alle $A_i \in S$ folgt, daß $E = E'$ ist.

Es reicht, die Werte bei einem Schlüssel zu kennen, um ein Objekt eindeutig zu identifizieren. Ein spezieller Schlüssel wird im allgemeinen als sogenannter *Primärschlüssel* ausgezeichnet. Es gibt im allgemeinen mehrere Schlüssel. Die Nicht-Primärschlüssel heißen *Sekundärschlüssel*.

Beispiel

Kraftfahrzeuge lassen sich identifizieren über Hersteller, Fahrgestellnummer, über Besitzer und Schlüsselnummer, oder über Polizeiliches Kennzeichen.

Hier ist die Hinzunahme des redundanten Attributs "Polizeiliches Kennzeichen" aus Datenschutzgründen erwünscht.

Neben den Entity-sets bestehen auch Beziehungen zwischen diesen.

Definition

Eine *Beziehung* zwischen $\mathcal{E}_1, \ldots, \mathcal{E}_n$ ist eine Teilmenge von $\mathcal{E}_1 \times \ldots \times \mathcal{E}_n$, das heißt gewisse n-Tupel $(E_1, \ldots, E_n) \in \mathcal{E}_1 \times \ldots \times \mathcal{E}_n$.

Beispiel

Es sei $\mathcal{E}_1 =$ Angestellter und $\mathcal{E}_2 =$ Abteilung. Dann ist die Beziehung "arbeitet in" die Menge der (Ang, Abt) mit Ang $\in$ Angestellter und Abt $\in$ Abteilung

Ang arbeitet in Abt.

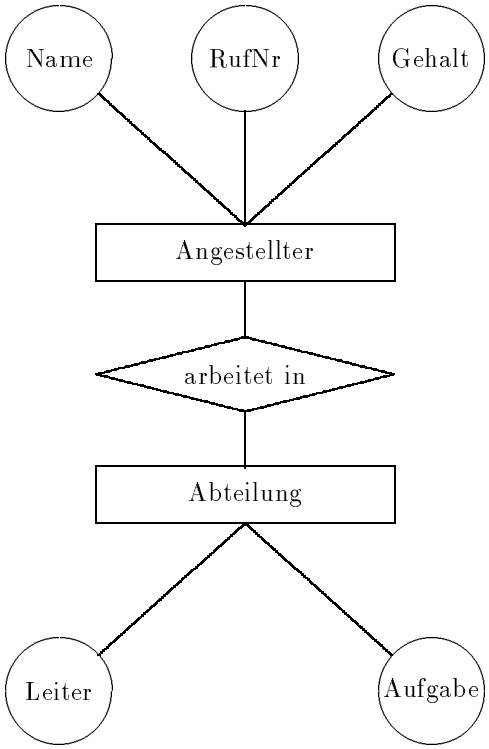
Somit gehören also (Hinz, Versand) und (Kunz, Reklamation) zu "arbeitet in".

2.2 Das Entity-Relationship-Diagramm

Eine Darstellung der realen Welt durch eine Menge von Entity-sets und Beziehungen wird durch einen Graphen veranschaulicht. Es gibt 3 Arten von Knoten

- Attribute werden Kreisen zugeordnet,
- Entity-sets werden Rechtecken zugeordnet,
- Beziehungen werden Rauten zugeordnet.

Ein Entity-Relationship-Diagramm sieht typischerweise wie folgt aus:



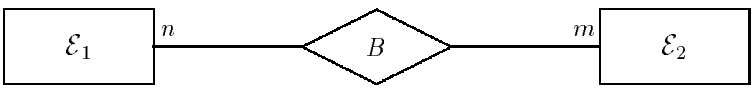
Die Kanten verlaufen zwischen Rechtecken und Kreisen zugehöriger Attribute und zwischen Rechtecken und Rauten zugehöriger Entity-sets. Die Beziehungen sind häufig 2-stellig, wobei es folgende Typen von Beziehungen gibt

- (1,n)-Beziehung

  Zu jedem Entity aus $\mathcal{E}_2$ gehört höchstens ein Entity aus $\mathcal{E}_1$. Somit folgt aus $(E_1, E_2) \in B$ und $(E_1', E_2) \in B$, daß $E_1 = E_1'$ ist.

  Sei $\mathcal{E}_1 =$ Abteilungsleiter und $\mathcal{E}_2 =$ Abteilung, dann ist die Relation B "gehört zu" ein Beispiel für eine (1,n)-Beziehung.

- (n,m)-Beziehung



  Es dürfen mehrere Entities aus $\mathcal{E}_1$ zu mehreren Entities aus $\mathcal{E}_2$ in der jeweiligen Beziehung stehen.

  Sei $\mathcal{E}_1 =$ Student und $\mathcal{E}_2 =$ Vorlesung, dann ist die Relation B "hört" ein Beispiel für eine (n,m)-Beziehung.

- is_a-Beziehung

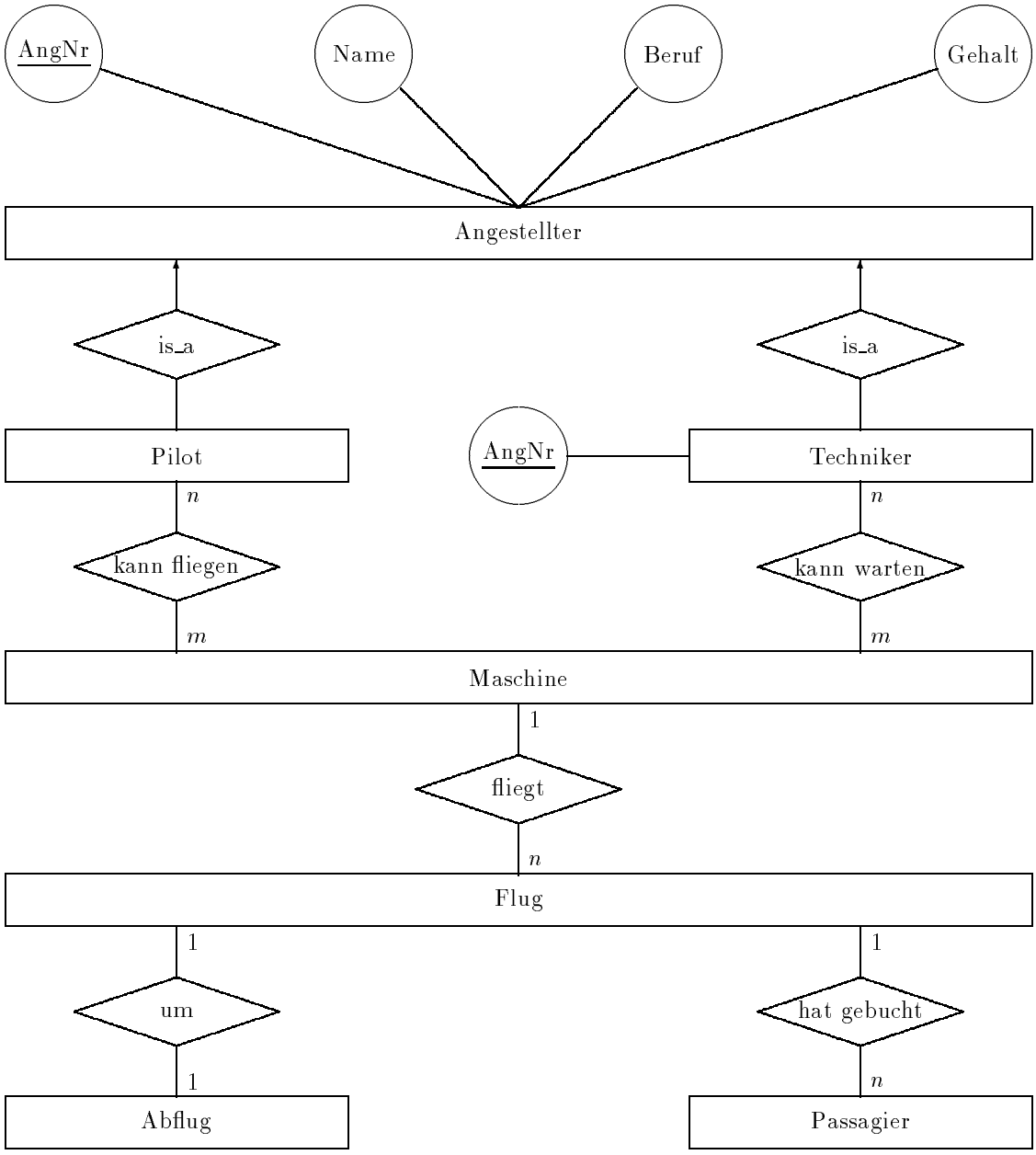
Jedes Entity aus $\mathcal{E}_1$ ist Spezialisierung eines Entities aus $\mathcal{E}_2$

Sei $\mathcal{E}_1$ = Techniker und $\mathcal{E}_2$ = Angestellter, dann ist die Relation B "ist ein" ein Beispiel für eine (is_a)-Beziehung.

Für eine is_a-Beziehung gilt

1. Jedes $E_1 \in \mathcal{E}_1$ hat die Attribute der $E_2 \in \mathcal{E}_2$.
2. Ist $E_1 \in \mathcal{E}_1$, so ist ihm ein $E_2 \in \mathcal{E}_2$ zugeordnet, das die gleichen Attributwerte hat.

Mit diesen Beispielen können wir ein komplexes Entity-Relationship-Diagramm konstruieren, wobei die Schlüsselattribute unterstrichen sind.



3. Datenmodelle

Es gibt im wesentlichen drei verschiedene Modelle, und zwar das Netzwerk, das Hierarchische Modell und das Relationale Modell.

3.1 Netzwerk

In einem Netzwerk gibt es Entity-Sets und (1,n)-Beziehungen.

Definition

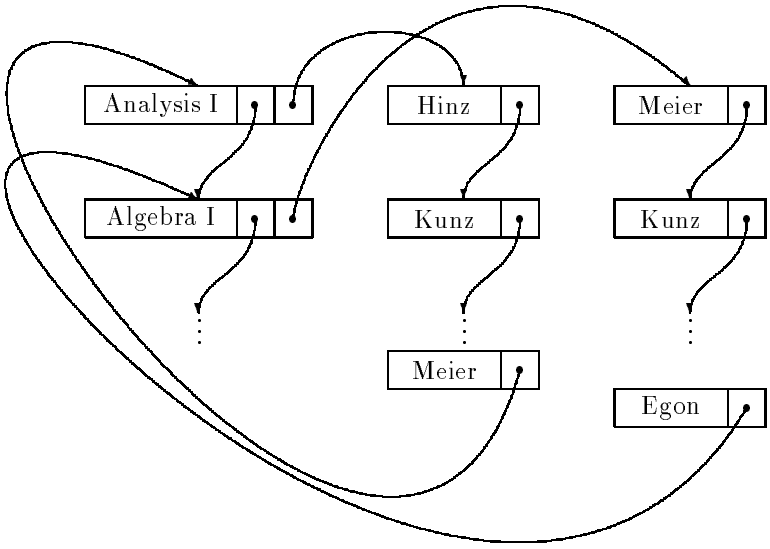
Ist $B \subseteq \mathcal{E}_1 \times \mathcal{E}_2$ eine (1,n)-Beziehung, so heißt $\mathcal{E}_1$ *Owner-Typ* und $\mathcal{E}_2$ *Member-Typ*.

Eine graphische Veranschaulichung für ein Netzwerk erfolgt durch das Bachmann-Diagramm.

Für jedes Entity-Set wird ein Rechteck gezeichnet. Eine (1,n)-Beziehung wird durch einen Pfeil dargestellt.

Beispiel

Es werden die Vorlesungen `Analysis I`, `Algebra I` und so weiter angeboten. Die Vorlesung `Analysis I` hören `Hinz`, `Kunz`, ... `Meier`, und die Vorlesung `Algebra I` hören die Studenten `Meier`, `Kunz`, ... `Egon`. Dann sieht das Bachmann-Diagramm wie folgt aus:



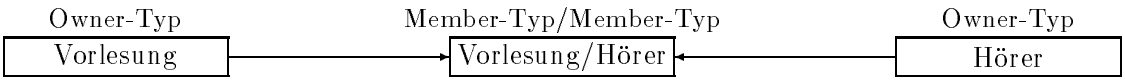
Implementierung von Beziehungen durch Zeigerketten (Lineare Liste)

Für jede Beziehung benötigt man eine eigene Zeigerkette. Die Position eines Zeigers bestimmt die Beziehung, die durch die entsprechende Zeigerkette dargestellt wird.

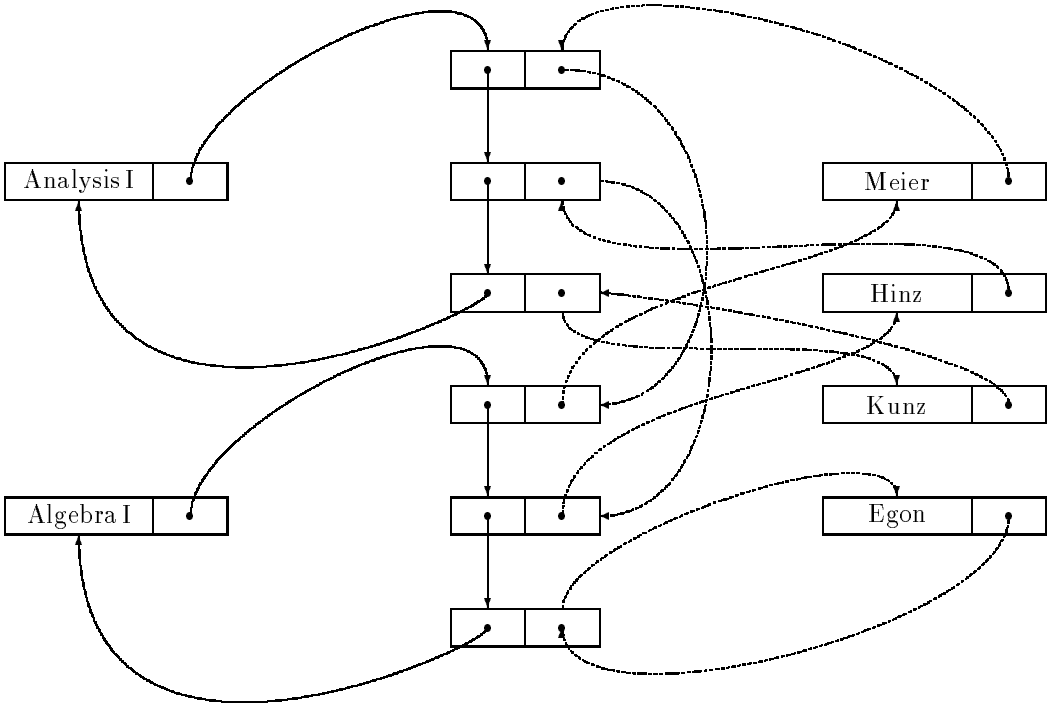
Probleme gibt es bei den (n,m)-Beziehungen. Diese werden aufgelöst zu einer (1,n)-Beziehungen und einer (1,m)-Beziehung.

Beispiel

In diesem Beispiel wollen wir wissen welche Studenten welche Vorlesungen besuchen. Da ein Student mehrere Vorlesung hören kann, und in jeder Vorlesung mehrere Studenten sind, handelt es sich um eine sogenannte (n,m)-Beziehung. Nach Auflösen, dieser für das Netzwerkmodell nicht zu verarbeitenden Beziehung, erhalten wir die Entity-Sets



Für das konkrete Beispiel `Meier`, `Hinz` und `Kunz` hören `Analysis I` und `Meier`, `Hinz` und `Egon` hören `Algebra I` erhalten wir folgende Entities und Beziehungen:



Um zu starten, wird ein Einstiegspunkt benötigt. Dazu wird zu jeder Zeigerkette die zuletzt verwendete Adresse in einer Tabelle gehalten. Wenn (n,m)-Beziehungen so realisiert werden sollten, hätte ein Member mehrere Owner, und dazu müßten mehrere Zeiger für dieselbe Beziehung bei demselben Entity stehen. Somit ist es schwierig zu entscheiden, welcher Zeiger zu welcher Beziehung gehört.

Navigieren

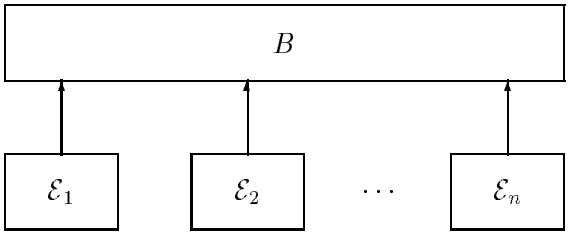
Um die Hörer der Vorlesung Analysis I auszugeben, ist die Zeigerkette startend bei Analysis I zu durchlaufen. Bei jedem Member ist die andere Zeigerkette, die zum Hörer gehört, solange zu durchlaufen, bis der Owner gefunden wird. Dieser wird ausgegeben und der Durchlauf der 1. Zeigerkette fortgesetzt. Breche ab, wenn die Kette zu Analysis I zurückführt.

Mehrstellige Beziehungen

Es sei die mehrstellige Beziehung

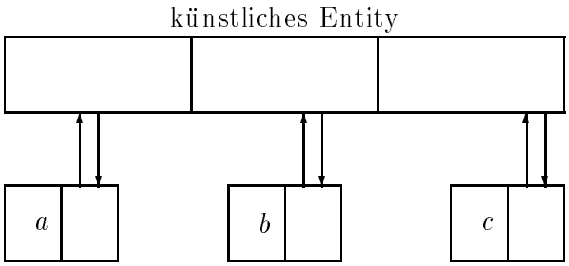
$$B \subseteq \mathcal{E}_1 \times \mathcal{E}_2 \times \ldots \times \mathcal{E}_n$$

gegeben. Die im Netzwerkmodell nicht darstellbare (1,n)-Beziehung



wird durch Einführung eines künstlichen Entity-Types mit n Attributen wie folgt aufgelöst.

Die Entities dieses künstlichen Attributes bestehen aus den Zeigerkombinationen der Entities der $\mathcal{E}_i$'s, die in Relation zueinander stehen. Man erhält also für $(a, b, c) \in B$



Bewertung

Der Zugriff über Zeigerketten ist schnell, aber die Zugriffsroutinen sind völlig abhängig von der Speichermethode. Das Netzwerkmodell entspricht also nicht dem 3-Ebenen-Modell. Denn eine Anfrage erfolgt auf der physikalischen Ebene.

3.2 Hierarchisches Datenmodell (IMS von IBM)

Bei diesem Modell werden die Daten in Bäumen abgespeichert. Man erhält einen "Wald".

Auf der logischen Ebene werden Bäume gebildet, deren Knoten Entity-Sets darstellen. Es gibt nur (1,n)-Beziehungen, die Member-Typen sind Söhne der Owner-Typen. Tritt ein Entity-Set in mehreren Beziehungen auf, so werden virtuelle Kopien definiert. Als Member-Set tritt ein Entity-Set nur einmal auf sonst werden die virtuellen Kopien verwendet. Diese Kopien sind nie Owner-Typ einer Beziehung.

Auf der physikalischen Ebene werden bei jedem Entity variabel viele Zeiger zu Nachfolgern erlaubt. Dafür ist die Menge der Nachfolger geordnet.

Datenstruktur

Die Datenstruktur wird durch einen in Informatik II vorgestellten Baum realisiert.

Darstellung eines Netzwerkes durch eine hierarchische Datenbank

1. Wähle ein Entity-Set als Wurzel des Baumes, wenn möglich ein solches, das nie als Member-Typ einer Beziehung erscheint.
2. Ordne alle die Entity-Sets als Söhne an, die als Member-Typ einer Beziehung zu diesem gewählten Entity-Set auftreten.
3. Versuche, jeden der Söhne als Owner-Typ und damit als Vater einer weiteren Beziehung darzustellen und mache deren Member-Typ zu Söhnen.
4. Ist jedoch ein Member-Typ bereits im Wald aufgetreten, so erfinde an dieser Stelle einen virtuellen Member-Typ, dessen Entities aus Zeigern auf die bereits bestehenden Entities des betrachten Entity-Sets bestehen. Diese virtuellen Entity-Sets werden nie Vater.
5. Kann man den Baum nicht vergrößern, so suche weitere Entity-Sets, die noch nicht verwendet wurden. Dann fahre fort mit 1. Sind alle Entity-Sets verbraucht, so höre auf.

Bewertung

Der Zugriff bezüglich der jeweiligen Hierarchie ist schnell, der Zugriff anhand von Eigenschaften, die in den Blättern beschrieben sind, ist jedoch langsam. Außerdem besteht eine hohe Abhängigkeit der Anwenderprogramme von der Speicherung, das heißt das 3-Ebenen-Modell wird nicht ausreichend unterstützt.

3.3 Relationales Modell (Codd 1971)

Das relationale Modell vermeidet die Verwendung von Zeigern. Die Beziehungen werden wie die Entites behandelt. Die bei den relationalen Datenbanken verwendete Datenstruktur ist die Tabelle.

A_1	A_2	...	A_n
$\vdots$	$\vdots$		$\vdots$
a_1	a_2		a_n
$\vdots$	$\vdots$		$\vdots$

Die Spalten der Tabellen tragen Attributnamen (A_1, A_2, ..., A_n), und jede Zeile ist ein Wertetupel. Ein Tupel $t = (a_1, a_2, \ldots, a_n)$ hat in der i-ten Spalte einen Eintrag a_i aus dem A_i zugeordneten Wertebereich.

Beispiel

Die Datenbank eines Unternehmens enthält die Tabellen `Lieferant`, `Teil` und `Lieferung` mit folgenden Daten

`Lieferant`

LNR	LName	LOrt
L1	Meier	Köln
L2	Meier	Berlin
L3	Schmitz	Köln

`Teil`

TNR	TName	Gewicht
T1	Schraube	20
T2	Mutter	10
T3	Schraube	15

`Lieferung`

LFNR	TNR	Menge
LF1	T1	1000
LF2	T2	500
LF3	T1	2000

Um die in einer Menge von Tabellen abgespeicherten Informationen abzurufen, kann man die Tabellen verknüpfen und gezielt Informationen heraussuchen. Dazu dient als Hilfsmittel eine relationale Algebra.

Wir können folgende Operationen auf Mengen von Tabellen anwenden, wobei die Ergebnisse wieder Tabellen sind.

1. Schnitt, Vereinigung und Differenz

Sind T_1 und T_2 Tabellen und haben für alle i die Attribute $A_i(T_1)$ und $A_i(T_2)$ denselben Wertebereich, so kann man mit T_1 und T_2 die Operationen "$\cap$", "$\cup$" und "$\setminus$" ausführen.

2. Projektion

Sei T eine Tabelle mit den Attributen $A_1, \ldots, A_n$ und $B_1, \ldots, B_k \in \{A_1, \ldots, A_n\}$. Dann kann man T auf $(B_1, \ldots, B_k)$ projizieren, das heißt

1. Streiche jede Spalte A_j mit $A_j \notin \{B_1, \ldots, B_k\}$.
2. Sortiere die Spalten in der Reihenfolge $B_1, \ldots, B_k$.
3. Streiche Duplikate unter den Tupeln.

Die Projektion der Tabelle T auf die Attribute $A_{i_1}, \ldots, A_{i_k}$ wird mit $\pi_{i_1,\ldots,i_k}(T)$ bezeichnet.

4. Selektion

Sei T eine Tabelle mit Attributen $A_1, \ldots, A_n$. Sei D_i der Wertebereich von A_i und auf D_i sei "$\leq$", "$<$", "$=$", "$>$" und "$\geq$" definiert. Dann kann man eine Formel F bilden, indem man in jeder Komponente i vorschreibt, welche Prädikate die Einträge der gesuchten Tupel — formuliert in den Vergleichen und Konstanten aus D_i — erfüllen sollen. Das Ergebnis ist die Menge der Tupel, die allen Prädikaten genügt, also in obigem Beispiel beispielsweise alle Tupel aus `Lieferant`, für die der Ort den Eintrag `Köln` besitzt.

Die Selektion der Tabelle T bezüglich der Formel F wird mit $\sigma_F(T)$ bezeichnet.

5. Natürlicher Verbund ("natural join")

Seien T_1 und T_2 Tabellen. T_1 habe die Attribute $A_1, \ldots, A_n, B_1, \ldots, B_m$ und T_2 habe die Attribute $C_1, \ldots, C_l, B_1, \ldots, B_m$. Weiter sei $A_i \neq C_j$ für alle i und j. $T_1 \bowtie T_2$ ist eine Tabelle mit den Attributen $A_1, \ldots, A_n, B_1, \ldots, B_m, C_1, \ldots, C_l$. Die Tupel von $T_1 \bowtie T_2$ erhält man, indem man zu jedem Tupel $t_1 \in T_1$ jedes Tupel $t_2 \in T_2$ aufsucht, dessen Einträge auf $B_1, \ldots, B_m$ mit den Einträgen von t_1 übereinstimmen, und t_1 auf $C_1, \ldots, C_l$ durch die Einträge von t_2 bei diesen Attributen verlängert. Wenn zu t_1 kein passendes t_2 existiert, wird aus t_1 kein Tupel von $T_1 \bowtie T_2$ erzeugt.

In obigem Beispiel ergibt sich für `Lieferung` $\bowtie$ `Teil`

`Lieferung` $\bowtie$ `Teil`

TNR	TName	Gewicht	LFNR	Menge
T1	Schraube	20	LF1	1000
T1	Schraube	20	LF3	2000
T2	Mutter	10	LF2	500

`Teil` $\bowtie$ `Lieferant` liefert hingegen das kartesische Produkt, da die beiden Tabellen keine gemeinsamen Attribute besitzen.

Es gilt unter anderem $(T_1 \bowtie T_2) \bowtie T_3 = T_1 \bowtie (T_2 \bowtie T_3)$, das heißt der "natural join" ist assoziativ.

6. θ-Verbund

Die Tabelle T_1 habe die Attribute $A_1, \ldots, A_n$ und T_2 habe die Attribute $B_1, \ldots, B_m$. Es seien $C_1, \ldots, C_l \in \{A_1, \ldots, A_n\}$ und $D_1, \ldots, D_l \in \{B_1, \ldots, B_m\}$. Weiter gelte

$$\text{Wertebereich}(C_i) = \text{Wertebereich}(D_i) \quad \text{und} \quad \theta_i \in \{\leq, <, =, >, \geq\} \qquad \text{für alle } i.$$

$T_1 \bowtie_\theta T_2$ entsteht aus den Tupel von T_1, indem aus T_2 die Tupel t_2 mit den Tupeln $t_1 \in T_1$ verbunden werden, bei denen der Eintrag $a_i(t_1)$ von t_1 bei C_i in der θ-Relation zu dem Eintrag $a_i(t_2)$ bei D_i steht. Es ist auch möglich statt $a_i(t_2)$ eine Konstante a zu wählen.

Beispiel

Mittels eines θ-Joins können wir aus der Tabelle

Zugnummer	Ankunft	Abfahrt
601	10:00	10:15
603	10:30	11:00
607	12:00	12:15

eine Umsteigetabelle konstruieren.

Der Eintrag bei `Ankunft` muß kleiner als der bei `Abfahrt` sein. Man bilde also den Verbund der Tabelle mit sich selbst, sodaß die "$\leq$"-Beziehung erfüllt ist.

Bewertung

Das relationale Modell erlaubt es eine Anfrage zu formulieren, ohne daß man bereits angibt, wie die Antwort zu realisieren ist ("deklarativ"). Damit kann die logische Datenbankstruktur getrennt werden von der realen Implementation und die Datenunabhängigkeit nach dem 3-Ebenen-Modell ist gewährleistet.

Allerdings sind die relationalen Datenbanken vor allem bei Verwendung von Verbundoperationen langsamer als die beiden anderen konkurrierenden Modelle.

4. Die Sprache SQL

Die relationale Datenbanksprache SQL wird weiterentwickelt zu SQL2 und SQL3. Für Implementierungen durch mehrere Hersteller wird die Sprache durch eine ANSI-Norm beschrieben. Neuerdings wird die Sprache objektorientiert weiterentwickelt.

4.1 Einige Befehle

```
CREATE TABLE Tafelname
     (Attributname 1    Typ 1    [NOT NULL]
    [,Attributname 2    Typ 2    [NOT NULL]
        ⋮               ⋮            ⋮
     ,Attributname n    Typ m    [NOT NULL]]);
```

Erlaubte Typen

In SQL gibt es die folgenden Typen

- integer $\in [-2^{15}+1, 2^{15}-1]$
- small integer $\in [-2^{31}+1, 2^{31}-1]$
- decimal(m,n): m-stellige Zahl ($m \leq 15$) mit n Nachkommastellen ($0 \leq n \leq m$)
- float $\in [5.4 \cdot 10^{-79}, 7.2 \cdot 10^{75}]$
- char(n): Zeichenkette fester Länge ($n \leq 254$)
- varchar(n): Zeichenkette variabler Länge ($n \leq 254$)
- longchar: Zeichenkette variabler Länge (bis $2^{15}-1$ Symbole)
- drei Graphik-Datentypen

Nur die festen Basistypen haben Vergleichsfunktionen "<", "≤", "<>", ">", "≥", "=". Nullwerte sind erlaubt, außer wenn sie explizit durch "Not Null" verboten worden sind.

```
ALTER TABLE [Erzeuger] Tafelname
     ADD    Attributname    Typ;
```

```
DROP TABLE Tafelname;
```

```
CREATE [UNIQUE] INDEX Indexname ON [Erzeuger.] Tafelname
     (Attributname 1    [ASC|DESC]
    [,Attributname 2    [ASC|DESC]
        ⋮               ⋮
     ,Attributname n    [ASC|DESC]]);
```

Es wird ein B*-Baum erzeugt zum Wertebereich $W_1 \times \ldots \times W_n$, wobei W_i der Wertebereich von Attributname i mit lexikographischer Anordnung ist.

Der B*-Baum ist eine Variation des B-Baumes, bei dem die Daten nur in Blättern gehalten werden. Damit ergibt sich mehr Platz für Zeiger in den inneren Knoten. Werden nur die Adressen bereits vorhandener Daten abgespeichert, so kann ein solcher Zugriffsweg jederzeit hinzugefügt oder gelöscht werden.

Das Schlüsselwort `UNIQUE`

Bei den angegebenen Attributnamen kann zu einem Wertetupel nur ein Datensatz existieren. (zum Beispiel TelNr, Name, Gebdatum, Gehalt). Damit kann man sogenannte Primärschlüssel spezifizieren.

Weitere SQL-Befehle sind

```
INSERT INTO [Erzeuger.] {Tafelname|Sichtname} [(Attributname)]
     VALUES    (Daten);
```

```
DELETE FROM [Erzeuger.] Tafelname
     [WHERE    Bedingung];
```

```
UPDATE [Erzeuger.] Tafelname SET
     Attributname 1 = Ausdruck 1,
     Attributname 2 = Ausdruck 2,
          ⋮               ⋮
     Attributname n = Ausdruck n
    [WHERE Bedingung];
```

Beispiel

```
CREATE TABLE Professor
     (Pnr    char(7)     NOT NULL,
      Pname  varchar(20) NOT NULL,
      Ort    varchar(15) NOT NULL);
CREATE UNIQUE INDEX Profindex ON Professor (Pnr);
INSERT INTO Professor VALUES(``N44´´,``Gauss´´,``Bayreuth´´);
```

Das Data Dictionary

Das Data Dictionary stellt eine Art Datenbank über die Datenbank dar. Es gibt die Strukturen

- SYSCATALOG: Für jede Tabelle ein Tupel.

TNAME	CREATOR	TABLETYP	NCOLS	REMARKS	ROWCOUNT
Name der Tabelle	Benutzer-Kennzeichen	Basis oder Sicht	Attribut-zahl	...	Tupelzahl

- SYSCOLOUMS: Für jedes Attribut jeder Tabelle ein Tupel.

TNAME	CREATOR	NAME	COLNO	COLTYPE	LENGTH	REMARKS
		Attribut-name	Attribut-nummer	Wertebereichs-typ	Länge	...

- Weitere Tabellen beinhalten die Zugriffswege, die Zugriffsrechte und die Views.

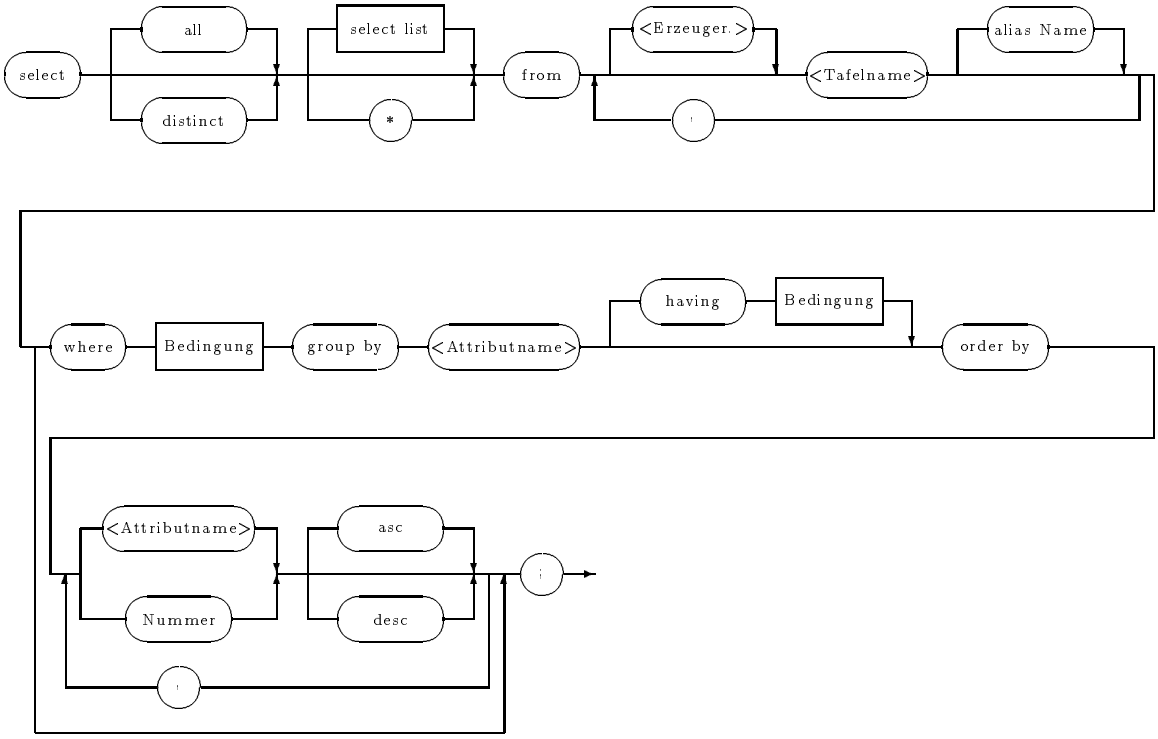
...	VIEWTEXT	...
	Ausdruck, der das View definiert	

Das SELECT-Kommando

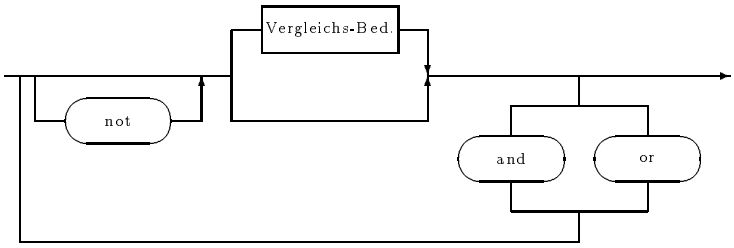
Ein typisches Beispiel für die SELECT-Anweisung ist

```
SELECT Professor_Ort,Student_Name
   FROM Professor,Student
   WHERE Professor_Ort = Student_Ort
   AND Student_Fach = ´Mathematik´
   ORDER BY Professor.Pname ASC;
```

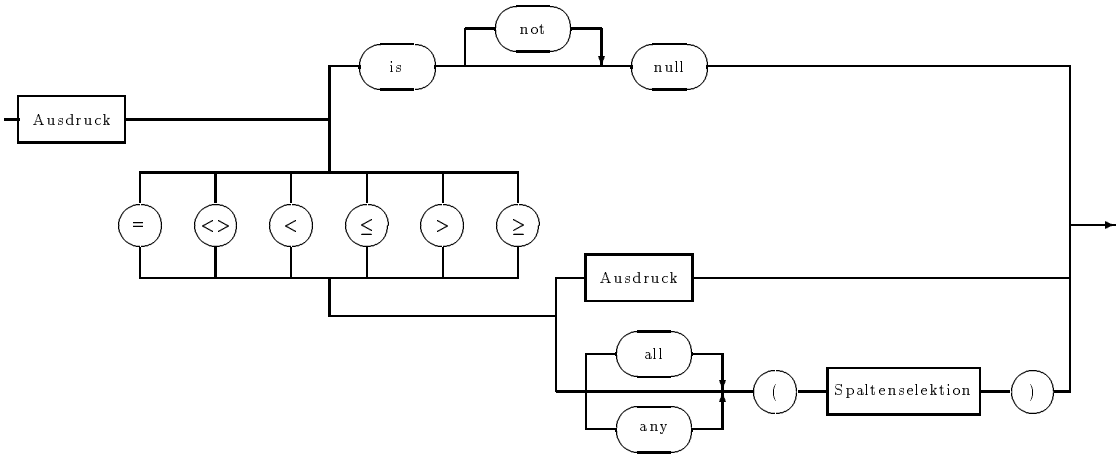
Die allgemeine Semantik für ein SELECT-Kommando lautet



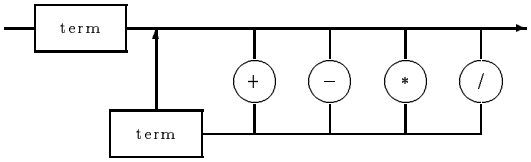
wobei Bedingung für



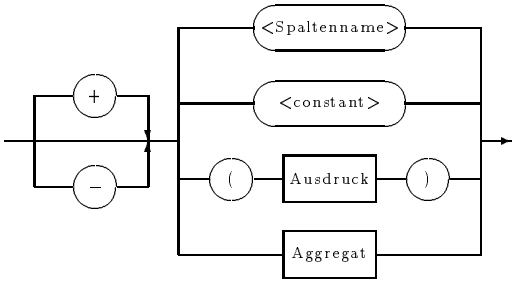
steht. Der Ausdruck Vergleichs-Bed. wiederum steht für



Dabei ist Ausdruck von der Gestalt



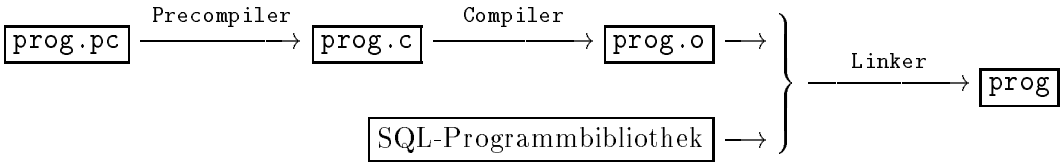
und Term steht für



Die Sprache SQL arbeitet mengenorientiert. Die Tupel werden anhand der Einträge angesprochen. In C hingegen werden Variablen oder Strukturen über ihre Adressen angesprochen. Werden Werte aus C-Variablen in SQL übernommen, so geschieht dies mittels Einfügeoperationen. Soll umgekehrt aus einer Tabelle ein Tupel in C benutzt werden, so muß für dieses Tupel einzeln die Zuweisung an Variablen erfolgen, die über Adressen ansprechbar sind.

Übersetzung

Das Quellprogramm enthält neben C-Anweisungen auch SQL-Anweisungen. Der Precompiler sucht aus dem Quellprogramm die SQL-Anweisungen heraus und ersetzt sie durch den Aufruf von C-Bibliotheksfunktionen.



In einer Declare-Section werden Variablen sowohl dem C-Teil als auch dem SQL-Teil bekannt gemacht.

```
exec SQL begin declare section;
     int ...
     char ...
exec SQL end declare section;
```

Ein einfaches Einfügen geschieht zum Beispiel folgendermaßen

```
exec SQL execute immediate
     insert into Aufträge
     values (1027,´Jan. 7´,´Schmidt´);
```

oder um ein Tupel aus C-Variablen einzufügen

```
exec SQL execute immediate
     insert into Aufträge
     values (:Auftragsnummer,:Datum,:Name);
```

Ein prepare-Statement bewirkt, daß ein Statement nicht bei jedem Auftreten interpretiert werden muß.

```
exec SQL begin declare section
     int Auftragsnummer,Menge;
     char Datum[10],Name[20],Teil[10];
exec SQL end declare section;
exec SQL connect; /* vor auszuführendem SQL-Aufruf */
exec SQL prepare Auftr_Eing from /* Vorbereiten der Einfüge-Operation */
     insert into Aufträge
     values (:Auftragsnummer,:Datum,:Name);
exec SQL prepare Einfügen from
     insert into Enthält
     values (:Auftragsnummer,:Teil,:Menge);
Schreib("Eingabe von Auftragsnummer, Datum und Kunde: ");
Lies(&Auftragsnummer);
Lies(Datum);
Lies(Name);
exec SQL execute Auftr_Eing using :Auftragsnummer,:Datum,:Name;
Schreib("Eingabe von Teil-Mengen-Paaren (Abschluß mit ´Ende´: ");
Lies(Teil);
while (strcmp(Teil,"Ende"))
{
   Lies(&Menge);
   exec SQL execute Einfügen using :Auftragsnummer,:Teil,:Menge;
   Lies(Teil);
}
```

Um eine aus SQL zurückgegebene Tabelle in C zu verarbeiten, wird eine Tupel-Variable eingeführt,

die die Tabelle wie ein Cursor durchläuft.

```
exec SQL prepare S from ⟨select_statement⟩;
exec SQL declare C cursor for S;
exec SQL open C;
exec SQL whenever notfound goto Ende;
for (;;)
{
   exec SQL fetch C into :A1,:A2,...,:An ;
   ... /* Hier kann nun das Wertetupel (A1,A2,...,An) verarbeitet werden */
}
Ende:
exec SQL close C;
```

4.2 Algebraische Optimierung von Anfragen

Wir haben zwei Möglichkeiten, die Ausführung von Datenbank-Anfragen zu beschleunigen

a) Nur Benutzung der relationalen Algebra, und

b) Hinzunahme von Indexen, die im Hauptspeicher abgelegt werden.

Zunächst wollen wir uns mit der Nutzung der Möglichkeit a) befassen.

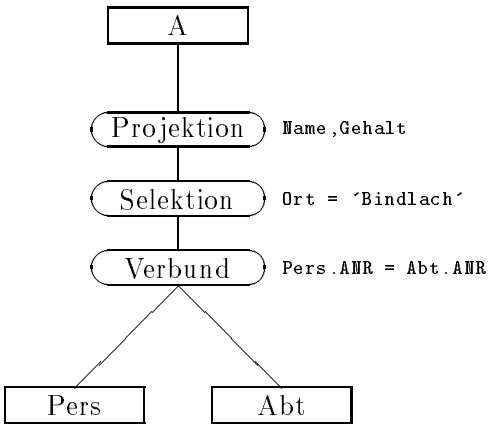
Die Ausdrücke werden durch Bäume dargestellt. Die Blätter stellen die Relationen (Tabellen) dar, auf die zugegriffen wird. Die inneren Knoten stellen einzelne Operationen dar, mit denen die aus den Nachfolgern bestehenden Datenmenge zu verarbeiten sind. Die Wurzel ist die Ergebnis-Relation.

Beispiel

Wir haben die Relationen `Pers` mit Attributen `ANR`, `Name`, `Gehalt`, `Tel`, `Kind` und `Beruf` sowie `Abt` mit Attributen `ANR`, `Ort`, `Manager` und `Aufgabe`. Es soll folgende Anfrage beantwortet werden "Finde Name und Gehalt aller Angestellten, die in Bindlach arbeiten".

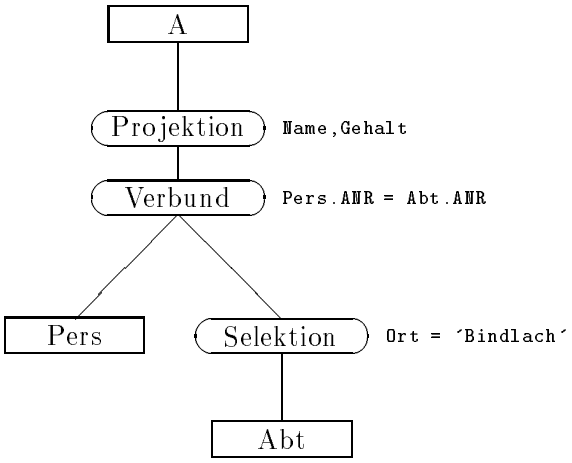
```
SELECT Name,Gehalt  ← Projektion
FROM Pers,Abt
WHERE Pers.ANR = Abt.ANR  ← Verbund
AND Ort = ´Bindlach´  ← Selektion
```

Ein Anfrage-Baum sieht folgendermaßen aus

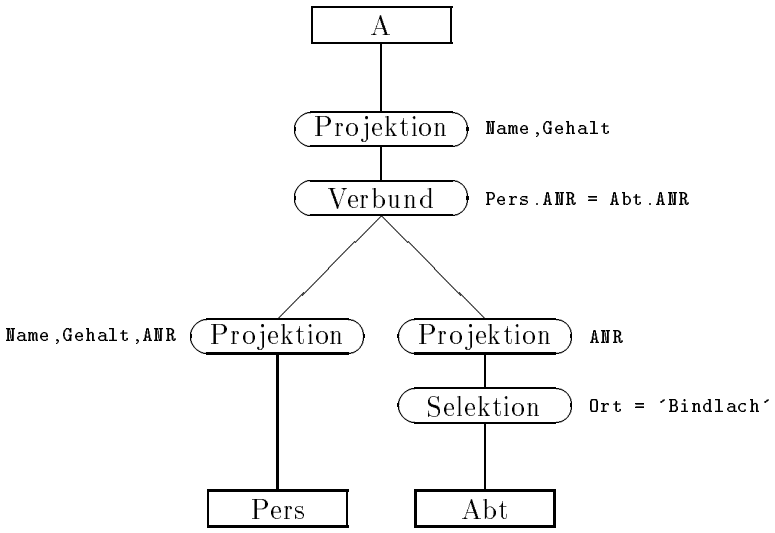


Direkte Abarbeitung dieses Baumes, wie er sich zum Beispiel aus einem Compiler ergeben würde, verursachte beim Verbund einen Aufwand von $O(|\texttt{Pers}| \cdot |\texttt{Abt}|)$.

Durch eine Veränderung der Reihenfolge der Operationen kann der Aufwand verringert werden. Dazu ist der Baum in einen anderen zu transformieren, der weniger Aufwand bei der Abarbeitung erfordert.



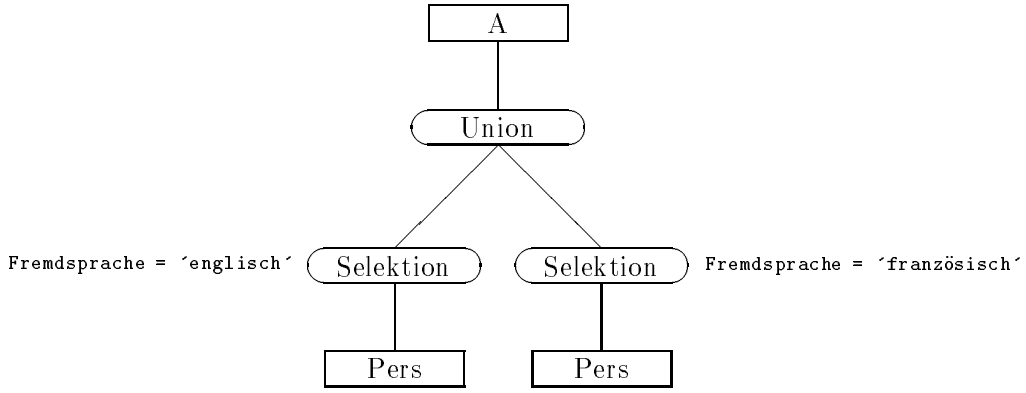
Um die Projektion mit Verbund zu vertauschen, muß in die Projektion das Verbundattribut mit aufgenommen werden. Die ursprüngliche Projektion ist wiederum auszuführen, um das Verbundattribut zu eliminieren.



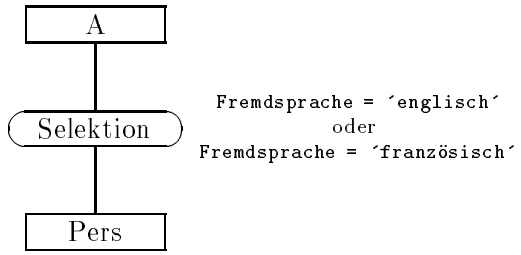
Beispiel

In der Tabelle `Pers` aus dem vorausgegangenen Beispiel sei zusätzlich noch das Attribut

`Fremdsprache` vorhanden. Wir suchen alle englisch oder französisch sprechenden Mitarbeiter.



Hier erhalten wir kürzer

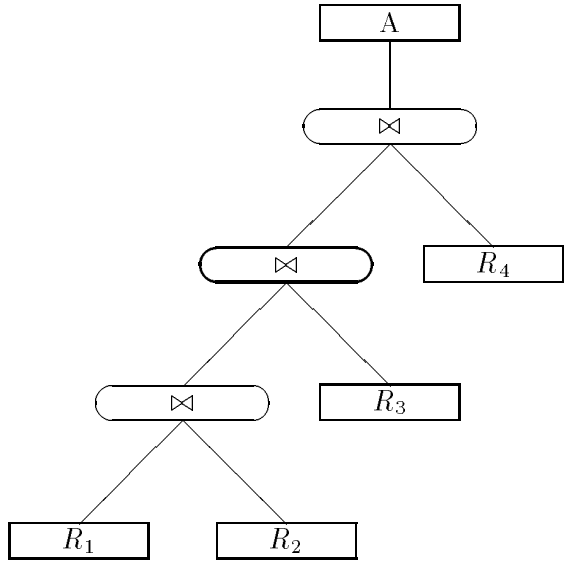


Assoziative Operatoren

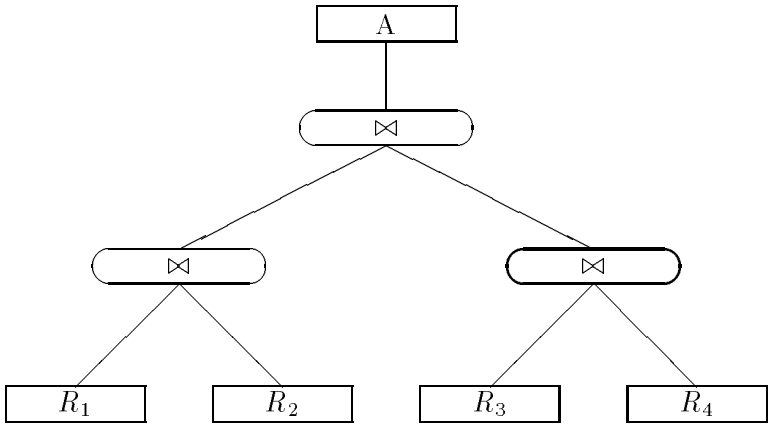
Das Ergebnis der Operation $R_1 \bowtie R_2 \bowtie R_3 \bowtie R_4$ ist von der Klammerung unabhängig. Bezüglich des Aufwands ergeben sich jedoch Unterschiede.

Beispiel

Für die Berechnung von $((R_1 \bowtie R_2) \bowtie R_3) \bowtie R_4$ erhält man



Für die Berechnung von $(R_1 \bowtie R_2) \bowtie (R_3 \bowtie R_4)$ erhält man hingegen



Die Größe der Zwischenergebnisse (in obigen Bildern fett gezeichnet) entscheidet darüber, welcher Baum günstiger ist.

Heuristik für die Abschätzung der Größe der Zwischenergebnisse

Es wird angenommen, daß bei allen vier Relationen der Verbund über dasselbe Attribut mit j gleichwahrscheinlichen Werten gebildet wird, das heißt

$$j \leq |R_1| \leq |R_2| \leq |R_3| \leq |R_4| \,.$$

Zum Beispiel hat $R_1 \bowtie R_2$ ca. $|R_1| \cdot \frac{|R_2|}{j}$ Tupel. Insgesamt erhalten wir die Entscheidung über die Wahl der besseren Alternative aus der Ungleichung

$$\frac{|R_1| \cdot |R_2|}{j} \cdot \frac{|R_3|}{j} \leq \frac{|R_3| \cdot |R_4|}{j}$$

und damit einfacher aus

$$\frac{|R_1| \cdot |R_2|}{j} \leq |R_4| \,.$$

Ähnliches läßt sich für "$\cup$" und "$\cap$" durchführen.

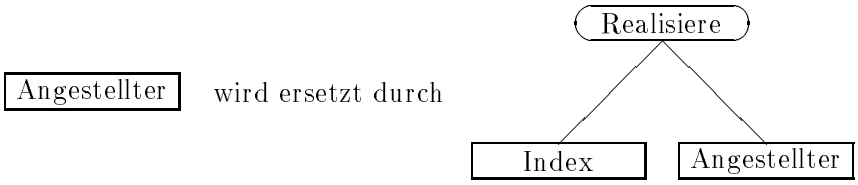
Regeln zur Optimierung

1) Projektionen werden soweit wie möglich zu den Blättern gezogen.
2) Selektionen werden soweit wie möglich zu den Blättern gezogen.
3) Selektionen, die die gleiche Relation betreffen, sowie entsprechende Projektionen werden zusammengefaßt.
4) Bei Folgen von Verbund- und Mengenoperationen wird die Reihenfolge nach der Mächtigkeit der beteiligten Relationen festgelegt.
5) Eventuell auftretende gemeinsame Teilbäume sind als ein gemeinsamer Baum zu berechnen.

Die zweite Möglichkeit zur algebrarischen Optimierung von Anfragen besteht in der Optimierung der Bäume für die relationalen Ausdrücke unter Berücksichtigung der Zugriffswege.

4.3 Methoden zur Beschleunigung des Zugriffs

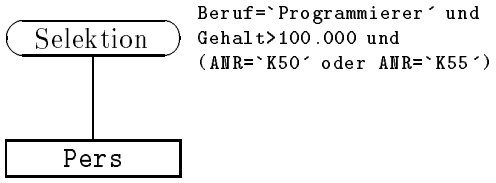
Aufspaltung eines Blattknotens für alle Relationen durch Anwendung eines Operators "Realisiere" auf den Zugriffsweg und die Relation



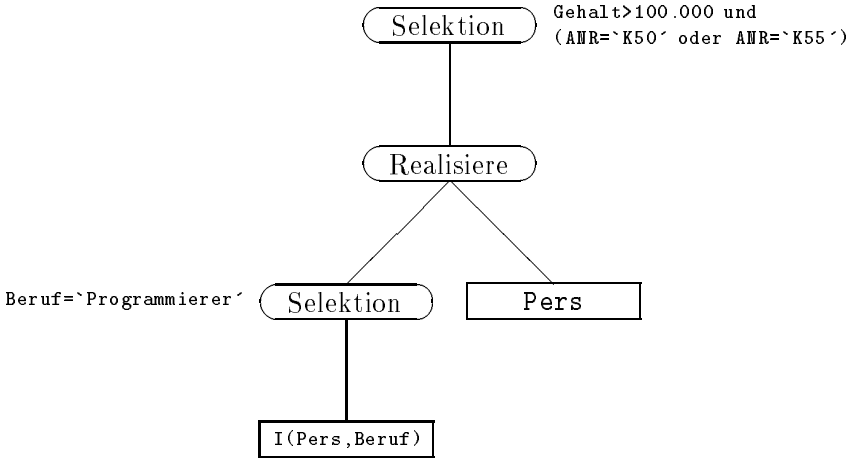
Die Selektionen und Projektionen können auch hier mit dem Operator Realisiere vertauscht werden. Somit ist der Baum formal transformierbar in einen Baum mit günstigerem Zugriffsverhalten.

Beispiel

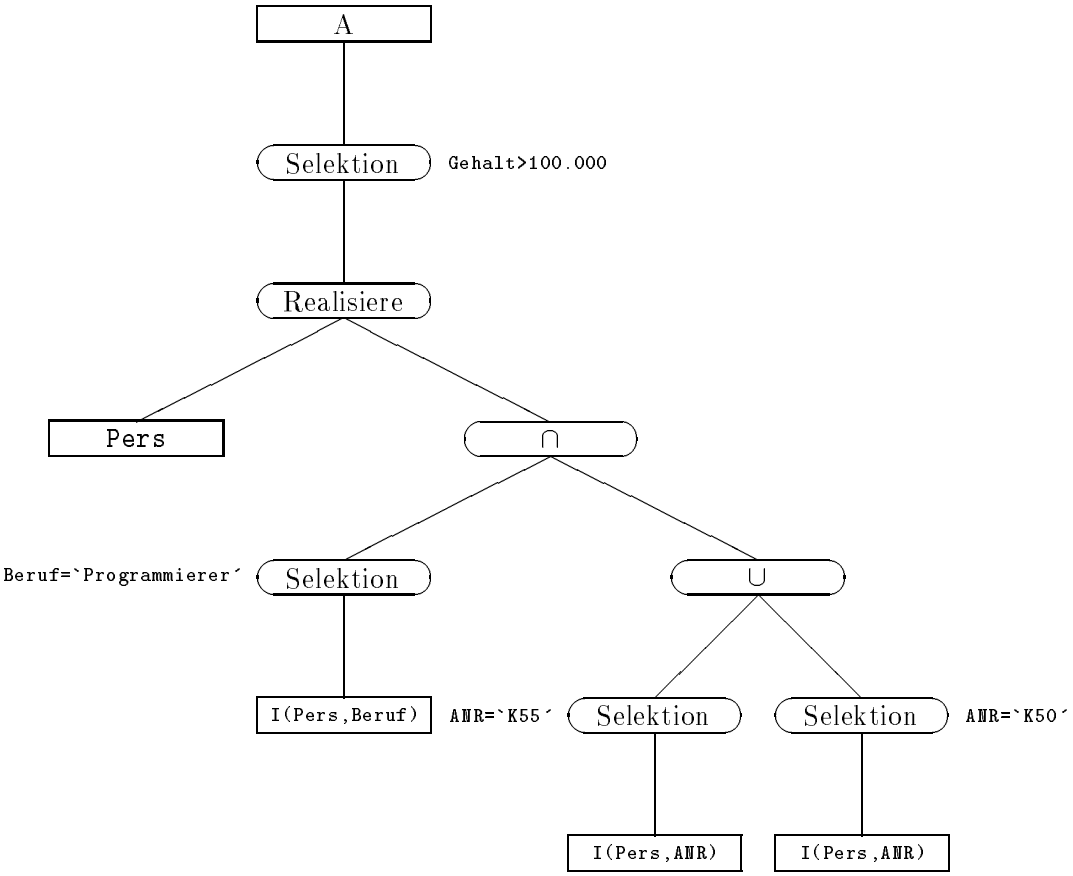
Gesucht sind alle Programmierer mit einem Gehalt von mehr als 100000 (Mark) und mit einer bestimmten Angestelltennummer, das heißt



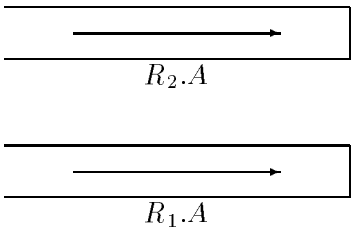
Sei bezüglich des Attributs `Beruf` ein Index I(`Pers`,`Beruf`) vorhanden. Dann können wir die obige Anfrage umtransformieren zu



Ist weiterhin ein Index I (`Pers`, `ANR`) vorhanden, so verwende

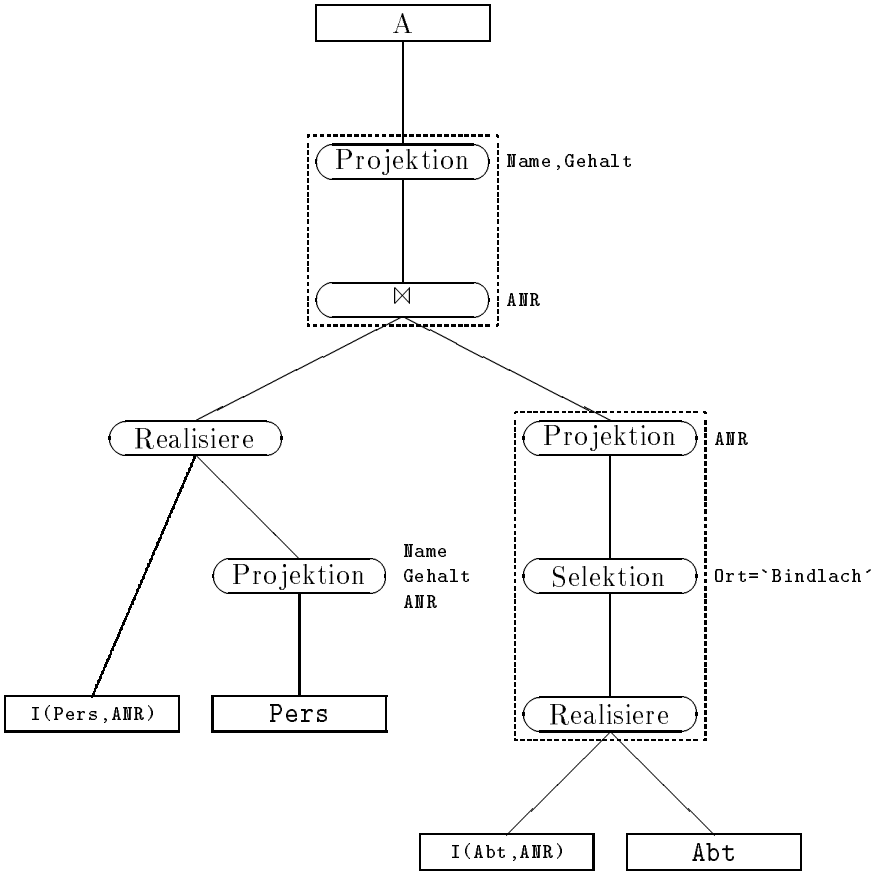


Wir brauchen also die Vereinigung zu dem Attribut `A` mit zwei Relationen, die beide einen Index bezüglich `A` besitzen. Der Index liefert gerade die Adressen der Tupel, sortiert nach dem Wert des Verbundattributes.



Beim aufsteigenden Durchlauf zweier sortierter Felder kann mit linearem Aufwand paarweise ver-

glichen werden (siehe Mergesort). Wir erhalten also für das Beispiel von Seite 17



Operationen, die dieselbe Datei oder das selbe Zwischenergebnis betreffen, lassen sich durch einen entsprechend mächtigen Operator zusammenfassen. Dadurch brauchen die Dateien nicht mehrfach durchlaufen werden.

Ist kein Index vorhanden, so sortiere bezüglich des Verbundattributs.

Aus Informatik II wissen wir, daß n Daten mit $o(n \log n)$ Zeitaufwand sortiert werden können. Der Aufwand, um R_1 und R_2 zu verbinden, beträgt ohne Sortierung $|R_1| \cdot |R_2|$. Mit Sortierung hingegen ist der Aufwand

$$\underbrace{c_1 \cdot |R_1| \cdot \log |R_1|}_{\text{Sortiere } R_1} + \underbrace{c_2 \cdot |R_2| \cdot \log |R_2|}_{\text{Sortiere } R_2} + \underbrace{|R_1| + |R_2|}_{\text{Durchlaufe linear wie bei Index}}$$

Damit ist das Sortieren in den meisten Fällen günstiger.

Daher stellt man einen Sortieroperator zur Verfügung, der dann eingesetzt werden kann, wenn kein Index vorhanden ist. Beim Sortieren können doppelte Records leicht erkannt werden. Vor einer Verbundoperation sind die Doubletten zu entfernen. "Realisiere" erhält die Daten, die bezüglich eines Attributs sortiert sein können.

Beispiel

Sei $(a_1, \ldots, a_n)$ eine bestimmte Reihenfolge, und die Daten wie folgt auf der physikalischen Ebene gespeichert.

a_1, a_4, a_6	a_2, a_5	a_3

Es läßt sich nun vermeiden, dieselbe Datenseite mehrfach von der Platte einzulesen, indem man die physikalische Adresse als Sortierkriterium verwendet und die Liste der zu holenden Tupel

danach sortiert. Wenn man die durch Sortieren bewirkte Permutation wieder rückgängig macht, so erhält man die ursprüngliche Reihenfolge zurück.

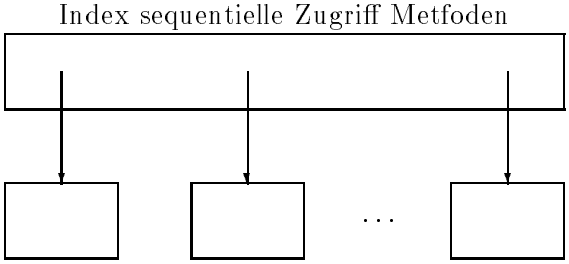
5. Die physikalische Ebene

Zur Speicherung auf der pysikalischen Ebene werden die Verfahren verwendet, die in Informatik II vorgestellt wurden. Schlagworte hierfür sind

- Sortierte Daten,
- binäres Suchen,
- interpolierendes binäres Suchen,
- Hashfunktionen,
- B-Bäume,
- k-d-Bäume und
- Gridfile.

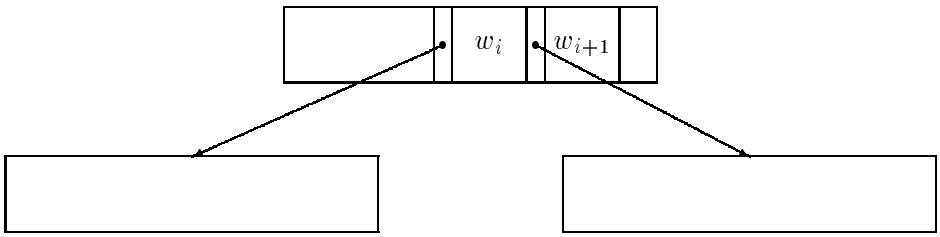
5.1 Sortierte Daten

Die Daten werden blockweise sortiert auf Platte abgelegt. Beim Initialisieren wird jeder Block zu 80% ausgelastet. Es wird ein Index für die Blockanfänge angelegt.



5.2 B-Bäume

Die Datenstruktur der Bayer-Bäume wurde bereits in Informatik II behandelt. Ein Knoten des B-Baumes enthält jeweils Vergleichswerte und Nachfolgerzeiger.



Wenn in den Indexseiten des B-Baumes bereits vollständige Datensätze liegen, so benötigen diese viel Platz. Daher passen weniger Zeiger und Vergleichswerte auf eine Indexseite, und somit wächst die Höhe des Baumes und die Anzahl der Seitenzugriffe. Besser geeignet sind daher die sogenannten B*-Bäume.

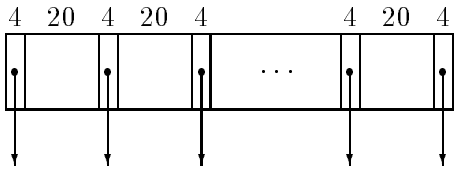
5.3 B*-Bäume

Hier werden nur in den Blättern vollständige Datensätze gespeichert. Die Indexseiten enthalten neben den Zeigern nur Vergleichswerte.

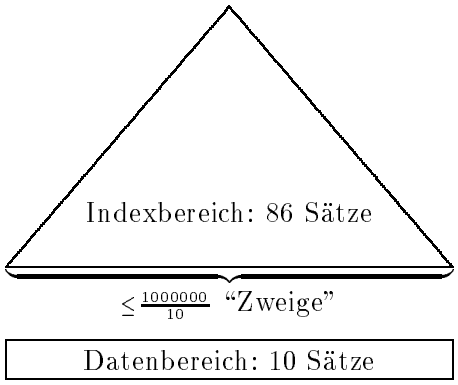
Beispiel

Zu speichern sind 1000000 Datensätze zu je 200 Bytes. Die Blockgröße sei 4096 Byte. Im B-Baum stehen pro Datenseite mindestens 10 und höchstens 19 Datensätze. Die Höhe des

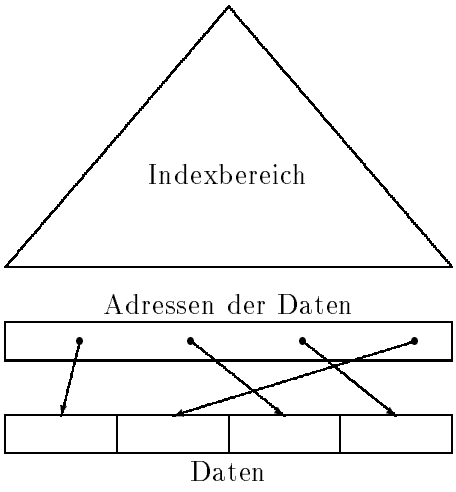
Baumes ist also $1 + \log_{10}(1000000) = 7$. Das Schlüsselwort habe 20 Byte, ein Zeiger 4 Byte. Die Indexseiten haben 4096 Byte, also passen insgesamt 170 Vergleichswerte und 171 Zeiger, $(170 \cdot 20 + 171 \cdot 4 = 4084)$ auf eine Indexseite.



Somit können mindestens 86 und höchstens 171 Nachfolgerzeiger in einem Knoten gespeichert werden. In den Blättern stehen 1000000 Sätze. Die Höhe ist also $2 + \log_{86} \frac{1000000}{10} < 5$, was der Anzahl der Seitenzugriffe pro Anfrage darstellt.



B*-Bäume in denen in den Blättern Zeiger stehen



Im B*-Baum werden nur die Adressen der Daten abgelegt. Somit können die Seiten bei den Daten voll ausgenutzt werden. Das bringt eine Ersparnis von ca.25% abzüglich 12% für die Adressebene, also werden etwa 13% gewonnen. Die Speicherung kann neu organisiert werden, es brauchen nur die entsprechenden Adressen überschrieben werden.

Problem bei mehreren Zugriffswegen

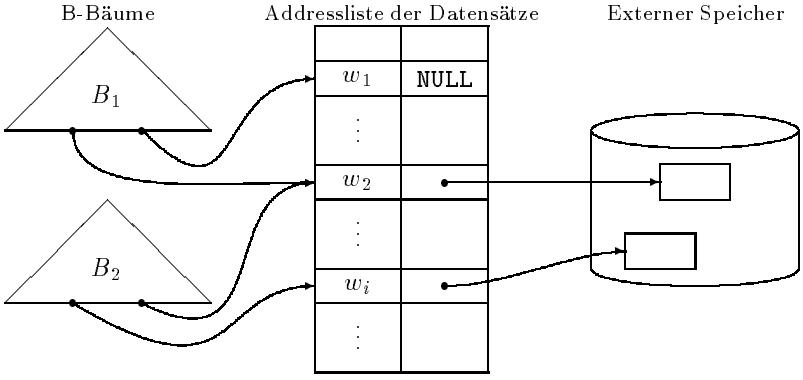
Es existieren verschiedene Zeiger, die auf denselben Datensatz zeigen. Von diesem Datensatz aus gesehen ist jedoch nicht bekannt welche Zeiger auf ihn zeigen, daher ergeben sich beim Löschen dieses Datensatzes Komplikationen.

Erste Lösung

Beim Datensatz wird ein Löschkennzeichen eingeführt, wodurch sich auf jedem Zugriffweg feststellen läßt, ob der Datensatz gültig ist oder nicht. Allerdings führt diese Methode langfristig zu einer Speicherplatzverschwendung, es ist daher von Zeit zu Zeit eine Reorganisation der Datenbank nötig, das heißt es wird eine neue Datenbank ohne die gelöschten Sätze angelegt.

Zweite Lösung

Um den Speicherplatzverbrauch für gelöschte Datensätze zu verrringern, kann man die eigentlichen Datensätze auf der untersten Ebene durch Zeiger auf die Datensätze ersetzen.



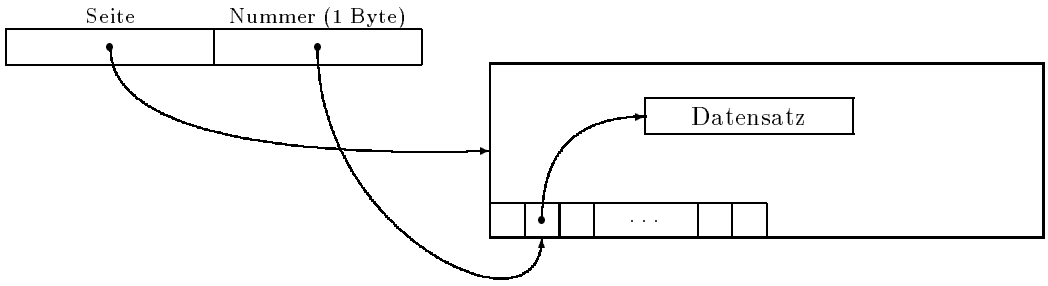
Somit ist es möglich den physikalischen Speicherplatz zu verschieben oder zu löschen.

Als Alternative könnte man bei den Sekundärindexen statt der Datensätze den Schlüsselwert, unter dem der Datensatz im Primärindex gefunden wird, ablegen. Der Zugriff bezüglich des Sekundarindexes erfordert dann allerdings den Durchlauf beider Zugriffswege und ist somit zeitaufwendiger.

5.4 TID-Konzept (Tupel-Identifier)

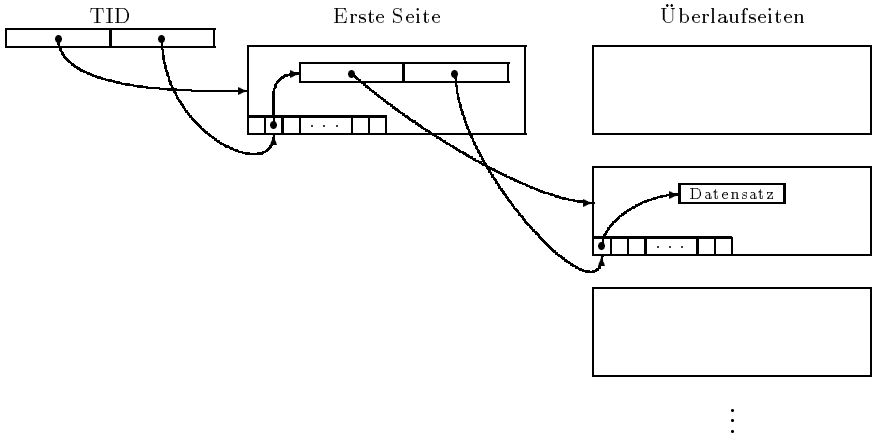
Ein Datensatz wird durch eine Adresse beschrieben, die sich in zwei Teile aufteilt

1. Adresse der Datenseite
2. Nummer der Adresse des Datensatzes in einem Adressfeld auf der Seite



Überlauf einer Seite

Statt der Datensätze können auf der Datenseite wieder TIDs stehen.



Für alle realistischen Situationen wird bei Überlaufseiten für das Aufsuchen eines Datensatzes höchstens ein weiterer Seitenzugriff erforderlich.

5.6 Dynamisches Hashen

Gegeben ist die Hashfunktion $h: W \to A$, dabei ist $h(w)$ die Adresse für den Datensatz (oder die Datenseite), der beziehungsweise die mittels w gefunden werden soll.

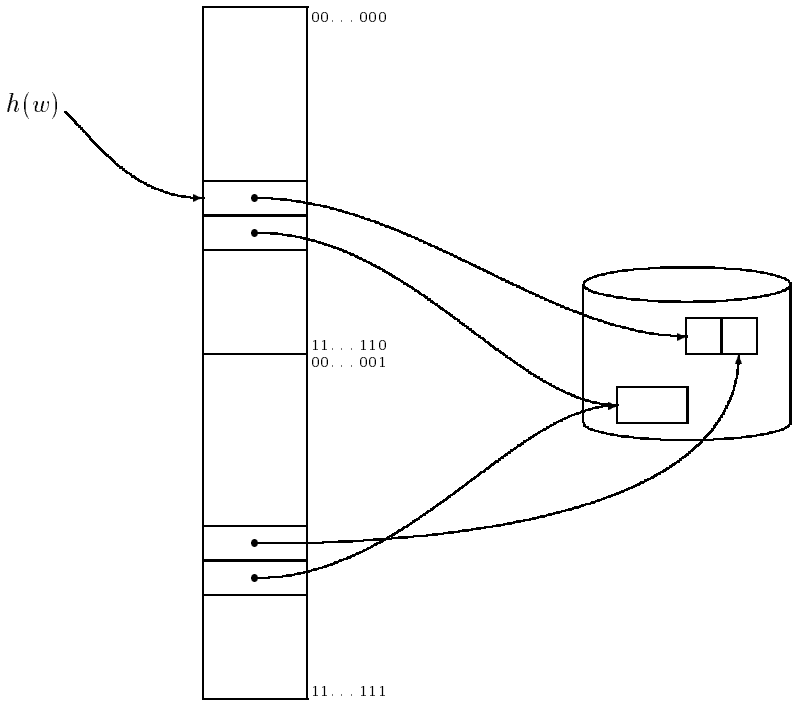
Die Datenmenge kann stark wachsen, damit wächst die Anzahl der Kollisionen ebenfalls stark und der Zugriff auf einen Datensatz erfordert viele Seitenzugriffe.

Ausweg

Es wird zu Beginn nur ein Teil der berechneten Hashadresse zur Datenadressierung verwendet.

$$h(w) = \underbrace{a_n a_{n-1} \ldots a_{i+1}}_{\text{Hashadresse}} | a_i \ldots a_1 a_0$$

Der Teil $a_n a_{n-1} \ldots a_{i+1}$ wird als Hashadresse verwendet. Treten zu viele Seitenzugriffe auf, so nehme eine weitere Stelle hinzu.



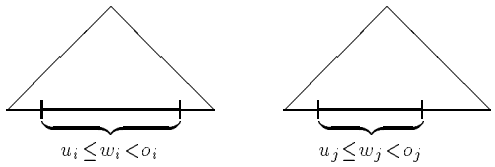
Berechne für die übergelaufene Seite alle Hashwerte neu und teile die Seite entsprechend dieser Adresse in die alte und eine neue Seite auf. Kopiere das vorhandene Adressfeld (00...000 bis 11...110) in den neu angelegten Bereich (00...001 bis 11...111) und überschreibe für die Überlaufseite den Zeiger auf die alte Seite mit der neuen Adresse.

Läuft eine Seite über, auf die bereits zwei Zeiger zeigen (dies ist durch Änderung einer Stelle in der Hashadresse und Zeigervergleich feststellbar), so braucht das Adressfeld nicht verdoppelt zu werden. Es wird nur die Überlaufseite entsprechend der Hashadresse gespalten und der eine Zeiger im Adressfeld auf die neue Seite umgebogen.

5.7 Mehrdimensionale Anfragen

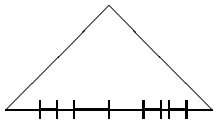
Es seien Datensätze $w = (w_1, w_2, \ldots, w_k)$ mit k Attributen gegeben. Gesucht werden alle Datensätze, die Bedingungen der Form $u_i \leq w_i < o_i$ für $i = 1, \ldots, k$ genügen. Die Daten seien in B-Bäumen gespeichert. Dann gibt es zwei Möglichkeiten, eine solche Anfrage zu realisieren.

a) Für jede Komponente existiert ein Baum



Alle Bedingungen sollen zugleich erfüllt sein, daher ist der Schnitt der Lösungsmengen zu betrachten. Das Problem bei dieser Methode ist, daß die Zwischenlösungen eventuell sehr umfangreich sein können.

b) Es existiert ein B-Baum für jede Kombination von Attributen (z.B. in SQL Index bzgl. mehrerer Attribute mit Reihenfolge und Sortierrichtung)



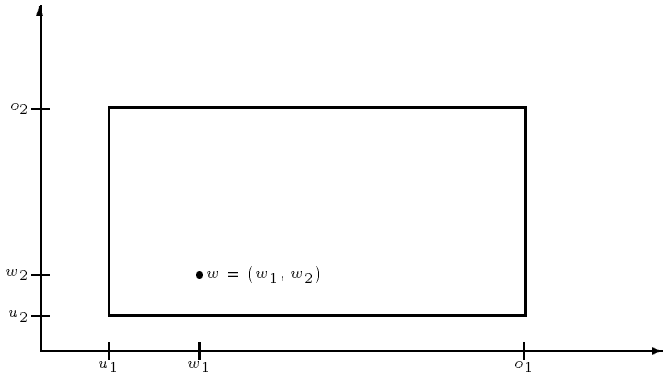
Hier gibt es keine überflüssigen Zwischenergebnisse, aber bei k Attributen gibt es 2^k Teilmengen der Attributmenge. Wenn man nicht vorher weiß, in Bezug auf welche Kombination von Attributen die Anfrage formuliert wird, würden bei dieser Lösung maximal 2^k Zugriffwege benötigt. Somit ist diese Lösung nur für spezielle häufig auftretende Anfragen brauchbar.

Wir interpretieren nun die Datensätze als Punkte in einem Vektorraum. Es ist

$$w = (w_1, w_2, \ldots, w_k) \in W_1 \times W_2 \times \ldots \times W_k.$$

Ohne Beschränkung der Allgemeinheit sei $W_i = \mathbb{R}$ für alle i, damit ist w ein Punkt im $\mathbb{R}^k$. Für $k = 2$

erhalten wir die folgende Veranschaulichung



Die Suchanfrage

$$u_1 \leq w_1 \leq o_1$$
$$u_2 \leq w_2 \leq o_2$$

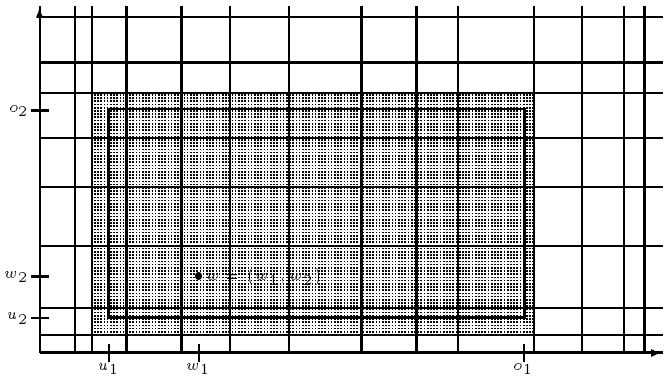
beschreibt die Datenpunkte, die in einem Quader im Datenraum liegen.

Dies liefert uns nun einen anderen Ansatz für mehrdimensionale Abfragen.

5.7.1 Gridfile

Zerlege den Datenraum in Quader. Alle Punkte in einem solchen werden auf derselben Datenseite abgelegt. Die Intervallgrenzen müssen dazu entsprechend gewählt werden, damit die Daten gleichmäßig über die Datenseiten verteilt werden.

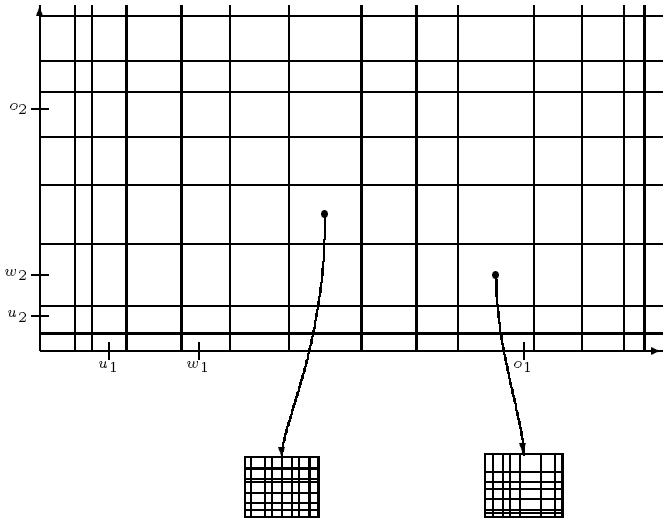
Bei einer Suchanfrage wird ein Suchquader definiert. Die darin liegenden Datenpunkte sind auf den Datenseiten der den Suchquader schneidenden Datenquader zu finden.



Es werden nur wenige Daten geholt, die der Anfrage nicht genügen.

Dieses Vorgehen ist allerdings nur bei statischen bekannten Datensätzen sinnvoll. Ziel ist eine mehrdimensionale dynamische Datenstruktur, die beim Einfügen, Löschen und beim Zugriff mit dem Bayerbaum vergleichbar ist. Verfeinere dazu schrittweise eine grobe Zerlegung des Raumes,

solange bis die in den Quadern liegenden Datensätze jeweils auf eine Seite passen.



Ein einzelner Knoten sieht darin wie folgt aus

k	$n_1 \ldots n_k$	$w_{11} \ldots w_{1n_1}$	$\cdots$	$w_{k1} \ldots w_{kn_k}$	$n_1 \cdot \ldots \cdot n_k$ Zeiger auf Nachfolger

Beispiel

Die Seitengröße betrage 4096 Bytes und ein Zeiger benötige 4 Bytes. Pro Komponente i ist eine Skala mit n_i Vergleichswerten zu speichern. Wir haben also $n_1 \cdot \ldots \cdot n_k$ Sub-Quader zu verwalten und somit $n_1 \cdot \ldots \cdot n_k$ Zeiger zu speichern. Außerdem sind die $n_1 + \ldots + n_k$ Vergleichswerte zu speichern sowie k und $n_1, \ldots, n_k$.

Sei jetzt $k = 3$, $n_1 = 10$, $n_2 = 10$ und $n_3 = 9$ und ein Vergleichswert benötige 4 Bytes, dann können wir auf einer Seite die 900 Zeiger unterbringen. Haben wir einen Baum der Höhe 2 (das heißt zwei Ebenen mit Knoten und eine Ebene mit Blättern), wobei in den Blättern 200 Datensätze gespeichert werden können, so können wir $900^2 \cdot 200 = 162\,000\,000$ Datensätze unterbringen.

Sei $w = (w_1, \ldots, w_k)$. In jeder Komponente i gibt es zu jedem w_i ein Intervall $[w_j^i, w_{j+1}^i)$ mit $w_i \in [w_j^i, w_{j+1}^i)$. Also ist die Nummer des Intervalls in der i-ten Komponente bestimmt.

Jedem Indextupel $(i_1, \ldots, i_k)$ wird ein Zeiger zugewiesen

$$f\colon [0, n_1 - 1) \times \cdots \times [0, n_k - 1) \to [0, \prod_{i=1}^{k} n_i - 1],$$

der auf dem Platz $f(i_1, \ldots, i_k)$ abgelegt wird.

$$f(i_1, \ldots, i_k) = i_1 + i_2 n_1 + i_3 n_1 n_2 + \ldots + i_k n_1 n_2 \ldots n_{k-1}$$

Behauptung: f ist bijektiv

Beweis

Der Urbildbereich und der Bildbereich sind gleichmächtig, es bleibt also nur die Surjektivität von f zu zeigen.

Sei dazu $j \in [0, \prod_{i=1}^{k} n_i - 1]$.

Induktion nach k

$k = 1$: $f = id$

$k > 1$: Sei also l maximal mit $\prod_{m=1}^{l} n_m < j$.

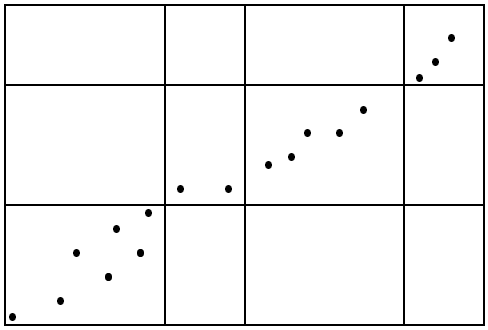
$$j = i_{l+1} \cdot \prod_{m=1}^{l} n_m + r, \qquad 0 \leq r \leq \prod_{m=1}^{l} n_m$$

Für r ist nach IV die gesuchte Darstellung bereits vorhanden.

$$j = \sum_{l=1}^{k} i_l \cdot \prod_{m=1}^{l-1} n_m$$

Damit ist ein mehrdimensionales Zeigerfeld eindimensional verwaltet! Ein Suchquader wird auf seine Komponenten projeziert. Die Endpunkte bestimmen ein Intervall, das im Allgemeinen mehrere der Skalenintervalle schneidet. Es sind alle Kombinationen von Skalenintervallen in die Suche einzubeziehen, die hier gefunden werden.

Die Struktur ist gut geeignet für gleichverteilte Daten. Bei stark korrelierten Daten ergeben sich viele Quader, die keine Daten enthalten.



Die Einteilung der Quader ist immer so zu gestalten, daß die Datenpunkte eines Quaders stets auf eine Seite passen. Für jeden Quader wird ein Zeiger benötigt, auch wenn er leer ist, das heißt keine Daten enthält. Es kann also vorkommmen, daß viele Zeiger nutzlos sind.

Ausweg

Die Zeiger brauchen nicht auf paarweise verschiedene Datenseiten zeigen!

Bedingung

Gebiete, auf die derselbe Zeiger zeigt, sollen Quader sein. Das heißt leere Blöcke werden nur dann zusammengefaßt, wenn sie Quader geben.

Die Bedingung wird beim Aufbau eines solchen Verwaltungsbaumes sichergestellt.

Aufbau

Wenn ein Quader durch Einfügen zuviele Einträge erhält, so muß er zerlegt werden. Suche dazu eine Komponente i auf. Füge in die i-te Skala in das zu diesem Quader gehörende Intervall eine neue Grenze (einen Skalenwert) ein. Zerlege alle Quader, die die gleiche Projektion auf diese Komponente haben. Bei den Quadern, die nicht überbelegt waren, wird bei den beiden entstehenden neuen Quadern derselbe Zeiger eingetragen, den der Quader vorher hatte.

Strategie für die Wahl der Komponente, in der aufgespalten wird

a) Durchlaufe die Komponenten zyklisch

b) Wähle so aus, daß die Zerlegung möglichst in gleichgroße Teile erfolgt.

Strategie zur Auswahl der Grenze

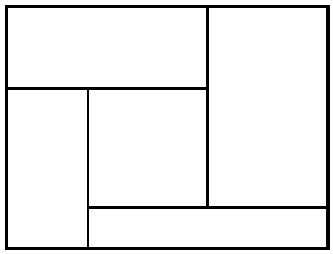
Suche Intervallmitte auf ("Buddy-System").

Löschen

Es werden Datenpunkte entfernt. Sind Seiten unterbelegt, können die entsprechenden Quader mit Nachbarn so verschmolzen werden, so daß wiederum Quader entstehen.

Verschmelzungsregel

Es sind nur solche Verschmelzungen erlaubt, die zu Situationen führen, die auch durch iteriertes Zerlegen hätten erreicht werden können.



Im Bild kommt es zu einer Verklemmungssituation, da der mittlere Quader mit keinem Nachbarn zu einem Quader verschmolzen werden kann!

Da die jeweils vorliegende Situation legal hervorgerufen wurde, ist immer ein solches Verschmelzen möglich.

Algorithmus (rekursiv)

Seien B_1, B_2 Kandidaten, die auf Verschmelzbarkeit zu testen sind.

Bilde $B := B_1 \cup B_2$.

Starte mit $R =$ ganzer Datenraum.

Wiederhole

Ist $R = B$, so gib "Verschmelzen erlaubt" aus und breche ab.

Sonst suche eine Zerlegung $R = R_1 \cup R_2$, $R_1 \cap R_2 = \emptyset$, mit $B \subset R_1$ oder $B \subset R_2$. Ersetze R durch das R_i, das B enthält. Falls diese Suche erfolglos ist, so gib "Verschmelzen verboten" aus und breche ab.

Probleme, die dabei auftreten können

a) Wertebereiche, bei denen das Datenbanksystem nicht die Mitte eines Intervalls bestimmen kann.

Somit können nur Werte verwendet werden, die bei den vorliegenden Daten auftreten. Deshalb ist das Buddy-System nicht brauchbar! Man muß deshalb aus den vorliegenden Werten geeignete auswählen. Dieses Verfahren ist aber sehr aufwendig.

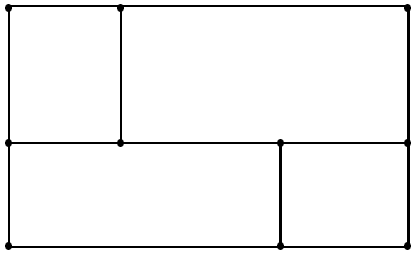
b) Doppelte Zeiger vermeiden!

Bemerkung

Die Zahl der Seitenzugriffe ist entscheidend für die Zeitkomplexität. Der Aufwand für das Behandeln von Daten auf einer Seite ist zweitrangig.

5.7.2 Verbesserter Ansatz für das Gridfile

Es ist möglich, Quader durch Angabe von Unter- und Obergrenze auf einer Seite zusammen mit dem zugehörigen Zeiger zu beschreiben.



Der Zugriff erfordert dann bei jeder Seite sequentiellen Durchlauf der Quader, um den zugehörigen Zeiger zu finden. Dieses Verfahren ist daher gut geeignet bei nicht gleichverteilten Daten.

Es tritt maximal linearer Aufwand bezüglich der Quaderanzahl auf, die Seitenzugriffe werden nicht berührt und unnütze Doppelzeiger vermieden.

Zusammenlegen von Datenseiten

Sollen Datenseiten verschmolzen werden, so entferne alle Daten aus einem umgebenden Quader des Baumes und füge sie anschließend wieder ein. Man hat eine bezüglich des Einfügealgorithmus optimale Reorganisation des betroffenen Datenbereichs.

Auffinden der Quader, die einen Suchquader schneiden

Durchlaufe die Hyperebenen, die Grenzen darstellen. Verwerfe jeden Würfel, der auf einer anderen Seite liegt als der Suchquader.

Beispiel

Es wurden Datensätzen mit 30 Komponenten benutzt, wobei 20 davon Suchkomponenten waren. 15 Komponenten hatten feste und 15 hatten variable Länge. Die gesamte Datenbank beinhaltete 100 000 Datensätze und umfasste 26 607 616 Bytes. Seitengröße war 4096 Bytes, die Baumhöhe 3. Die Seitenauslastung betrug 69% (dies ist ein typischer Wert, auch beim B-Baum erhält man Werte von 69–70%). Jeder Knoten hatte zwischen 8 und 43 Nachfolger. Das Platzverhältnis zwischen den inneren Knoten, die nur Vergleichswerte enthalten und den Datenseiten betrug ca. $1:28$.

Auf einer HP 845 wurden beim Anlegen der Datenbank 5500 Sekunden user-time und 500 Sekunden system-time gemessen. Bei einer Bereichsanfrage, bei der in jeder Komponente das Intervall $[0;10000]$ verwendet wurde, betrug die Suchzeit zwischen 20 und 120 Sekunden (im Mittel 70 Sekunden)

Anmerkung zur Verwendung variabel langer Komponenten

Die Verwendung variablel langer Datensätze stört die Theorie. Zum Beispiel kann wegen der festen Seitengröße keine Mindestanzahl von Datensätzen pro Datenseite garantiert werden.

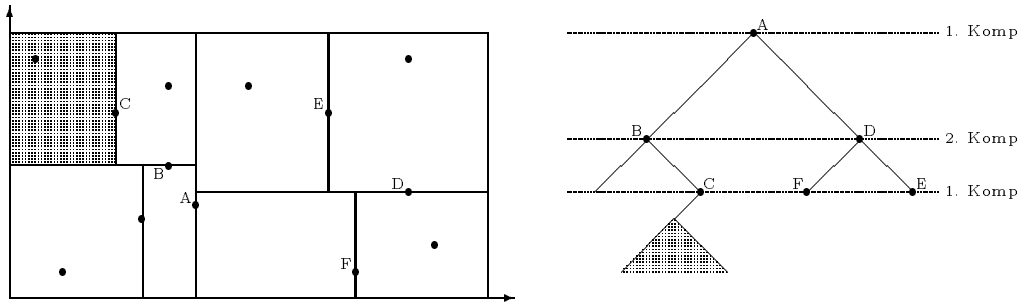
Daher muß beim Datensatz nun ebenfalls seine Länge mit abgespeichert werden.

Bei großen Datensätzen können die Seiten schnell gefüllt sein. Werden also innere Knoten mit Daten gefüllt, so haben diese häufig wenige Nachfolgerknoten. Auch aus diesem Grunde wurde die Strategie gewählt, wie beim B*-Baum die Datensätze nur in Blättern zu halten. In den inneren Knoten liegen nur Vergleichswerte aus einzelnen Komponenten.

Um zu vermeiden, daß Vergleichswerte variabler Länge mehrfach auf einer Seite gespeichert werden, kann man ein Feld von Vergleichswerten auf der Seite anlegen und nur deren Indizes verwenden.

5.7.3 k-d-Baum

Um die Quader platzsparend auf einer Seite zu beschreiben, kann man k-d-Bäume verwenden. Für zweidimensionale Datensätze läßt sich das Prinzip folgendermaßen veranschaulichen



Die Vergleichskomponenten werden zyklisch durchlaufen und bei gleichverteilten Daten hat ein k-d-Baum mit n Knoten eine Höhe von $O(\log n)$.

Beweis

Die hier verwendete Strategie ähnelt Quicksort. Siehe deshalb dortige Aufwandsabschätzung.

Beim k-d-Baum sind die Operation Einfügen und exaktes Suchen genauso wie beim eindimensionalen binären Suchbaum.

Beim Löschen allerdings kann nicht wie beim eindimensionalen Baum der kleinste Nachfolger im rechten Teilbaum (oder auch der größte Nachfolger im linken Teilbaum) an die Stelle des zu löschenden Knotens gesetzt werden, um so zu erreichen, daß ein Blatt zu löschen ist. Es muß der Vergleichswert durch den kleinsten aller Blätter im rechten Teilbaum in dieser Komponente ersetzt werden.

Eine Operation, die beim eindimensionalen binären Suchbaum nicht praktikabel ist, ist die sogenannte "partial match query". Das heißt, einen Datensatz zu suchen, von dem nur einige Komponenten bekannt sind, die übrigen sind unbekannt. Sei der Datensatz w gesucht und der aktuelle betrachtete Baumknoten enthalte einen Vergleichswert zur Komponente i.

a) Die i-te Komponente von w ist bekannt

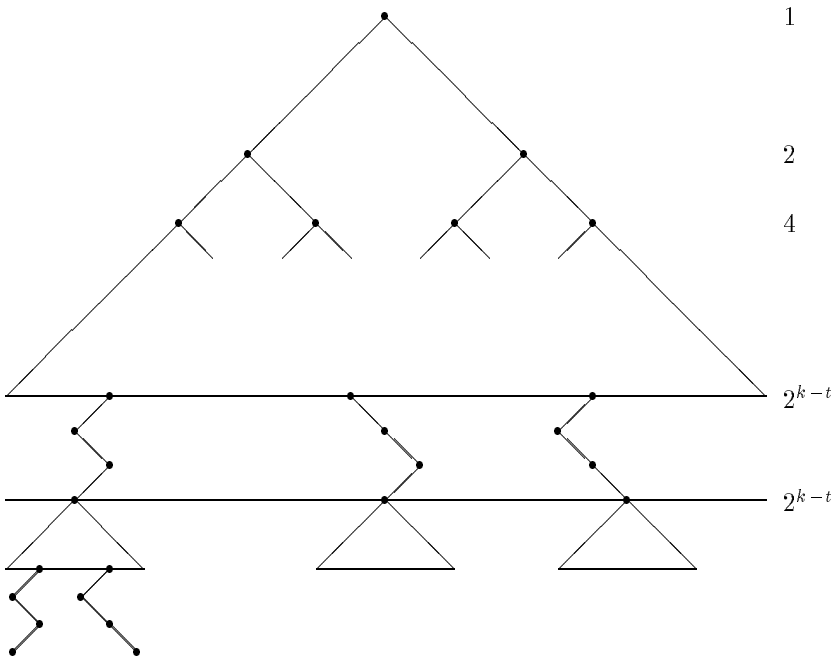
   Folge dem linken oder rechten Zeiger je nach Ausgang des Vergleichs (oder gebe den Inhalt aus, falls gefunden).

b) Die i-te Komponente von w ist unbekannt

   Es müssen beide Zeiger verfolgt werden.

Sei ein optimaler k-d-Baum gegeben mit $2^{k \cdot h} - 1 = n$ Knoten, das heißt ein Baum minimaler Höhe. Es seien t der k Komponenten in der Anfrage bekannt — wobei $t < k$ ist, da sonst eine exakte Suche vorliegt. Wir nehmen an, daß die unbekannten Komponenten die ersten $k - t$ Komponenten

sind, dies ist der schlechteste Fall, da hier die meisten Vergleiche durchgeführt werden müssen.



Für den ersten Zyklus durch die Komponenten erhalten wir für die Anzahl der Vergleiche

$$1+2+4+\ldots+2^{k-t}+\underbrace{2^{k-t}+\ldots+2^{k-t}}_{t},$$

für den nächsten

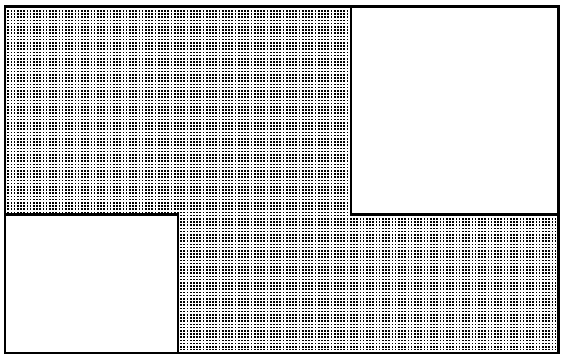
$$2^{k-t}\left(1+2+4+\ldots+2^{k-t}+2^{k-t}+\ldots+2^{k-t}\right)$$

und insgesamt ergibt sich für die Anzahl der Vergleiche

$$\begin{aligned}\sum_{i=0}^{h-1} 2^{(k-t)i}\left(\frac{2^{k-t}-1}{2-1}+t2^{k-t}\right) &= \frac{2^{(k-t)h}-1}{2^{k-t}-1}\left(2^{k-t}-1+t2^{k-t}\right)\\ &= C\cdot\left(2^{(k-t)h}-1\right) = C\cdot\left(2^{\frac{k-t}{k}hk}-1\right)\\ &= C\cdot\left(\left(2^{kh}\right)^{\frac{k-t}{k}}-1\right) = C\cdot\left((n+1)^{1-\frac{t}{k}}-1\right)\\ &= O(n^{1-\frac{t}{k}}).\end{aligned}$$

5.7.4 Lomet-Salzberg

Es befinden sich k-d-Bäume auf den Seiten. Die Strategie Bereiche aufzuteilen ist gegenüber dem Gridfile verändert. Es braucht die Form von Quadern nicht erhalten zu bleiben. Dazu werden mehrere Komponenten zugleich verwendet, um aus einem Gebiet ein Teilgebiet herauszulösen.



6. Entwurfstheorie relationaler Datenbanken

Wir wenden uns nun wieder der logischen Ebene des Datenbankentwurfs zu.

Sei $\mathcal{A} = \{A_1, \ldots, A_n\}$ eine Attributmenge und jedem $A \in \mathcal{A}$ sei ein Wertebereich $D(A)$ zugeordnet, ohne Beschränkung der Allgemeinheit sei $D(A) = \mathbb{Z}$.

Definition

Eine Relation R auf $\mathcal{A}$ besteht aus einer Menge von Abbildungen $f: \mathcal{A} \to \bigcup_{i=1}^{n} D(A_i)$ mit $f(A_i) \in D(A_i)$ für jedes $i = 1, \ldots, n$. $\mathcal{A} = \mathrm{Attr}(R)$ ist die Attributmenge von R. Die Tupel werden also durch Abbildungen beschrieben.

Anmerkung

Bei den Abbildungen ist die Reihenfolge der Attribute nicht festgelegt. Bei den Relationen ist die Reihenfolge der Abbildungen ebenfalls nicht festgelegt. Die Tabellen sind Darstellungen der Relationen.

Schreibweise

Die Abbildungen einer Relation werden "Tupel" genannt.

Definition Natürlicher Verbund

Seien R_1, R_2 Relationen. Weiter sei $\mathcal{A}_1 = \mathrm{Attr}(R_1)$ und $\mathcal{A}_2 = \mathrm{Attr}(R_2)$. Dann heißt

$$R_1 \bowtie R_2 := \left\{ f: \mathcal{A}_1 \cup \mathcal{A}_2 \to \bigcup_{A \in \mathcal{A}_1 \cup \mathcal{A}_2} D(A) \;\middle|\; f|_{\mathcal{A}_1} \in R_1 \text{ und } f|_{\mathcal{A}_2} \in R_2 \right\}$$

der *natürliche Verbund* von R_1 und R_2.

Definition Projektion

Sei R eine Relation mit $\mathcal{A} = \mathrm{Attr}(R)$. Weiter sei $\mathcal{B} \subseteq \mathcal{A}$. Dann heißt

$$\pi_{\mathcal{B}}(R) := \{ f|_{\mathcal{B}} \mid f \in R \}$$

die *Projektion* von R auf $\mathcal{B}$.

Die mengentheoretischen Operationen "$\cup$", "$\cap$" und "$\setminus$" lassen sich ebenfalls definieren.

Durch Hinzufügen und Löschen ändert sich der Inhalt einer Datenbank. Beim Datenbankentwurf wird eine zeitlich gegenüber den Änderungen invariante Definition von Relation vorgenommen. Es werden Eigenschaften formuliert, die die jeweils gültigen Relationen besitzen müssen.

$$F(\mathcal{A}) := \left\{ R \;\middle|\; R \text{ Relation auf } \mathcal{A} \text{ mit } |R| < \infty \right\}$$

Definition Semantische Bedingung, Relationsschema

Eine Abbildung $\gamma: F(\mathcal{A}) \to \{\text{wahr}, \text{falsch}\}$ heißt *semantische Bedingung* auf $\mathcal{A}$.

Sei Γ eine Menge von semantischen Bedingungen auf $\mathcal{A}$, dann heißt

$$S(\mathcal{A}, \Gamma) := \left\{ R \in F(\mathcal{A}) \;\middle|\; \gamma(R) = \text{wahr für alle } \gamma \in \Gamma \right\}$$

Relationsschema auf $\mathcal{A}$.

Beispiele

i) Prüfung der Korrektheit einer Eingabe (z.B. kein 31. Februar bei Datumseingabe)

ii) Sei die Attributmenge

$$\begin{aligned}\mathcal{A} = \{&\texttt{KundenNr}, \texttt{Name}, \texttt{Anschrift}, \texttt{VersScheinNr}, \texttt{Sparte}, \texttt{Deckung},\\ &\texttt{Abschlussdatum}, \texttt{Vertragsende}, \texttt{SchadensNr}, \texttt{Schadenshöhe},\\ &\texttt{Schadenstag}, \texttt{Auszahlungsbetrag}, \texttt{Agent}\}\end{aligned}$$

gegeben. Denkbar sind dann die semantischen Bedingungen

$$\texttt{Abschlussdatum} \leq \texttt{Schadenstag} \leq \texttt{Vertragsende}$$
$$\texttt{Auszahlungsbetrag} \leq \texttt{Deckung}$$

oder

"Wenn keine `VersScheinNr` existiert, darf auch keine `SchadensNr` angelegt werden."

Definition Funktionale Abhängigkeiten

Seien $X, Y \subseteq \mathcal{A} = \mathrm{Attr}(R)$. Eine semantische Bedingung γ geschrieben $X \to Y$ heißt *funktionale Abhängigkeit*. R erfüllt $X \to Y$ genau dann, wenn für alle $f_1, f_2 \in R$ gilt

$$f_1|_X = f_2|_X \quad \text{dann folgt} \quad f_1|_Y = f_2|_Y\,,$$

das heißt bei R sind die Einträge auf Y bereits festgelegt durch die Einträge auf X.

Beispiel

Im obigen Beispiel wären die folgenden funktionalen Abhängigkeiten denkbar

$$\{\texttt{KundenNr}\} \to \{\texttt{Name}, \texttt{Anschrift}, \texttt{Agent}\}$$
$$\{\texttt{VersScheinNr}\} \to \{\texttt{KundenNr}\}$$
$$\{\texttt{SchadensNr}\} \to \{\texttt{VersScheinNr}, \texttt{Schadenshöhe}, \texttt{Schadenstag}\}$$

Definition 1. Normalform

Eine Relation R ist in *erster Normalform*, wenn jeder Eintrag von R atomar ist, das heißt ein einzelner Wert des Wertebereichs.

Beispielsweise ist die folgende Tabelle nicht in erster Normalform

KundenNr	Name	VersScheinNr	SchadensNr
K1	Otto	4711	S1,S2,S3
		4712	S4,S5,S6
		4713	S7,S8,S9,S10

Führt man eine "Normalisierung" durch, so erhält man

KundenNr	Name	VersScheinNr	SchadensNr
K1	Otto	4711	S1
K1	Otto	4711	S2
K1	Otto	4711	S3
K1	Otto	4712	S4
K1	Otto	4712	S5
K1	Otto	4712	S6
K1	Otto	4713	S7
K1	Otto	4713	S8
K1	Otto	4713	S9
K1	Otto	4713	S10

Neuere Entwicklungen behandeln auch Datenbanken in NFNF ("Non-First-Normal-Form"), zum Beispiel das NF^2- oder das eNF^2-Modell. Hier sollen jedoch im weiteren nur Tabellen in erster Normalform betrachtet werden.

Nullwerte

Es kann vorkommen, daß einige Eintragungen nicht vorgenommen werden können, da sie bisher nicht existieren oder nicht bekannt sind. Solche Werte heißen Nullwerte und werden mit δ bezeichnet.

Beispiele

Im obigen Beispiel

- Ein Kunde der Versicherung hat keinen Schaden gemeldet.
- Ein potentieller Kunde braucht keinen Vertrag abgeschlossen zu haben.
- Eine Telefonnummer braucht nicht zu existieren.

Beispiel

Sei R die folgende Relation über $\{A, B, C\}$

A	B	C
a	δ	c

Gesucht sind alle Tupel t mit $t(B) = b$. Es gibt zwei Alternativen

a) Nur für Tupel mit $t(B) = b$ wird die Bedingung als erfüllt angesehen.

b) Auch Tupel mit $t(B) = \delta$ erfüllen die Bedingung.

Von Codd wurde eine dreiwertige Logik mit den Werten `wahr` ($\equiv$ `1`), `falsch` ($\equiv$ `0`) und δ eingeführt.

$\wedge$	0	δ	1
0	0	0	0
δ	0	δ	δ
1	0	δ	1

$\vee$	0	δ	1
0	0	δ	1
δ	δ	δ	1
1	1	1	1

$\text{not}(0) = 1$
$\text{not}(\delta) = \delta$
$\text{not}(1) = 0$

Äußerer Verbund

Bei Tupeln, die nicht in den Ausgangstabellen auftauchen, werden die Werte bei den nicht erklärten Attributen mit δ aufgefüllt.

A	B
1	4
2	2
4	1
6	1
7	3

$\otimes$

C	B
1	1
2	2
3	2
4	5

$=$

A	B	C
1	4	δ
2	2	2
2	2	3
4	1	1
6	1	1
7	3	δ
δ	5	4

(Solch ein äußerer Verbund ist zum Beispiel auch in SQL möglich)

Ein Nachteil des äußeren Verbundes ist, daß $\otimes$ nicht assoziativ ist, denn für

R_1

A	B
1	2

R_2

B	C
2	3

R_3

A	C
1	4

gilt beispielsweise

$(R_1 \otimes R_2) \otimes R_3$

A	B	C
1	2	3
1	δ	4

$R_1 \otimes (R_2 \otimes R_3)$

A	B	C
δ	2	3
1	δ	4
1	2	δ

Da im allgemeinen die Auswertreihenfolge vom Datenbank-System bestimmt wird, können hier also Probleme auftreten.

Eine alternative Möglichkeit Nullwerte zu verarbeiten ist, sie als indizierte Werte δ_1, δ_2, ... zu betrachten. Die Nullwerte sind zunächst paarweise verschieden, nur wenn geschlossen werden kann, daß zwei Nullwerte gleich sein müssen werden sie identifiziert. Zum Beispiel sind Nullwerte häufig als verschieden anzusehen, etwa Telefonnummer, Autokennzeichen, Kinder.

Für funktionale Abhängigkeiten vereinbaren wir folgendes: Eine Relation R erfüllt $X \to Y$, dann wenn aus $t_1|_X = t_2|_X$ folgt, daß die Nullwerte von t_1 und t_2 auf Y übereinstimmen.

Im weiteren werden wir alle Nullwerte als indizierte Nullwerte betrachten.

Schlüssel

Sei $S(\mathcal{A}, \Gamma)$ ein Relationenschema. $X \subseteq \mathcal{A}$ heißt Schlüssel von $S(\mathcal{A}, \Gamma)$, wenn gilt $X \to \mathcal{A}$ und für alle $A \in X$ darf $X \setminus \{A\} \to \mathcal{A}$ nicht von allen $R \in S(\mathcal{A}, \Gamma)$ erfüllt werden.

Schlüssel können verschieden viele Attribute enthalten.

Sei zum Beispiel $\mathcal{A} = \{A, B, C, D\}$ und $\Gamma = \{ABC \to D, BD \to A, ABD \to C\}$, dann sind in $S(\mathcal{A}, \Gamma)$ ABC und BD Schlüssel. Für ABC zeigt man dies so: Es gilt $ABC \to \mathcal{A}$, aber nicht $AB \to \mathcal{A}$, $AC \to \mathcal{A}$ oder $BC \to \mathcal{A}$, da zum Beispiel die Tabelle

A	B	C	D
1	1	0	0
1	1	1	1

Γ erfüllt aber nicht $AB \to \mathcal{A}$. Für $AC \to \mathcal{A}$ und $BC \to \mathcal{A}$ lassen sich ähnliche Gegenbeispiele finden.

Die Anzahl der Schlüssel kann exponentiell wachsen.

Beispiel

Sei wieder die Attributmenge

$$\mathcal{A} = \{\texttt{KundenNr}, \texttt{Name}, \texttt{VersScheinNr}, \texttt{Sparte}, \texttt{Deckung}, \texttt{SchadensNr}, \texttt{Schadenstag}\}$$

und die folgende Tabelle gegeben

KundenNr	Name	VersScheinNr	Sparte	Deckung	SchadensNr	Schadenstag
1	Otto	4711	Kfz	1000000	1	1.1.90
1	Otto	4712	Kfz	1000000	2	2.1.90
1	Otto	4711	Kfz	1000000	3	3.1.90
2	Hinz	15	Kfz	1000000	---	---
3	Hinz	16	Kfz	1000000	---	---

Der einzig denkbare Schlüssel wäre `SchadensNr`, aber bei einem Schlüssel dürfen keine Nullwerte eingetragen werden.

Aus einer Menge Γ funktionaler Abhängigkeiten läßt sich eine Menge Γ^+ funktionaler Abhängigkeiten bestimmen, die logisch impliziert werden.

Beispiel

Sei $\mathcal{A} = \{A, B, C\}$ und $\Gamma = \{A \to B, B \to C\}$. In Γ^+ liegen zum Beispiel

$$A \to C, A \to BC, A \to AB, A \to A, A \to \emptyset, A \to AC, A \to ABC$$
$$B \to BC, B \to \emptyset, \ldots$$
$$C \to C, C \to \emptyset,$$
$$AB \to C, \ldots$$

Schwierigkeiten bei großen Schemata

- Informationen, die keinen Wert beim Schlüssel besitzen, können nicht gespeichert werden.
- Informationen, die an andere gekoppelt gespeichert sind, werden beim Löschen der anderen Informationen mitgelöscht.
- Dieselbe Information kann häufig gespeichert werden müssen.

Daraus ergibt sich ein hoher Speicherverbrauch und Redundanz.

Eine Information soll möglichst nur an einer Stelle gespeichert werden. Bilde also mehrere Relationenschemata für die verschiedenen Typen von Informationen.

Definition

Seien $S_1(\mathcal{A}_1, \Gamma_1), \ldots, S_n(\mathcal{A}_n, \Gamma_n)$ Relationenschemata und $\gamma\colon S_1 \times \cdots \times S_n \to \{\texttt{wahr}, \texttt{falsch}\}$ eine semantische Bedingung auf $(S_1, \ldots, S_n)$.

Ist Γ eine Menge semantischer Bedingungen auf $(S_1, \ldots, S_n)$, so heißt

$$\begin{aligned}\texttt{DS} &= \texttt{DS}(S_1, \ldots, S_n, \Gamma) \\ &= \{(R_1, \ldots, R_n) \in S_1 \times \cdots \times S_n \mid \gamma(R_1, \ldots, R_n) = \texttt{wahr} \text{ für alle } \gamma \in \Gamma\}\end{aligned}$$

ein *Datenbankschema*.

Ein $(R_1, \ldots, R_n) \in \texttt{DS}$ heißt Beispiel für das Schema.

Beispiel

Seien die Attributmengen

$$\mathcal{A}_1 = \{\texttt{SchadensNr}, \texttt{Schadenstag}, \texttt{Schadenhöhe}, \texttt{VersScheinNr}\}$$
$$\mathcal{A}_2 = \{\texttt{VersScheinNr}, \texttt{KundenNr}, \texttt{Sparte}, \texttt{Deckung}, \texttt{Abschlussdaten}\}$$
$$\mathcal{A}_3 = \{\texttt{KundenNr}, \texttt{Name}, \texttt{Anschrift}, \texttt{Agent}\}$$

und die funktionalen Abhängigkeiten

$$\gamma_1 = \{\texttt{SchadensNr}\} \to \{\texttt{Schadenstag}, \texttt{Schadenhöhe}, \texttt{VersScheinNr}\}$$
$$\gamma_2 = \{\texttt{VersScheinNr}\} \to \{\texttt{KundenNr}, \texttt{Sparte}, \texttt{Deckung}, \texttt{Abschlussdaten}\}$$
$$\gamma_3 = \{\texttt{KundenNr}\} \to \{\texttt{Name}, \texttt{Anschrift}, \texttt{Agent}\}$$

gegeben. Daraus ergibt sich

$$S_i = S_i(\mathcal{A}_i, \{\gamma_i\}) \text{ sind Relationsschemata für } i = 1,2,3$$
$$\texttt{DS} = \texttt{DS}(S_1, S_2, S_3; \{\gamma_{\texttt{verträglich}}\}) \ni (R_1, R_2, R_3)$$

Bei einem solchen Schema können unabhängig Daten eingefügt werden. Es braucht etwa kein `Schaden` (beziehungsweise `Vertrag`) vorzuliegen, damit eine `VersScheinNr` (beziehungsweise `KundenNr`) vergeben werden kann.

Verträglichkeit

$\gamma_{\texttt{verträglich}}$ muß so aussehen, daß kein `Schaden` (beziehungsweise `Vertrag`) gespeichert werden kann, zu dem kein `Vertrag` (beziehungsweise `Kunde`) existiert.

Fremdschlüsselbedingung

Die Attributmenge $\mathcal{A}_i$ enthalte einen Schlüssel X von $S(\mathcal{A}_j, \Gamma_j)$. Dann lautet die Fremdschlüsselbedingung

$$t \in R_i \in S(\mathcal{A}_i, \Gamma_i) \Longrightarrow \exists t' \in R_j \in S(\mathcal{A}_j, \Gamma_j) : t|_X = t'|_X.$$

Das heißt, falls in R_i ein Tupel t eingefügt werden soll, muß in R_j ein Tupel existieren, das in X mit t übereinstimmt.

Im obigen Beispiel ist `VersScheinNr` ein Fremdschlüssel in $\mathcal{A}_1$ von $\mathcal{A}_2$.

Sei $\mathcal{A} = \bigcup_{i=1}^{n} \mathcal{A}_i$, Γ die Menge der funktionalen Abhängigkeiten $X \to Y$, für die ein $\mathcal{A}_i$ existiert mit $X, Y \in \mathcal{A}_i$ und $X \to Y \in \Gamma_i$. Betrachte nun die Abbildung

$$\pi : S(\mathcal{A}, \Gamma) \to S_1 \times \ldots \times S_n$$
$$S(\mathcal{A}, \Gamma) \ni R \mapsto (\pi_{\mathcal{A}_1}(R)), \ldots, \pi_{\mathcal{A}_n}(R) = (R_1, \ldots, R_n).$$

Man sagt $(R_1, \ldots, R_n) \in S_1 \times \cdots \times S_n$ erfüllt die *Universalrelationbedingung* $\gamma_{\texttt{UR}}$, wenn ein $R \in S(\mathcal{A}, \Gamma)$ existiert mit $\pi(R) = (R_1, \ldots, R_n)$.

Und $(R_1, \ldots, R_n) \in S_1 \times \cdots \times S_n$ erfüllt die *schwache Universalrelationbedingung* $\gamma_{\texttt{SUR}}$, wenn ein $R \in S(\mathcal{A}, \Gamma)$ exstiert mit $\pi(R) \supseteq (R_1, \ldots, R_n)$.

In der Theorie wird immer $\gamma_{\texttt{UR}}$ vorausgesetzt. Dann ist nämlich

$$\texttt{DS}(S_1, \ldots, S_n; \{\gamma_{\texttt{UR}}\})$$

ein Datenbankschema.

Wir beschäftigen uns im folgenden mit der Frage welche Information in $(R_1, \ldots, R_n)$ enthalten ist. Wir wissen bereits, daß $\overset{n}{\underset{i=1}{\bowtie}} R_i$ auf $\mathcal{A}$ definiert ist. Gilt aber auch $R = \overset{n}{\underset{i=1}{\bowtie}} R_i$ im allgemeinen?

Die Inklusion "$\subseteq$" gilt nach Übungsaufgabe 5 immer.

Beispiel

Seien die folgenden drei Relationen R, R_1 und R_2 gegeben, mit $R_1 = \pi_{\texttt{Nummer,LName,Ort}}(R)$ und $R_2 = \pi_{\texttt{TNummer,TName,Ort}}(R)$. Sei

R

Nummer	LName	Ort	TNummer	TName
L1	Meier	Köln	T1	Mutter
L2	Meier	Berlin	T2	Bolzen
L1	Meier	Köln	T3	Schraube
L3	Schmidt	Köln	T1	Mutter

R_1

Nummer	LName	Ort
L1	Meier	Köln
L2	Meier	Berlin
L3	Schmidt	Köln

R_2

TNummer	TName	Ort
T1	Mutter	Köln
T2	Bolzen	Berlin
T3	Schraube	Köln

Es folgt

$R_1 \bowtie R_2$

Nummer	LName	Ort	TNummer	TName
⋮	⋮	⋮	⋮	⋮
L3	Schmidt	Köln	T3	Schraube

Der letzte Datensatz ist falsch, die Inklusion "$\supseteq$" gilt also im allgemeinen nicht!

Definition

Gilt nun stets

$$R = \bowtie_{i=1}^{n} R_i$$

für

$$(R_1, \ldots, R_n) \in \texttt{DS}(S_1, \ldots, S_n; \{\gamma_{\texttt{UR}}\}) \text{ und } R \in S(\mathcal{A}, \Gamma),$$

so besitzt DS einen *verlustlosen Verbund* bezüglich $(\mathcal{A}_1, \ldots, \mathcal{A}_n)$.

Bemerkung

Die Aussage, daß

$$\pi\colon S(\mathcal{A}, \Gamma) \to \texttt{DS}(S_1, \ldots, S_n; \{\gamma_{\texttt{UR}}\})$$

injektiv ist, ist gleichbedeutend damit, daß die Zerlegung von $S(\mathcal{A}, \Gamma)$ in DS einen verlustlosen Verbund besitzt.

Beweis

Im obigen Beispiel wäre dasselbe R_1 und R_2 resultiert, wenn in R das Tupel

```
L3 Schmidt Köln T3 Schraube
```

vorgekommen wäre. Das heißt wir erhalten das gleiche Bild für verschiedene Urbilder, somit ist die Zerlegung nicht injektiv.

Algorithmus für den Test auf verlustlosen Verbund bei funktionalen Abhängigkeiten

Gegeben sei $S(\mathcal{A}, \Gamma)$ mit einer Menge von funktionalen Abhängigkeiten Γ und $(\mathcal{A}_1, \ldots, \mathcal{A}_m)$ mit

$$\mathcal{A} = \bigcup_{i=1}^{m} \mathcal{A}_i.$$

Ausgegeben werde, ob die Zerlegung von $S(\mathcal{A}, \Gamma)$ bezüglich $(\mathcal{A}_1, \ldots, \mathcal{A}_m)$ verlustlos ist.

Erstelle dazu eine Tabelle T mit den Zeilen $\mathcal{A}_1, \ldots, \mathcal{A}_m$ und den Spalten $A_1, \ldots, A_n$, wobei $\mathcal{A} = \{ A_1 , \ldots , A_n \}$, und mit Einträgen

$$T(\mathcal{A}_i, A_j) = \begin{cases} a & \text{falls } A_j \in \mathcal{A}_i \\ b_i & \text{sonst.} \end{cases}$$

Wiederhole bis sich T nicht mehr ändert

Durchlaufe Γ mit $X \to Y$

Falls zwei Zeilen $\mathcal{A}_k$ und $\mathcal{A}_l$ existieren, deren Einträge in den Spalten von X übereinstimmen, so setze deren Einträge auf Y ebenfalls gleich. Dabei wird als Ergebnis a eingetragen, wenn eine der Zeilen in der entsprechenden Spalte einen Eintrag a enthält, sonst das b mit dem kleineren der Indizes.

Genau dann, wenn die stabile Tafel eine Zeile enthält, die nur Einträge a besitzt, existiert ein verlustloser Verbund.

Beispiel

Sei $\mathcal{A} = \{A, B, C, D, E\}$ und $\Gamma = \{A \to C, B \to C, C \to D, DE \to C, CE \to A\}$ und $\mathcal{A}_1 = \{A, D\}$, $\mathcal{A}_2 = \{A, B\}$, $\mathcal{A}_3 = \{B, E\}$, $\mathcal{A}_4 = \{C, D, E\}$, $\mathcal{A}_5 = \{A, E\}$.

	A	B	C	D	E
$\mathcal{A}_1$	a	b_1	b_1	a	b_1
$\mathcal{A}_2$	a	a	b_2	b_2	b_2
$\mathcal{A}_3$	b_3	a	b_3	b_3	a
$\mathcal{A}_4$	b_4	b_4	a	a	a
$\mathcal{A}_5$	a	b_5	b_5	b_5	a

$A \to C$

	A	B	C	D	E
$\mathcal{A}_1$	a	b_1	b_1	a	b_1
$\mathcal{A}_2$	a	a	b_1	b_2	b_2
$\mathcal{A}_3$	b_3	a	b_3	b_3	a
$\mathcal{A}_4$	b_4	b_4	a	a	a
$\mathcal{A}_5$	a	b_5	b_1	b_5	a

$B \to C$

	A	B	C	D	E
$\mathcal{A}_1$	a	b_1	b_1	a	b_1
$\mathcal{A}_2$	a	a	b_1	b_2	b_2
$\mathcal{A}_3$	b_3	a	b_1	b_3	a
$\mathcal{A}_4$	b_4	b_4	a	a	a
$\mathcal{A}_5$	a	b_5	b_1	b_5	a

$C \to D$

	A	B	C	D	E
$\mathcal{A}_1$	a	b_1	b_1	a	b_1
$\mathcal{A}_2$	a	a	b_1	a	b_2
$\mathcal{A}_3$	b_3	a	b_1	a	a
$\mathcal{A}_4$	b_4	b_4	a	a	a
$\mathcal{A}_5$	a	b_5	b_1	a	a

$DE \to C$

	A	B	C	D	E
$\mathcal{A}_1$	a	b_1	b_1	a	b_1
$\mathcal{A}_2$	a	a	b_1	a	b_2
$\mathcal{A}_3$	a	a	a	a	a
$\mathcal{A}_4$	a	b_4	a	a	a
$\mathcal{A}_5$	a	b_5	a	a	a

$CE \to A$

	A	B	C	D	E
$\mathcal{A}_1$	a	b_1	b_1	a	b_1
$\mathcal{A}_2$	a	a	b_1	a	b_2
$\mathcal{A}_3$	a	a	a	a	a
$\mathcal{A}_4$	a	b_4	a	a	a
$\mathcal{A}_5$	a	b_5	a	a	a

Also handelt es sich um einen verlustlosen Verbund.

Beweis

Es ist $R \in S(\mathcal{A}, \Gamma)$, das heißt R erfüllt alle $X \to Y \in \Gamma$. Wir haben zu zeigen, daß "$\supseteq$" gilt.

Sei dazu also $(a_1, \ldots, a_n) \in \bowtie_{i=1}^{m} \pi_{\mathcal{A}_i}(R)$. Dann ist $(a_1, \ldots, a_n) \in R$ zu zeigen. Das Tupel $a = (a_1, \ldots, a_n)$ wurde aus den $\pi_{\mathcal{A}_i}(R)$ durch Verbund erzeugt, daher existiert also in jedem $\pi_{\mathcal{A}_i}(R)$ ein t_i mit $\pi_{\mathcal{A}_i}(a) = t_i$. In R existiert somit zu jedem t_i ein Tupel f_i mit $f_i|_{\mathcal{A}_i} = t_i$, das heißt

$$f_i(A_j) = \begin{cases} a_j & \text{für } A_j \in \mathcal{A}_i, \\ b_{ij} & \text{sonst,} \end{cases}$$

dabei ist b_{ij} ein bisher unbekannter Wert.

Wir haben somit $f_1, \ldots, f_n \in R$ und R erfüllt alle $X \to Y \in \Gamma$. Stimmen ein f_k und ein f_l auf X überein, so müssen sie auch auf Y übereinstimmen. Damit können die Einträge gleichgesetzt

werden, ist ein Eintrag dabei bekannt, das heißt ein a_j, so muß der unbekannte Wert im anderen Tupel ebenfalls a_j gewesen sein.

Besteht schließlich ein f_i nur noch aus a_j's, so ist also $f_i = (a_1, \ldots, a_n) \in R$.

Enthalte umgekehrt die stabile Tafel keine Zeile, die nur aus a_j's besteht. Dann besetze die Tafel mit paarweise verschiedenen Werten. Es ergibt sich eine Relation, die alle $X \to Y \in \Gamma$ erfüllt. Für diese Relation R ist $(a_1, \ldots, a_n) \notin R$ aber $(a_1, \ldots, a_n) \in \overset{m}{\underset{i=1}{\bowtie}} \pi_{\mathcal{A}_i}(R)$. In diesem Fall ist der Verbund also nicht verlustlos.

(Beim formalen Durchführen der Gleichsetzungen kann der Spaltenindex entfallen!)

Folgerung

Sei $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2$ und Γ eine Menge funktionaler Abhängigkeiten auf $\mathcal{A}$. Ist

$$\mathcal{A}_1 \cap \mathcal{A}_2 \to \mathcal{A}_1 \setminus \mathcal{A}_2 \in \Gamma^+,$$

so gibt es einen verlustlosen Verbund der Zerlegung von $S(\mathcal{A}, \Gamma)$ bezüglich $(\mathcal{A}_1, \mathcal{A}_2)$.

Beweis

Die Startbelegung der Tafel sieht folgendermaßen aus

	$\mathcal{A}_1 \cap \mathcal{A}_2$	$\mathcal{A}_1 \setminus \mathcal{A}_2$	$\mathcal{A}_2 \setminus \mathcal{A}_1$
$\mathcal{A}_1$	$a \ldots a$	$a \ldots a$	$b_1 \ldots b_1$
$\mathcal{A}_2$	$a \ldots a$	$b_2 \ldots b_2$	$a \ldots a$

Betrachten wir die Abhängigkeit $\mathcal{A}_1 \cap \mathcal{A}_2 \to \mathcal{A}_1 \setminus \mathcal{A}_2$, so wird daraus

	$\mathcal{A}_1 \cap \mathcal{A}_2$	$\mathcal{A}_1 \setminus \mathcal{A}_2$	$\mathcal{A}_2 \setminus \mathcal{A}_1$
$\mathcal{A}_1$	$a \ldots a$	$a \ldots a$	$b_1 \ldots b_1$
$\mathcal{A}_2$	$a \ldots a$	$a \ldots a$	$a \ldots a$

Beispiel

Sei $\mathcal{A} = \{\texttt{Lehrer}, \texttt{Kurs}, \texttt{Student}\}$ und $\Gamma = \{\texttt{Kurs} \to \texttt{Lehrer}\}$. Dann läßt sich nach obiger Folgerung jedes $R \in S(\mathcal{A}, \Gamma)$ verlustflos bezüglich $(\{\texttt{Kurs}, \texttt{Lehrer}\}, \{\texttt{Kurs}, \texttt{Student}\})$ zerlegen.

Die Armstrong-Axiome

Um formal bestimmen zu können, welche funktionalen Abhängigkeiten von einer Menge Γ funktionaler Abhängigkeiten logisch impliziert werden, kann man folgende drei Regeln verwenden.

Für alle $X, Y, Z \subseteq \mathcal{A}$ gilt

$$\text{Ist } Y \subseteq X \text{ so gilt } X \to Y. \tag{A1}$$

$$\text{Gilt } X \to Y \text{ so auch } XZ \to YZ. \tag{A2}$$

$$\text{Gilt } X \to Y \text{ und } Y \to Z, \text{ so auch } X \to Z. \tag{A3}$$

Diese Armstrong-Axiome sind korrekt und vollständig.

Beweis

Die Korrektheit läßt sich leicht zeigen, wir wollen uns hier mit der Vollständigkeit beschäftigen.

Es ist zu zeigen, daß sich jede funktionale Abhängigkeit $X \to Y \in \Gamma^+$ durch Anwenden der Regeln (A1), (A2) und (A3) aus Γ ableiten läßt. Sei dazu

$$X^+ := \{A \subseteq \mathcal{A} \mid X \to A \text{ ist durch (A1), (A2) und (A3) aus } \Gamma \text{ ableitbar}\}$$

- Sei $Y \subseteq X^+$. Dann ist die Abhängigkeit $X \to Y$ mit (A1), (A2) und (A3) ableitbar, denn für $Y = \{A_1, \ldots, A_n\}$ wissen wir, daß jedes $X \to A_i$ ableitbar ist. Vergrößere mit Induktion $\{A_1, \ldots, A_i\}$ für $i = 1, 2, \ldots$ bis Y erreicht ist. Dies funktioniert so: Aus $X \to A_i$ und $X \to A_1 \ldots A_{i-1}$ (IV) folgt mittels (A2) $XX \to XA_i$ und $XA_i \to A_1 \ldots A_{i-1}A_i$ und hieraus mit (A3) $X \to A_1 \ldots A_{i-1}A_i$.
- Sei nun $Y \not\subseteq X^+$. Dann kann $X \to Y$ keine logische Konsequenz der Abhängigkeiten in Γ sein, denn wir können eine Relation R definieren, die zwar alle Abhängigkeiten aus Γ erfüllt, aber $X \to Y$ nicht.

$$R = \begin{array}{|c|c|} \hline X^+ & \mathcal{A} \setminus X^+ \\ \hline 1 \ldots 1 & 0 \ldots 0 \\ 1 \ldots 1 & 1 \ldots 1 \\ \hline \end{array}$$

  Nach der Voraussetzung $Y \not\subseteq X^+$ erfüllt R $X \to Y$ nicht. Nun ist noch zu zeigen, daß R alle $V \to W \in \Gamma$ erfüllt. Sei dazu angenommen, daß ein $V \to W$ von R nicht erfüllt werde. Dann muß $V \subseteq X^+$ sein, da sonst $V \to W$ trivialerweise erfüllt wäre, da dann R keine auf V gleichen Tupel enthielte. Weiter muß $W \not\subseteq X^+$ gelten, da sonst $V \to W$ erfüllt wäre.

  Aus $V \subseteq X^+$ erhalten wir mittels (A1) $X^+ \to V$ und mit (A3) ergibt sich daraus $X \to V$. Weiter folgt hieraus mit (A3) und $V \to W$ die Abhängigkeit $X \to W$.

  Wegen $W \not\subseteq X^+$ existiert ein Attribut $A \in W \setminus X^+$. Damit gilt nach (A1) $W \to A$ und nach (A3) folgt damit $X \to A$, wobei $A \notin X^+$.

  Um dies zu zeigen haben wir nur (A1), (A2) und (A3) benutzt. Damit ist ein Widerspruch gefunden, da R diese Abhängigkeit nicht erfüllt.

Um zu entscheiden, ob $X \to Y \in \Gamma^+$ ist, braucht nur $Y \subseteq X^+$ geprüft zu werden.

Algorithmus zur Berechnung von X^+

Sei $X^{(0)} := X$. Sei

$$X^{(i+1)} := X^{(i)} \cup \{A \mid \text{Es exitiert ein } Y \to Z \in \Gamma \text{ mit } Y \subseteq X^{(i)} \text{ und } A \subseteq Z\}$$

Ist $X^{(i+1)} = X^{(i)}$ so ist $X^{(i)} = X^+$.

Beispiel

Sei $\mathcal{A} = \{A, B, E, G, H, I\}$, $\Gamma = \{AB \to E, E \to G, BE \to I, GI \to H\}$ und $X = AB$. Der Algorithmus liefert dann

$$\begin{aligned} X^{(0)} &= AB \\ &\Big\downarrow AB \to E \\ X^{(1)} &= ABE \\ &\Big\downarrow E \to G \\ X^{(2)} &= ABEG \\ &\Big\downarrow BE \to I \\ X^{(3)} &= ABEGI \\ &\Big\downarrow GI \to H \\ X^{(4)} &= ABEGIH \end{aligned}$$

Der Algorithmus bricht nach diesem Durchgang ab, da bereits $X^{(4)} = \mathcal{A}$ ist und somit nicht mehr vergrößert werden kann. Es ist also $X^+ = \mathcal{A}$. Weiter ist $A^+ = A$ und $B^+ = B$. Somit haben wir gezeigt, daß $X = AB$ Schlüssel von $S(\mathcal{A}, \Gamma)$ ist.

Ableitungsgraphen

Sei $\mathcal{A}$ eine Attributmenge und Γ eine Menge funktionaler Abhängigkeiten.

(1) Ist $X \subseteq \mathcal{A}$, so ist der Graph, der nur die Attribute aus X als Knoten und keine Pfeile enthält, ein Ableitungsgraph zu Γ.

(2) Sind $A_1,\ldots,A_k$ Knoten eines Ableitungsgraphen G_1 und ist $A_1 \ldots A_k \to \{C\} \cup Z \in \Gamma$, so ist auch G_2 mit

$$\text{Knoten}(G_2) = \text{Knoten}(G_1) \cup C$$
$$\text{Pfeile}(G_2) = \text{Pfeile}(G_1) \cup \{A_1 \to C, \ldots, A_k \to C\}$$

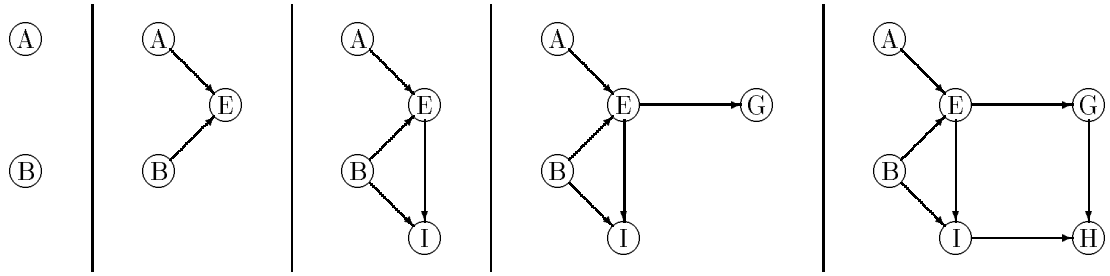
ein Ableitungsgraph zu Γ.

(3) Nur die mit (1) und (2) konstruierten Graphen sind Ableitungsgraphen zu Γ.

Ist X die Menge der Knoten eines Ableitungsgraphen zu Γ, auf die keine Pfeile zeigen, so heißt X Anfangsmenge. Ist $Y \subseteq \text{Knoten}(G)$, so heißt G Ableitungsgraph zu $X \to Y$ bezüglich Γ. Es gilt $X \to Y \in \Gamma^+$ genau dann, wenn es zu $X \to Y$ einen Ableitungsgraph bezüglich Γ gibt.

Beispiel

Seien $\mathcal{A}$ und Γ wie in obigem Beispiel, dann können wir folgenden Ableitungsgraph konstruieren. Wir verwenden dazu in dieser Reihenfolge die Regeln $AB \to E$, $BE \to I$, $E \to G$ und $GI \to H$.



Dieser Graph ist ein Ableitungsgraph zu $AB \to GH$.

Welche Regeln werden benutzt um einen Ableitungsgraph zu konstruieren

Die Axiome

$$X \to X \tag{B1}$$
$$\text{Ist } X \to YZ \in G \text{ und } Z \to CW \in \Gamma, \text{ dann gilt auch } X \to ZC \tag{B2}$$
$$\text{Gilt } X \to YZ, \text{ so auch } X \to Y \tag{B3}$$

werden verwendet, um einen Ableitungsgraphen zu konstruieren. Damit wird X^+ mitbestimmt.

Behauptung

Die Axiome (B1), (B2) und (B3) sind äquivalent zu (A1), (A2) und (A3).

Beweis

$B_i \Longrightarrow A_i$

- $\left.\begin{array}{l} X \to X \\ X = YZ (\Leftrightarrow X \supseteq Y) \end{array}\right\} \overset{(B3)}{\Longrightarrow} X \to Y$, das heißt (A1).
- $\left.\begin{array}{l} X \to Y \\ XZ \to XZ \end{array}\right\} \overset{(B2)}{\Longrightarrow} XZ \to XYZ \overset{(B3)}{\Longrightarrow} XZ \to YZ$, das heißt (A2).
- $\left.\begin{array}{l} X \to Y \\ Y \to Z \end{array}\right\} \overset{(B2)}{\Longrightarrow} X \to YZ \overset{(B3)}{\Longrightarrow} X \to Z$, das heißt (A3).

$A_i \Longrightarrow B_i$

Die Armstrongaxiome sind vollständig, implizieren also alle gültigen Regeln.

Damit ist also die Äquivalenz von

$$X \to Y \in \Gamma^+$$

zur Existenz eines Ableitungsgraphen für $X \to Y$ gezeigt.

Algorithmus zur Berechnung von X^+

Die einfachste Version dieses Algorithmus durchläuft zu jedem $A \in X$ alle $\gamma \in \Gamma$ und versucht X zu vergrößern, bis X nicht mehr wächst.

Die Effizienz kann verbessert werden durch

(1) Wurde $U \to V \in \Gamma$ einmal benutzt, so wird diese Abhängigkeit nie wieder verwendet.

(2) Wann läßt sich $U \to V$ verwenden, um X^i zu X^{i+1} zu vergrößern?

$U \subseteq X^i$: Zähle die Attribute von U, die noch nicht erfaßt sind. Sobald diese Zahl auf 0 heruntergezählt ist, diese Regel anwenden.

(3) Damit ohne Suchaufwand heruntergezählt werden kann, wird zu jedem Attribut eine Liste mit Verweisen auf die Regeln geführt, bei denen es auf der linken Seite vorkommt.

Algorithmus Linclosure zur Berechnung der Hülle X^+ von X bezüglich Γ

```
(1) Initialisierung
   for each FD W → Z ∈ Γ do
   begin
      count (W → Z) := |W|
      for each (A ∈ W)
         add (W → Z) to list(A)
   end
   newdep = X;
   update = X;
(2) Berechnung
   while update ≠ ∅ do
   begin
      wähle ein A ∈ update
      entferne A aus update
      for each W → Z ∈ list(A)
      begin
         count(W → Z) − −
         if count(W → Z) == 0 then
         begin
            add := Z \ newdep
            newdep = newdep ∪ add
            update = update ∪ add
         end
      end
   end
(3) Gebe newdep aus
```

Der Algorithmus leistet das gleiche wie der Tafelalgorithmus, ist also korrekt.

Aufwand

Sei n die Länge der Eingabe, das heißt

$$n = |X| + \sum_{W \to Z \in \Gamma} (|W| + |Z|)$$

und X und Γ wie immer. Dann ist der Aufwand von Linclosure $O(n)$.

Beweis

Bei der `while`-Schleife zählt man über alle Schleifendurchgänge hinweg global, welcher Aufwand insgesamt bei den Operationen anfällt. Da kein Suchen erforderlich ist, ist nur zu berücksichtigen, wie oft Herunterzählen und Vergrößern nötig ist. Schlimmstenfalls wird beim Herunterzählen jede linke Seite einer Abhängigkeit benötigt, das heißt es fällt der Aufwand

$$\sum_{W \to Z \in \Gamma} O(|W|)$$

an.

Bei add wird für jedes $A \in Z$ Aufwand $O(1)$ benötigt.

Die Berücksichtigung von Z erfordert den Aufwand $O(|Z|)$ für jede Abhängigkeit, die verwendet wird, insgesamt wird also über alle Abhängigkeiten hinweg der Aufwand

$$\sum_{W \to Z \in \Gamma} O(|Z|)$$

nötig.

Da das Initialisieren einen Aufwand der Größenordnung $O(|X|)$ erfordert, folgt nach Aufsummierung die Behauptung.

Beispiel

Sei $\Gamma = \{A \to D, AB \to E, BI \to E, CD \to I, E \to C\}$ und $X = \{A, E\}$.

Die Initialisierung liefert:

$$\text{newdep} = \{A, E\}, \quad \text{update} = \{A, E\}$$

$\text{list}(A) = (A \to D, AB \to E)$	$\text{count}(A \to D) = 1$
	$\text{count}(AB \to E) = 2$
$\text{list}(B) = (AB \to E, BI \to E)$	$\text{count}(BI \to E) = 2$
$\text{list}(C) = CD \to I$	$\text{count}(CD \to I) = 2$
$\text{list}(D) = CD \to I$	
$\text{list}(E) = E \to C$	$\text{count}(E \to C) = 1$
$\text{list}(I) = BI \to E$	

Der Algorithmus macht folgendes:

```
Wähle A ∈ update, update = {E}.
Durchlaufe list(A)
    {count(A → D) = 0, count(AB → E) = 1, add = {D}
    newdep = {A, E, D}, update = {E, D}
Wähle E ∈ update, update = {D}
Durchlaufe list(E)
    count(E → C) = 0
    add = {C}
usw.
```

Ziel des Folgenden wird es nun sein, eine sinnvolle Abbildung

$$S(\mathcal{A}, \Gamma) \to \texttt{DS}(S_1, \ldots, S_n, \Gamma)$$
$$R \to (R_1, \ldots, R_n)$$
$$R_i = \pi_{\mathcal{A}_i}(R)$$

zu finden. Die update-Operationen sollen auf den R_i vorgenommen werden. Dabei soll es ausreichen, statt Γ die lokalen Mengen Γ_i von Abhängigkeiten zu berücksichtigen.

Das Problem ist also zunächst die Γ_i zu bestimmen. Sei dazu $\gamma = X \to Y$ eine Abhängigkeit auf $\mathcal{A}$ und $\mathcal{B} \subseteq \mathcal{A}$, so definiere

$$\pi_{\mathcal{B}}(X \to Y) := \begin{cases} X \to Y \text{ auf } \mathcal{B} & \text{falls } X, Y \subseteq \mathcal{B} \\ \text{wahr} & \text{sonst,} \end{cases}$$

und

$$\Gamma_{\mathcal{B}} := \{\pi_{\mathcal{B}}(\gamma) \mid \gamma \in \Gamma^+\} = \pi_{\mathcal{B}}(\Gamma)$$

ist die Menge der auf $\mathcal{B}$ implizierten funktionalen Abhängigkeiten.

Beispiel

Sei $\mathcal{A} = \{A, B, C, D, E\}$ und $\Gamma = \{A \to E, B \to E, CE \to D\}$ und $\mathcal{B} = \{A, B, C, D\}$. Dann ist $\Gamma_{\mathcal{B}} = \{AC \to D, BC \to D\}$, wobei die trivialen Abhängigkeiten weggelassen wurden.

Sei die Tabelle

A	B	C	D
1	3	5	7
2	4	5	8
1	4	6	8

die die Abhängigkeiten in $\Gamma_{\mathcal{B}}$ trivialerweise erfüllt, gegeben. Erfüllt nun das Urbild über $\mathcal{A}$ die funktionalen Abhängigkeiten in Γ?

Wir müssen dazu das Relationenschema R kunstruieren. Sei also

$R =$

A	B	C	D	E
1	3	5	7	x
2	4	5	8	y
1	4	6	8	z
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Aus $A \to E$ folgt nun $x = z$, aus $B \to E$ folgt $y = z$ und daher aus $CE \to D$ $7 = 8$, also ein Widerspruch, das heißt die auf $\mathcal{B}$ implizierten Abhängigkeiten sind nicht notwendigerweise mit den funktionalen Abhängigkeiten auf $\mathcal{A}$ verträglich.

Sei Γ auf $\mathcal{A}$ gegeben und $\mathcal{A}_1, \ldots, \mathcal{A}_m$ mit

$$\bigcup_{i=1}^{m} \mathcal{A}_i = \mathcal{A}.$$

Weiter sei $\Gamma_i = \pi_{\mathcal{A}_i}(\Gamma)$. Gegeben sei die Abbildung

$$\pi : S(\mathcal{A}, \Gamma) \to \mathtt{DS}(S_1(\mathcal{A}_1, \Gamma_1), \ldots, S_n(\mathcal{A}_n, \Gamma_n); \Gamma_0).$$

Wegen $\gamma_{\mathtt{UR}}$ existiert dann für jedes $(R_1, \ldots, R_n) \in \mathtt{DS}$ ein $R \in S(\mathcal{A}, \Gamma)$ mit

$$R_i = \pi_{\mathcal{A}_i}(R) \quad \text{und} \quad \Gamma_0 = \{\gamma_{\mathtt{UR}}\}.$$

Falls dann R jedes $\gamma \in \Gamma$ erfüllt, so ist π *abhängigkeitserhaltend*.

Man braucht also bei einer abhängigkeitserhaltenden Zerlegung nur die Γ_i lokal zu prüfen, um sicherzustellen, daß das globale R alle $\gamma \in \Gamma$ erfüllt.

Seien Γ, Γ' Mengen von funktionalen Abhängigkeiten auf $\mathcal{A}$. Es gilt:

$$\Gamma \equiv \Gamma' \iff \begin{cases} \text{alle } \gamma' \in \Gamma' \text{ werden von } \Gamma \text{ logisch induziert} \\ \text{alle } \gamma \in \Gamma \text{ werden von } \Gamma' \text{ logisch induziert.} \end{cases}$$

Ein Verfahren zur Prüfung

Berechne X^+ bezüglich Γ. Sei $\gamma' = X \to Y \in \Gamma'$. Falls $Y \subseteq X^+$ bezüglich Γ, so wird γ' von Γ impliziert.

Eine solche Problemstellung ist natürlich rechnerisch entscheidbar.

Gegeben seien nun $\Gamma_1, \ldots, \Gamma_m$. Für alle i erfülle R_i die Abhängigkeiten aus Γ_i. Weiterhin existiere R über $\mathcal{A}$ mit der Eigenschaft $\pi_{\mathcal{A}_i}(R) = R_i$.

Welche Abhängigkeiten erfüllt nun R?

Sei $(X \to Y)_{\mathcal{A}_i} \in \Gamma_i$.

Erfüllt R_i die Abhängigkeit $(X \to Y)_{\mathcal{A}_i}$, so erfüllt R die Abhängigkeit $(X \to Y)_{\mathcal{A}}$.

Sei

$$\bigtimes_{i=1}^{n} \Gamma_i = \{(X \to Y)_{\mathcal{A}} \mid \text{Es existiert ein } i \text{ mit } X \cup Y \subset \mathcal{A}_i \text{ und } (X \to Y)_{\mathcal{A}_i} \in \Gamma_i\}.$$

Dann ist $\pi: S \to \texttt{DS}$ abhängigkeitserhaltend, wenn

$$\Gamma \equiv \bigtimes_{i=1}^{n} \Gamma_i$$

gilt. Mathematisch ausgedrückt heißt dies

$$\pi \text{ abhängigkeitserhaltend} \iff \pi \text{ surjektiv}$$
$$\pi \text{ hat verlustlosen Verbund} \iff \pi \text{ injektiv.}$$

Welche Eigenschaften soll ein Datenbanksystem(DS) haben?

Ziel ist es, Redundanz zu vermeiden, das heißt jede Information nur einmal zu speichern. Dies geschieht über die Definition von Normalformen.

Sei $S(\mathcal{A},\Gamma)$ ein Relationenschema in 1. Normalform, wobei Γ eine Menge von funktionalen Abhängigkeiten ist.

Definition

Eine Abhängigkeit $X \to Y \in \Gamma^+$ heißt *voll*, wenn es kein $Z \subsetneq X$ gibt mit $Z \to Y \in \Gamma^+$. Dann ist Y voll von X abhängig. A heißt Schlüsselattribut, wenn ein Schlüssel X von $S(\mathcal{A},\Gamma)$ mit $A \in X$ existiert.

Definition 2. Normalform

Jedes Nichtschlüsselattribut (NSA) ist von jedem Schlüssel voll abhängig.

Definition

Eine Abhängigkeit der Form $X \to Y \in \Gamma^+$, $Y \to \{A\} \in \Gamma^+$, $Y \not\to X$, $A \notin X$, $A \notin Y$ heißt *transitiv*. A heißt dann transitiv über Y von X abhängig.

Definition 3. Normalform

Jedes NSA ist von jedem Schlüssel nicht transitiv abhängig.

Definition Boyce-Codd-Normalform(BCNF)

Jedes Attribut ist von jedem Schlüssel nicht transitiv abhängig.

Bemerkung

Es gelten die folgenden Implikationen:

$$R \text{ erfüllt BCNF} \Longrightarrow R \text{ erfüllt 3NF}$$
$$R \text{ erfüllt 3NF} \Longrightarrow R \text{ erfüllt 2NF}$$
$$R \text{ erfüllt 2NF} \Longrightarrow R \text{ erfüllt 1NF}$$

Beispiel

Sei

$$\mathcal{A} = \{\texttt{Lieferant}, \texttt{Teil}, \texttt{Anzahl}, \texttt{Ort}, \texttt{Entfernung}\}$$

und

$$\Gamma = \{\{\texttt{Lieferant}, \texttt{Teil}\} \to \{\texttt{Anzahl}\}, \{\texttt{Lieferant}\} \to \{\texttt{Ort}\}, \{\texttt{Ort}\} \to \{\texttt{Entfernung}\}\}.$$

Dann hat $S(\mathcal{A}, \Gamma)$ als einzigen Schlüssel $X = \{\texttt{Lieferant}, \texttt{Teil}\}$.

Die funktionale Abhängigkeit $\{\texttt{Lieferant}\} \to \{\texttt{Ort}\}$ liegt in Γ^+, `Ort` ist ein NSA und es ist $\{\texttt{Lieferant}\} \not\subseteq X$. Also liegt keine 2NF vor. Dies verursacht die Redundanz, daß der Wohnort eines Lieferanten jedesmal erneut gespeichert wird, wenn eine Lieferung des Lieferanten erfolgt.

Setze

$$\mathcal{A}_1 = \{\texttt{Lieferant}, \texttt{Teil}, \texttt{Anzahl}\}$$
$$\Gamma_1 = \{\{\texttt{Lieferant}, \texttt{Teil}\} \to \{\texttt{Anzahl}\}\}$$
$$\mathcal{A}_2 = \{\texttt{Lieferant}, \texttt{Ort}, \texttt{Entfernung}\}$$
$$\Gamma_2 = \{\{\texttt{Lieferant}\} \to \{\texttt{Ort}\}, \{\texttt{Ort}\} \to \{\texttt{Enfernung}\}\}$$

Die Abbildung

$$S(\mathcal{A}, \Gamma) \xrightarrow{\pi} \texttt{DS}(S_1(\mathcal{A}_1, \Gamma_1), S_2(\mathcal{A}_2, \Gamma_2); \Gamma_0)$$

ist abhängigkeitserhaltend. $\mathcal{A}_1 \cap \mathcal{A}_2 = \{\texttt{Lieferant}\}$ und $\mathcal{A}_2 \setminus \mathcal{A}_1 = \{\texttt{Ort}, \texttt{Entfernung}\}$ und es gilt

$$\mathcal{A}_1 \cap \mathcal{A}_2 \to \mathcal{A}_2 \setminus \mathcal{A}_1 \in \Gamma^+,$$

also ist der Verbund verlustlos. Weiterhin sind S_1 und S_2 in 2NF.

S_1 ist darüberhinaus in 3NF, aber S_2 nicht, da $\{\texttt{Lieferant}\} \to \{\texttt{Ort}\}$, $\{\texttt{Ort}\} \to \{\texttt{Enfernung}\}$ eine transitive Abhängigkeit ist.

Sind mehrere Lieferanten am gleichen Ort, so wird die Entfernung dieses Ortes redundant gespeichert. Zerlege daher S_2 weiter in

$$\mathcal{A}_{21} = \{\texttt{Lieferant}, \texttt{Ort}\}$$
$$\Gamma_{21} = \{\{\texttt{Lieferant}\} \to \{\texttt{Ort}\}\}$$
$$\mathcal{A}_{22} = \{\texttt{Ort}, \texttt{Entfernung}\}$$
$$\Gamma_{22} = \{\{\texttt{Ort}\} \to \{\texttt{Enfernung}\}\}$$

$\texttt{DS}(S_{21}(\mathcal{A}_{21}, \Gamma_{21}), S_{22}(\mathcal{A}_{22}, \Gamma_{22}); \Gamma_0)$ ersetzt $S_2(\mathcal{A}_2, \Gamma_2)$ abhängigkeitserhaltend und der Verbund ist verlustlos.

Die Abbildung

$$S(\mathcal{A}, \Gamma) \to \texttt{DS}(S_1(\mathcal{A}_1, \Gamma_1), S_{21}(\mathcal{A}_{21}, \Gamma_{21}), S_{22}(\mathcal{A}_{22}, \Gamma_{22}); \Gamma_0)$$

ist also ebenfalls abhängigkeitserhaltend, der Verbund ist verlustlos und die Schemata sind in 3NF und in BCNF.

Ein Relationenschema in 3NF muß nicht notwendigerweise in BCNF sein, wie das folgende Beispiel lehrt.

Beispiel

Sei

$$\mathcal{A} = \{\texttt{Hersteller}, \texttt{Modell}, \texttt{Gerätenummer}\}$$

und

$$\Gamma = \{\{\texttt{Hersteller}, \texttt{Gerätenummer}\} \to \{\texttt{Modell}\}, \{\texttt{Modell}\} \to \{\texttt{Hersteller}\}\}.$$

Dann hat $S(\mathcal{A}, \Gamma)$ als Schlüssel $\{\texttt{Hersteller}, \texttt{Gerätenummer}\}$ und $\{\texttt{Modell}, \texttt{Gerätenummer}\}$. Jedes Attribut ist Schlüsselattribut, also liegt 3NF vor. Wegen der transitiven Abhängigkeit

$$\{\texttt{Modell}, \texttt{Gerätenummer}\} \to \{\texttt{Modell}\} \quad \{\texttt{Modell}\} \to \{\texttt{Hersteller}\}$$

liegt aber keine BCNF vor. Die Redundanz besteht darin, daß bei jedem Gerät eines Modelltyps aufgelistet wird, wer Hersteller des Modells ist.

Setze

$$\mathcal{A}_1 = \{\texttt{Modell}, \texttt{Gerätenummer}\}$$
$$\Gamma_1 = \emptyset$$
$$\mathcal{A}_2 = \{\texttt{Modell}, \texttt{Hersteller}\}$$
$$\Gamma_2 = \{\{\texttt{Modell}\} \to \{\texttt{Hersteller}\}\}$$

Die Zerlegung

$$S(\mathcal{A}, \Gamma) \to \texttt{DS}(S_1(\mathcal{A}_1, \Gamma_1), S_2(\mathcal{A}_2, \Gamma_2); \Gamma_0)$$

hat einen verlustlosen Verbund, ist aber nicht abhängigkeitserhaltend, weil die ursprüngliche funktionale Abhängigkeit $\{\texttt{Hersteller}, \texttt{Gerätenummer}\} \to \{\texttt{Modell}\}$ nicht mehr vorhanden ist. Eine abhängigkeitserhaltende Zerlegung ist hier nicht möglich.

Ein Beispiel für einen Widerspruch liefert die konkrete Tabelle

Hersteller	Gerätenummer	Modell
Opel	1	Kadett
Ford	1	Fiesta
Opel	2	Kadett

die zerlegt wird in

Gerätenummer	Modell
1	Kadett
1	Fiesta
2	Kadett

und

Modell	Hersteller
Kadett	Opel
Fiesta	Ford

In den zerlegten Tabellen kann man die Tupel $(1, \texttt{Taunus})$ und $(\texttt{Taunus}, \texttt{Ford})$ einfügen, wogegen in der oberen Tabelle kein entsprechendes Tupel $(\texttt{Ford}, 1, \texttt{Taunus})$ existieren darf, da $\{\texttt{Hersteller}, \texttt{Gerätenummer}\}$ sonst kein Schlüssel mehr wäre.

Algorithmus zur Zerlegung in BCNF-Schemata

Gegeben sei $S(\mathcal{A}, \Gamma)$ mit einer Menge Γ von funktionalen Abhängigkeiten. Ausgegeben wird eine verlustlose Zerlegung

$$\texttt{DS}(S_1(\mathcal{A}_1, \Gamma_1), \ldots, S_n(\mathcal{A}_n, \Gamma_n); \Gamma_0)$$

mit der Eigenschaft, daß alle $S_i(\mathcal{A}_i, \Gamma_i)$ in BCNF sind.

Methode

1. Initialisiere $\texttt{DS}(S_1, \Gamma_0)$, wobei $S_1 = S$.
2. Ist S_i ein Schema aus `DS` und nicht in BCNF, so sei $X \to \{A\} \in \Gamma_i^+$, X enthalte keinen Schlüssel von S_i und es sei $A \notin X$.

   Zerlege S_i in S_{i1} und S_{i2}

   $$\begin{aligned} \mathcal{A}_{i1} &= \{A\} \cup X \\ \Gamma_{i1} &= \pi_{\mathcal{A}_{i1}}(\Gamma_i) \\ \mathcal{A}_{i2} &= \mathcal{A}_i \setminus \{A\} \\ \Gamma_{i2} &= \pi_{\mathcal{A}_{i2}}(\Gamma_i) \end{aligned}$$

   Wiederhole dies solange, bis jedes S_i in BCNF ist.
3. Gebe die Zerlegung aus.

Test auf BCNF

Falls X kein Schlüssel von $\mathcal{A}_i$ ist, können wir $A, B \in \mathcal{A}_i$ mit $A \in (\mathcal{A}_i \setminus \{A, B\})^+$ bezüglich Γ finden. Damit folgt die transitive Abhängigkeit

$$\mathcal{A}_i \setminus \{A\} \to \underbrace{\mathcal{A}_i \setminus \{A, B\}}_{X} \to A.$$

Dieses Kriterium läßt sich verwenden, um eine Abhängigkeit $X \to A$ für die obige Zerlegung zu finden.

Algorithmus

1. Falls $\mathcal{A}_i$ kein $\{A, B\}$ mit $A \in (\mathcal{A}_i \setminus \{A, B\})^+$ besitzt, so hat $S_i(\mathcal{A}_i, \pi_{\mathcal{A}_i}(\Gamma))$ bereits BCNF.
2. Nimm ein solches Paar $\{A, B\}$ und setze $Y = \mathcal{A}_i \setminus \{B\}$.
3. Solange in Y ein Paar $\{A, B'\}$ mit $A \in (Y \setminus \{A, B'\})^+$ existiert:

   setze $Y = Y \setminus \{B'\}$.
4. Setze $\mathcal{A}_{i1} = Y$ und $\mathcal{A}_{i2} = \mathcal{A}_i \setminus \{A\}$.

Das Schema $S_{i1}(\mathcal{A}_{i1}, \pi_{\mathcal{A}_{i1}}(\Gamma_i))$ hat nun BCNF. Das Schema S_{i2} muß weiter untersucht und eventuell zerlegt werden.

Die Komplexität ist polynomial ($O(n^4)$) zur Länge n der Eingabe, da n-mal ein Attribut A entfernt werden kann und jedesmal bei den verbleibenden Attributen alle Paare gebildet werden müssen. Dabei ist dann `Linclosure()` aufzurufen.

Bemerkung

Nicht für jede Komponente wird $\pi_{\mathcal{A}_i}(\Gamma)$ bei jedem Zerlegungsschritt bestimmt.

Nur für die BCNF-Komponente wird ein Schlüssel ausgezeichnet: $\mathcal{A}_i = X \cup \{A\}$, falls $X \to A$ existiert mit einem Schlüssel X von $S_i(\mathcal{A}_i, \Gamma_i)$. Für die weitere Zerlegung wird stets Γ verwendet.

Die erreichte Zerlegung in BCNF-Schemata hat einen verlustlosen Verbund, ist aber nicht notwendig abhängigkeitserhaltend.

Beweis Verlustloser Verbund

Dann gilt mit $X \to \{A\}$ und $\mathcal{A}_1 = X \cup \{A\}$, daß $X \to \mathcal{A}_1 \in \Gamma^+$ ist. Weiter ist $\mathcal{A}_2 = \mathcal{A} \setminus \{A\}$ und $\mathcal{A}_1 \cap \mathcal{A}_2 = X$. Die Zerlegung bezüglich $\mathcal{A}_1$ und $\mathcal{A}_2$ ist also verlustlos und bei der Iteration wird diese Eigenschaft erhalten.

Beispiel (eigentlich für 3NF, da nur ein Schlüssel vorhanden ist!)

Sei

$$\mathcal{A} = \{\texttt{Vorlesung}, \texttt{Lehrer}, \texttt{Stunde}, \texttt{Raum}, \texttt{Student}\}$$

und

$$\begin{aligned}\Gamma = \{&\{\texttt{Vorlesung}\} \to \{\texttt{Lehrer}\}\\ &\{\texttt{Stunde}, \texttt{Raum}\} \to \{\texttt{Vorlesung}\},\\ &\{\texttt{Stunde}, \texttt{Lehrer}\} \to \{\texttt{Raum}\},\\ &\{\texttt{Stunde}, \texttt{Student}\} \to \{\texttt{Raum}\}\}\end{aligned}$$

Dabei ist $\{\texttt{Stunde}, \texttt{Student}\}$ Schlüssel. Da $\{\texttt{Vorlesung}\}$ kein Schlüssel ist, setze

$$\begin{aligned}\mathcal{A}_1 &= \{\texttt{Vorlesung}, \texttt{Lehrer}\}\\ \Gamma_1 &= \{\{\texttt{Vorlesung}\} \to \{\texttt{Lehrer}\}\}.\end{aligned}$$

Setzt man

$$\mathcal{A}_2 = \{\texttt{Vorlesung}, \texttt{Stunde}, \texttt{Raum}, \texttt{Student}\},$$

so ist $\{\texttt{Stunde}, \texttt{Student}\}$ der einzige Schlüssel.

Wegen

$$\{\texttt{Vorlesung}, \texttt{Stunde}\}^+ \supseteq \{\texttt{Vorlesung}, \texttt{Stunde}, \texttt{Lehrer}, \texttt{Raum}\}$$

verletzt $\{\texttt{Vorlesung}, \texttt{Stunde}\} \to \{\texttt{Raum}\}$ die BCNF-Bedingung, da das Attribut $\{\texttt{Raum}\}$ über $\{\texttt{Vorlesung}, \texttt{Stunde}\}$ transitiv von $\{\texttt{Stunde}, \texttt{Student}\}$ abhängig ist.

Spaltet man dieses Schema noch einmal auf in

$$\mathcal{A}_{21} = \{\texttt{Vorlesung}, \texttt{Stunde}, \texttt{Raum}\}$$

mit

$$\Gamma_{21} = \{\{\texttt{Vorlesung}, \texttt{Stunde}\} \to \{\texttt{Raum}\}, \{\texttt{Stunde}, \texttt{Raum}\} \to \{\texttt{Vorlesung}\}\}$$

und

$$\mathcal{A}_{22} = \{\texttt{Vorlesung}, \texttt{Stunde}, \texttt{Student}\}$$

mit

$$\Gamma_{22} = \{\{\texttt{Stunde}, \texttt{Student}\} \to \{\texttt{Vorlesung}\}\},$$

so sind die resultierenden Schemata in BCNF. Man beachte, daß die Projektionen der Γ_i von Γ^+ herkommen.

Es wurde also anschaulich eine Zerlegung in Vorlesungs-, Hörsaal- und Stundenplan vorgenommen.

Genauso hätte man bei $\mathcal{A}_2$ auch

$$\mathcal{A}_{21} = \{\texttt{Stunde}, \texttt{Raum}, \texttt{Student}\}$$

wählen können. Daraus wäre ein Raumplan für Studenten resultiert.

Daraus ersieht man, daß nicht jede Zerlegung, die automatisch erzeugt wird, sinnvoll ist. Manuelles Eingreifen ist hier nötig.

Die Abhängigkeit

$$\{\texttt{Stunde}, \texttt{Lehrer}\} \to \{\texttt{Raum}\}$$

bleibt bei dieser Zerlegung nicht erhalten. Es besteht also das Risiko, daß bei ausschließlich lokaler Beachtung der Abhängigkeiten global eine falsche Eintragung erzeugt wird, das heißt in unserem Beispiel eine "Doppelbelegung von Lehrern".

Statt BCNF läßt sich die dritte Normalform erreichen. Dabei ist es möglich, sowohl abhängigkeitserhaltend als auch verlustlos zu zerlegen.

Idee "Synthesealgorithmus"

Die Menge der Abhängigkeiten wird auf eine Basismenge reduziert. Die bei den einzelnen Abhängigkeiten verwendeten Attribute werden jeweils als Attributmenge eines Schemas aufgefaßt.

Definition

Sei Γ eine Menge von funktionalen Abhängigkeiten.

- Γ heißt *nichtredundant* genau dann, wenn $\Gamma \not\equiv \Gamma \setminus \{\gamma\}$ für alle $\gamma \in \Gamma$ ist.
- Γ heißt *rechtsreduziert* genau dann, wenn $\gamma = X \to \{A\}$ für alle $\gamma \in \Gamma$ ist.
- Γ heißt *linksreduziert* genau dann, wenn für alle Regeln $\gamma \in \Gamma$ mit $\gamma = X \to Y$ gilt, daß $\Gamma \not\equiv (\Gamma \setminus \{\gamma\}) \cup \{X \setminus \{A\} \to Y\}$ mit $A \in X$ ist.

Bemerkung

Die dritte Definition besagt, daß auf keiner linken Seite ein Attribut fortgelassen werden darf.

Jede Menge Γ aus funktionalen Abhängigkeiten ist äquivalent zu einer Menge Γ', die nichtredundant und reduziert ist.

Es kann vorkommen, daß $\Gamma_1 \equiv \Gamma_2$ ist und sowohl Γ_1 als auch Γ_2 nicht redundant und reduziert sind, aber daß $|\Gamma_1| \neq |\Gamma_2|$ ist.

Beispiel

Sei

$$\mathcal{A} = \{A, B, C, D, E, F\}$$

und

$$\Gamma = \{AB \to C, C \to A, BC \to D, ACD \to B, D \to EF, BE \to C, CF \to BD, CE \to AF\}.$$

Im ersten Schritt spaltet man die rechten Seiten auf und erhält als neue funktionale Abhängigkeiten

$$D \to E, D \to F, CF \to B, CF \to D, CE \to A, CE \to F.$$

Wegen $C \to A$ ist $CE \to A$ redundant. Weiterhin ist

$$CF \subseteq CFD \subseteq CFDA \subseteq ABCDF \subseteq (CF)^+$$

bezüglich Γ ohne $CF \to B$. Daher ist

$$\Gamma \equiv \Gamma \setminus \{CF \to B\}.$$

Es sind keine weiteren Abhängigkeiten redundant. Die funktionale Abhängigkeit $ACD \to B$ kann wegen $C \to A$ reduziert werden auf $CD \to B$. Die Menge

$$\Gamma_1 = \{AB \to C, C \to A, BC \to D, CD \to B, D \to E, D \to F, BE \to C, CF \to D, CE \to F\}$$

ist nichtredundant und reduziert. Man kann aber stattdessen auch $CE \to A$, $CF \to D$ und $ACD \to B$ weglassen.

Begründung

Aus $CF \subseteq CFB \subseteq BCFD$ folgt bereits $CF \to D$.

Ebenso kann wegen $ACD \subseteq ACDEF \subseteq ACDEFB$ die Abhängigkeit $ACD \to B$ weggelassen werden.

Die Menge

$$\Gamma_2 = \{AB \to C, C \to A, BC \to D, D \to E, D \to F, BC \to E, CF \to B, CE \to F\}$$

ist also ebenfalls nichtredundant und reduziert, aber es gilt

$$|\Gamma_1| = 9 \neq 8 = |\Gamma_2|.$$

Erster Synthesesalgorithmus

Gegeben sei $S(\mathcal{A}, \Gamma)$ mit einer nichtredundanten und reduzierten Menge von funktionalen Abhängigkeiten Γ und einem Schlüssel X.

Ausgegeben werde ein Datenbankschema

$$\mathtt{DS}(S_1(\mathcal{A}_1, \Gamma_1), \ldots, S_m(\mathcal{A}_m, \Gamma_m); \Gamma_0)$$

in 3NF und eine abhängigkeitserhaltende und verlustlose Zerlegung $\pi : S \to \mathtt{DS}$.

Methode

1. Durchlaufe Γ für $i = 1, \ldots, m-1$ mit $\gamma_i = U \to V$ und setze $\mathcal{A}_i = U \cup V$ und $\Gamma_i = \{\gamma_i\}$.
2. Setze $\mathcal{A}_m = X$ und $\Gamma_m = \emptyset$.
3. Gebe die $S_i(\mathcal{A}_i, \Gamma_i)$ aus.

Beweis

Nach Konstruktion ist die Zerlegung abhängigkeitserhaltend. Sie ist auch verlustlos, wie wir aus dem Test auf verlustlosen Verbund ersehen können.

	$\overbrace{A_1 \ldots A_r}^{\mathcal{A}_m}$	$A_{r+1} \ldots A_n$
$\mathcal{A}_1$	a	a
$\vdots$		
$\mathcal{A}_m$	$a \ldots a$	$b_m \ldots b_m$

Die Reihenfolge der Spalten entspricht der Reihenfolge, in der die Attribute bei der Berechnung von $\mathcal{A}_m^+$ hinzugenommen werden. Da $X = \mathcal{A}_m$ Schlüssel ist, folgt $\mathcal{A}_m^+ = \mathcal{A}$. Mit Induktion nach i zeigt man, daß das b_m in der Spalte A_{r+i} durch a ersetzt wird. Wird $\gamma_i = U \to V$ verwendet, so ist $U \subseteq \{A_1, \ldots, A_{r+i-1}\}$ und $V = \{A_{r+i}\}$ und folglich wird b_m in der $r+i$-ten Spalte durch a ersetzt. Mit Induktion ergibt sich eine Zeile bestehend nur aus a's.

Beispiel

Sei $\mathcal{A} = \{A, B, C\}$ und $\Gamma = \{AC \to B\}$. Dann liefert die Zerlegung der Tabelle R

A	B	C
a_1	b	c_1
a_2	b	c_2

die beiden Tabellen

R_1

A	B
a_1	b
a_2	b

und

R_2

B	C
b	c_1
b	c_2

$R_1 \bowtie R_2$ liefert schließlich

A	B	C
a_1	b	c_1
a_1	b	c_2
a_2	b	c_1
a_2	b	c_2

Dies ist nicht die ursprüngliche Tabelle. Wir berechnen nun $\pi_X(R)$

A	C
a_1	c_1
a_2	c_2

und bilden $\pi_X(R) \bowtie (R_1 \bowtie R_2)$ und erhalten wieder R. Der Schlüssel bewirkt also die Entfernung falscher Informationen.

Bemerkung

Ist ein $\gamma \in \Gamma$ mit $\gamma = X \to Y$ in $S(\mathcal{A}, \Gamma)$, wobei X Schlüssel ist, so kann das Schema $\mathcal{A}_m$ für einen gesonderten Schlüssel entfallen.

Es bleibt zu zeigen, daß jedes $S_i(\mathcal{A}_i, \Gamma_i)$ in 3NF ist. Ist $\Gamma_i = \emptyset$ bei $i = m$, so liegt 3NF vor. Betrachte nun $\Gamma_i = \{X \to \{A\}\}$, wobei $\mathcal{A}_i = X \cup \{A\}$. Sei angenommen, es gebe $U \to V \to \{B\}$ mit $V \not\to U$ und $B \notin V$ und $B \notin U$ in S_i, das heißt es liegt keine 3NF vor. Wir vergrößern nun U zu

$$Z := U \cup ((X \setminus V) \setminus \{B\}).$$

Damit gilt ebenfalls $Z \to V \to \{B\}$, da $Z \to U$ gilt. Weiter ist $B \notin Z$, $B \notin V$ und $V \not\to Z$.

Behauptung

Es ist $Z \to X \in \Gamma^+$. Man beachte, daß U Schlüssel ist, was direkt aus der Definition von 3NF folgt.

Beweis

Es gilt $Z \to \{B\}$ und $Z \to (X \setminus V) \setminus \{B\}$ und damit auch $Z \to X \setminus V$. Mit $Z \to V$ erhalten wir schließlich $Z \to X$.

Also enthält Z einen Schlüssel von $S_i, (\mathcal{A}_i, \Gamma_i)$.

Wegen $B \notin Z$ und $Z \supset U$ gilt auch $B \notin U$.

- Wäre $A \in Z$, so wäre $A \notin U$, da A kein Schlüsselattribut ist (Wäre A Schlüsselattribut, so wäre jedes Attribut von $\mathcal{A}_i$ Schlüsselattribut und es läge 3NF vor.)

  Also ist $U \subsetneq X$, da $A \neq B \in X$. Dies ist ein Widerspruch, da der Schlüssel X keine echte Teilmenge als Schlüssel enthalten darf.

- Also ist $A \notin Z$ und somit $Z \subseteq X$.

  1. Ist $B \neq A$, so gilt wegen $B \in X$, $B \notin U$ und $U \subset Z \subset X$ auch hier $U \subsetneq X$. Widerspruch!

2. Sei also $B = A$. Wegen $Z \to X$, $X \to Z$ und $Z \subset X$ ist $Z = X$. Es ist $W \subset X$, da $A = B \notin W$. Wir haben also

$$X = Z \to W \to \{B\} = \{A\}.$$

Wäre $W \subsetneq X$, so könnte man $X \to A$ durch $W \to A$ ersetzen und Γ wäre nicht reduziert.

Kritik

Das Ergebnis ist von der Auswahl der Schritte abhängig. Die Anzahl der Ergebnisschemata ist ebenfalls nicht eindeutig. Als Ausweg streben wir eine Verbesserung der Theorie an.

Praktisch

Gegenüber der BCNF-Zerlegung ist zwar die Zerlegung in 3NF abhängigkeitserhaltend, dies kann aber bedeuten, daß mehr Einzelschemata benötigt werden. In der Praxis ist abzuwägen, wie wichtig diese Eigenschaft ist.

Beispiel

Im Beispiel

$$\mathcal{A} = \{\texttt{Vorlesung}, \texttt{Lehrer}, \texttt{Stunde}, \texttt{Raum}, \texttt{Student}\}$$

war Γ bereits nichtredundant und reduziert. Wir erhalten

$$\begin{aligned}
\mathcal{A}_1 &= \{\texttt{Vorlesung}, \texttt{Lehrer}\} \\
\mathcal{A}_2 &= \{\texttt{Stunde}, \texttt{Raum}, \texttt{Vorlesung}\} \\
\mathcal{A}_3 &= \{\texttt{Stunde}, \texttt{Lehrer}, \texttt{Raum}\} \\
\mathcal{A}_4 &= \{\texttt{Stunde}, \texttt{Student}, \texttt{Raum}\}.
\end{aligned}$$

$\{\texttt{Student}, \texttt{Stunde}\}$ war Schlüssel von $\mathcal{A}$, somit enthält $\mathcal{A}_4$ einen Schlüssel.

Wir erhalten gegenüber der BCNF-Zerlegung ein zusätzliches Schema, dafür ist die Zerlegung abhängigkeitserhaltend.

Zur Struktur nichtredundanter Überdeckungen

Definition

Seien $X, Y, \subseteq \mathcal{A}$ und Γ eine Menge funktionaler Abhängigkeiten auf $\mathcal{A}$.

X und Y heißen *äquivalent bezüglich* Γ (in Zeichen $X \equiv Y$), wenn $X \to Y \in \Gamma^+$ und $Y \to X \in \Gamma^+$ gilt.

Lemma

Seien Γ_1 und Γ_2 nichtredundant und $\Gamma_1 \equiv \Gamma_2$ auf $\mathcal{A}$. Ist $X \to Y \in \Gamma_1$, so existiert ein $U \to V \in \Gamma_2$ mit $X \equiv U$ (bezüglich Γ_1 und Γ_2).

Beweis

Da $\Gamma_1 \equiv \Gamma_2$ ist, existiert ein Ableitungsgraph für $X \to Y$ bezüglich Γ_2, darin werden Abhängigkeiten aus Γ_2 verwendet. Jede dieser wiederum läßt sich aus Γ_1 ableiten, das heißt dazu existiert ein Ableitungsgraph bezüglich Γ_1.

Es ergibt sich durch das Einsetzen ein Ableitungsgraph für $X \to Y$ bezüglich Γ_1. Würde bei keiner Ableitung eines $U \to V$ aus Γ_2 durch Γ_1 die Abhängigkeit $X \to Y$ verwendet, so ließe sich in Γ_1 die Abhängigkeit $X \to Y$ entfernen. Dies ist ein Widerspruch zur Nichtredundanz von Γ_1.

Definition

Sei Γ eine Menge funktionaler Abhängigkeiten auf $\mathcal{A}$ und $X \subseteq \mathcal{A}$. Wir definieren

$$E_\Gamma(X) := \{U \to V \in \Gamma \mid X \equiv U\}$$

und

$$\overline{E}_\Gamma := \{E_\Gamma(X) \mid E_\Gamma(X) \neq \emptyset\}.$$

$\overline{E}_\Gamma$ ist eine Partition von Γ. Nach dem Lemma gilt für Γ_1 und Γ_2 mit $\Gamma_1 \equiv \Gamma_2$

$$E_{\Gamma_1}(X) \neq \emptyset \Longleftrightarrow E_{\Gamma_2}(X) \neq \emptyset$$

und somit

$$\left|\overline{E}_{\Gamma_1}\right| = \left|\overline{E}_{\Gamma_2}\right|.$$

Definition

Eine Menge von funktionalen Abhängigkeiten Γ heißt *minimal*, wenn für alle Γ' mit $\Gamma' \equiv \Gamma$

$$|\Gamma'| \geq |\Gamma|$$

gilt.

Bemerkung

Jedes minimale Γ ist nichtredundant.

Ziel

Algorithmus, der zu gegebenem Γ ein dazu äquivalentes minimales Γ' erzeugt.

Definition

Sei Γ eine Menge funktionaler Abhängigkeiten auf $\mathcal{A}$ und $X, Y \subseteq \mathcal{A}$.

X bestimmt das Y *direkt unter* Γ (in Zeichen $X \overset{\bullet}{\to} Y$), wenn es eine nichtredundante Menge $\Gamma' \equiv \Gamma$ gibt, bei der ein Ableitungsgraph für $X \to Y$ keine Abhängigkeit $U \to V$ mit $U \equiv X$ benutzt, das heißt $E_{\Gamma'}(X)$ wird nicht benötigt.

Lemma

Folgende Aussagen sind äquivalent

(i) $X \overset{\bullet}{\to} Y$

(ii) Für alle nichtredundanten Γ' mit $\Gamma' \equiv \Gamma$ existiert eine Ableitung für $X \to Y$ bezüglich Γ', die kein Element aus $E_{\Gamma'}(X)$ verwendet.

Beweis

"$\Leftarrow$" Siehe Definition

"$\Rightarrow$" Seien Γ' und Γ'' nichtredundant und es gelte $\Gamma' \equiv \Gamma \equiv \Gamma''$. Für $X \to Y$ existiere ein Ableitunggraph bezüglich Γ', der keine Abhängigkeit aus $E_{\Gamma'}(X)$ verwendet.

Behauptung

Auch zu Γ'' existiert ein solcher Ableitungsgraph.

Beweis

Jede der verwendeten Abhängigkeiten aus Γ' wird von Γ'' impliziert. Für jede Regel $W \to Z \in \Gamma'$, die verwendet wird, bilde den Ableitungsgraphen bezüglich Γ'' und ersetze den zu $W \to Z$ gehörenden Teilgraphen durch diesen Ableitungsgraphen bezüglich Γ''. Es ergibt sich ein Ableitungsgraph für $X \to Y$ bezüglich Γ''.

Behauptung

Der so konstruierte Ableitungsgraph verwendet keine Abhängigkeiten aus $E_{\Gamma''}(X)$

Beweis

Sei angenommen, daß $U \to V \in E_{\Gamma''}(X)$ verwendet würde, um $W \to Z \in \Gamma'$ zu ersetzen. Wenn wir $U \to W \in \Gamma^+$ zeigen können, so ist wegen $U \subseteq W^+$ $W \to U$, damit $W \equiv U$ und $W \equiv U \equiv W$ sowie schließlich $W \to Z \in E_{\Gamma'}(X)$, was einen Widerspruch darstellt.

Wir haben also noch $U \to W \in \Gamma^+$ zu zeigen. Es wurde $W \to Z$ benutzt um $X \to Y$ herzuleiten, also $W \subseteq X^+$ und damit $X \to W \in \Gamma^+$. Wegen $U \to V \in E_{\Gamma''}(X)$ ist $U \equiv X$, insbesondere $U \to X \in \Gamma^+$. Schließlich ist also $U \to W \in \Gamma^+$.

Satz

Seien Γ_1 und Γ_2 minimal und $\Gamma_1 \equiv \Gamma_2$. Dann gilt für alle $X \subseteq \mathcal{A}$

$$|E_{\Gamma_1}(X)| = |E_{\Gamma_2}(X)| \,.$$

Lemma A

Γ sei nichtredundant. Sei $X \to Y \in E_\Gamma(X)$ und $Y \subseteq \mathcal{A}$ mit $Y \equiv X$. Dann existiert eine Regel $Z \to W \in E_\Gamma(X)$ mit $Y \overset{\bullet}{\to} Z$.

Beweis

Seien X_1, X_2, ..., X_n alle zu X äquivalenten Attributmengen. Dann existieren i_0 und j_0 mit $Y = X_{i_0}$ und $X = X_{j_0}$. Im allgemeinen ist nicht jedes X_i linke Seite eines $\gamma \in \Gamma$. Gesucht ist ein $Z \in \{X_i \mid i = 1, \ldots, n\}$, das als linke Seite auftritt und für das $Y \overset{\bullet}{\to} Z$.

Falls Y selbst als linke Seite auftritt, so wähle $Z = Y$ (denn $Y \overset{\bullet}{\to} Y$ gilt immer).

Sei also Y keine linke Seite. Da $Y \equiv X_i$ für jedes i, kann man für jedes $Y \to X_i$ einen Ableitungsgraphen erstellen. Unter allen solchen Ableitungsgraphen wähle einen solchen aus, der minimale Knotenzahl hat und bei dem X_i linke Seite ist. (Ein solches X_i muß wegen $E_\Gamma(X) \neq \emptyset$ existieren.)

Behauptung

Dieser Ableitungsgraph verwendet kein $\gamma \in E_\Gamma(X)$.

Beweis

Sei angenommen, daß für $Y \to Z$ (Z linke Seite) ein Ableitungsgraph mit minimaler Knotenzahl ein $U \to V \in E_\Gamma(X)$ verwendet. Dann ist dieser Ableitungsgraph auch einer für $Y \to U$. Es gibt ein $A \in V$, das für die Ableitung von $Y \to U$ nicht gebraucht wird. Damit entsteht durch das Entfernen von A ein Ableitungsgraph für $Y \to U$, der weniger Knoten als der ursprüngliche besitzt, außerdem ist $U \to V \in E_\Gamma(X)$, im Widerspruch zur Minimalität des Graphen. Dieses $U \to V$ kann also nicht existieren, damit ist $Y \overset{\bullet}{\to} Z$.

Lemma B

Sei Γ minimal und $X \subseteq \mathcal{A}$. Dann existiert kein $Y \to U \in E_\Gamma(X)$ und kein $Z \to V \in E_\Gamma(X)$ mit $Y \neq Z$ und $Y \overset{\bullet}{\to} Z$.

Beweis

Sei angenommen, es gebe $Y \to U$ und $Z \to V \in E_\Gamma(X)$ mit $Y \overset{\bullet}{\to} Z$. Bilde einen Ableitungsgraph für $Y \to Z$, der keine Abhängigkeiten aus $E_\Gamma(X)$ verwendet. Dann können wir $E_\Gamma(X)$ abändern, ohne den Ableitungsgraphen für $Y \to Z$ zu verändern. Bilde Γ' aus Γ durch Ersetzen von $Y \to U$ und $Z \to V$ durch $Z \to UV$.

Behauptung

Es ist $\Gamma' \equiv \Gamma$

Beweis

- In $(\Gamma')^+$ ist $Z \to U$ und $Y \to Z$ und damit $Y \to U \in (\Gamma')^+$. Außerdem ist wegen $Z \to UV \in \Gamma'$ auch $Z \to V \in (\Gamma')^+$, also impliziert Γ' die Abhängigkeiten $Y \to U$ und $Z \to V$.
- Umgekehrt ist $Z \to V \in \Gamma^+$, $Y \to Z \in \Gamma^+$ und $Y \to U \in \Gamma^+$. Wegen $Z \equiv X$ gilt $Z \to Y$, denn $X \equiv Y$. Hieraus folgt schließlich $Z \to U \in \Gamma^+$ und damit impliziert Γ auch Γ'.

Aus $\Gamma' \equiv \Gamma$ folgt dann wegen $|\Gamma'| = |\Gamma| - 1$ ein Widerspruch zur Minimalität von Γ.

Beweis des Satzes

Seien also Γ_1 und Γ_2 minimal mit $\Gamma_1 \equiv \Gamma_2$. Weiter sei $X \subseteq \mathcal{A}$. Sei angenommen, daß

$$\underbrace{|E_{\Gamma_1}(X)|}_{=:m} < \underbrace{|E_{\Gamma_2}(X)|}_{=:n}.$$

Es sei

$$E_{\Gamma_1}(X) = \{X_1 \to \overline{X}_1, \ldots, X_m \to \overline{X}_m\}$$

und

$$E_{\Gamma_2}(X) = \{Y_1 \to \overline{Y}_1, \ldots, Y_n \to \overline{Y}_n\}.$$

Dann existiert ein Y_j mit $Y_j \neq X_i$ für alle i. (Sonst existieren k, l mit $Y_k = Y_l$, damit $Y_k \overset{\bullet}{\to} Y_l$, im Widerspruch zu Lemma B.)

Sei ohne Beschränkung der Allgemeinheit $j = 1$, das heißt $Y_1 \neq X_i$ für alle i. Nach Lemma A existiert ein X_i mit $Y_1 \overset{\bullet}{\to} X_i$. (Ohne Beschränkung der Allgemeinheit sei $i = 1$.) Bilde Γ_2' durch Ersetzen von $Y_1 \to \overline{Y}_1$ durch $X_1 \to \overline{Y}_1$.

Aus $Y_1 \overset{\bullet}{\to} X_1 \in (\Gamma_2')^+$ (Beweis wie oben) und $X_1 \to \overline{Y}_1$ folgt $Y_1 \to \overline{Y}_1 \in (\Gamma_2')^+$. Γ_2 impliziert $X_1 \to \overline{Y}_1$, da $X_1 \equiv Y_1$, denn $\Gamma_1 \equiv \Gamma_2$. Ist $X_1 = Y_j \neq Y_1$, so $Y_1 \to \overline{Y}_1$ und $Y_1 = Y_j \to \overline{Y}_j$, also $X_1 \overset{\bullet}{\to} Y_1$, im Widerspruch zu Lemma B. Also ist $X_1 \neq Y_j$ für alle $j \neq 1$.

Gegenüber Γ_2 ist die Anzahl der linken Seiten, die gleich irgendeinem X_i sind um eins gewachsen. Dies läßt sich iterieren: Jedes Y_j, das von allen X_i verschieden ist, kann man durch ein neues X_k ersetzen. Dies führt schließlich zum Widerspruch zu $n > m$.

Bemerkung

Die Abbildung

$$E_{\Gamma_1}(X) \to E_{\Gamma_2}(X);\ (X_i \to \overline{X}_i) \mapsto (Y_j \to \overline{Y}_j) \quad \text{mit } X_i \overset{\bullet}{\to} Y_j.$$

ist eine Bijektion, wenn Γ_1 und Γ_2 äquivalent und minimal sind.

Begründung

Zu $X_i \to \overline{X}_i$ existiert nach Lemma A ein Y_j mit $X_i \overset{\bullet}{\to} Y_j$ und ein k mit $Y_j \overset{\bullet}{\to} X_k$. Dann ist (siehe ÜA) $X_i \overset{\bullet}{\to} X_k$ für $i \neq k$, im Widerspruch zu Lemma B. Wir erhalten also

$$X_i \overset{\bullet}{\to} Y_j \Longrightarrow Y_j \overset{\bullet}{\to} X_i.$$

Ist $X_i \overset{\bullet}{\to} Y_j$ und $X_i \overset{\bullet}{\to} Y_k$, so $Y_j \overset{\bullet}{\to} X_i \overset{\bullet}{\to} Y_k$ und somit $Y_j \overset{\bullet}{\to} Y_k$, was wiederum einen Widerspruch zu Lemma B darstellt.

Somit ist die obige Abbildung wohldefiniert und bijektiv.

Man kann nach dem Beweis in $E_{\Gamma_1}(X)$ jedes X_i ersetzen durch Y_j mit $X_i \overset{\bullet}{\to} Y_j$, ohne Γ_1^+ zu verändern.

Beispiel

Es sei

$$\mathcal{A} = \{\texttt{Fahrgestellnummer}, \texttt{KFZ-Kennzeichen}, \texttt{Halter}, \texttt{Schaden}, \texttt{Tag}, \texttt{Zeit}, \texttt{SchadensNr}\}.$$

Weiter seien

$$\begin{aligned}\Gamma_1 = \{&\{\texttt{Fahrgestellnummer}\} \to \{\texttt{KFZ-Kennzeichen}, \texttt{Halter}\},\\ &\{\texttt{KFZ-Kennzeichen}\} \to \{\texttt{Fahrgestellnummer}\},\\ &\{\texttt{SchadensNr}\} \to \{\texttt{KFZ-Kennzeichen}, \texttt{Tag}, \texttt{Zeit}, \texttt{Schaden}\},\\ &\{\texttt{KFZ-Kennzeichen}, \texttt{Tag}, \texttt{Zeit}\} \to \{\texttt{SchadensNr}, \texttt{Schaden}\}\}\end{aligned}$$

und

$$\begin{aligned}\Gamma_2 = \{&\{\texttt{Fahrgestellnummer}\} \to \{\texttt{KFZ-Kennzeichen}\},\\ &\{\texttt{KFZ-Kennzeichen}\} \to \{\texttt{Fahrgestellnummer}, \texttt{Halter}\},\\ &\{\texttt{SchadensNr}\} \to \{\texttt{KFZ-Kennzeichen}, \texttt{Halter}, \texttt{Tag}, \texttt{Zeit}\},\\ &\{\texttt{Tag}, \texttt{Zeit}, \texttt{Fahrgestellnummer}\} \to \{\texttt{SchadensNr}, \texttt{Schaden}\}\}.\end{aligned}$$

Dann ist

$$\begin{aligned}E_{\Gamma_1}(\texttt{KFZ-Kennzeichen}) = \{&\{\texttt{Fahrgestellnummer}\} \to \{\texttt{KFZ-Kennzeichen}, \texttt{Halter}\},\\ &\{\texttt{KFZ-Kennzeichen}\} \to \{\texttt{Fahrgestellnummer}\}\}\\ E_{\Gamma_2}(\texttt{KFZ-Kennzeichen}) = \{&\{\texttt{Fahrgestellnummer}\} \to \{\texttt{KFZ-Kennzeichen}\},\\ &\{\texttt{KFZ-Kennzeichen}\} \to \{\texttt{Fahrgestellnummer}, \texttt{Halter}\}\}\end{aligned}$$

und

$$\begin{aligned}E_{\Gamma_1}(\texttt{SchadensNr}) = \{&\{\texttt{SchadensNr}\} \to \{\texttt{KFZ-Kennzeichen}, \texttt{Tag}, \texttt{Zeit}, \texttt{Schaden}\},\\ &\{\texttt{KFZ-Kennzeichen}, \texttt{Tag}, \texttt{Zeit}\} \to \{\texttt{SchadensNr}, \texttt{Schaden}\}\}\\ E_{\Gamma_2}(\texttt{SchadensNr}) = \{&\{\texttt{SchadensNr}\} \to \{\texttt{KFZ-Kennzeichen}, \texttt{Halter}, \texttt{Tag}, \texttt{Zeit}\},\\ &\{\texttt{Tag}, \texttt{Zeit}, \texttt{Fahrgestellnummer}\} \to \{\texttt{SchadensNr}, \texttt{Schaden}\}\}.\end{aligned}$$

Satz

Sei Γ nichtredundant und nicht minimal. Dann existiert $X \subseteq \mathcal{A}$ und $Y \to U \in E_\Gamma(X)$ sowie $Z \to V \in E_\Gamma(X)$ mit $(Y \to U) \neq (Z \to V)$, so daß $Y \overset{\bullet}{\to} Z$ bezüglich Γ.

Folgerung

Dann kann man in Γ die beiden Abhängigkeiten $Y \to U$ und $Z \to V$ durch $Z \to UV$ ersetzen.

Beweis des Satzes

Sei $\Gamma' \equiv \Gamma$ und Γ' sei minimal. Dann exitiert ein $X \subseteq \mathcal{A}$ mit

$$\underbrace{|E_{\Gamma'}(X)|}_{=:\,m} < \underbrace{|E_\Gamma(X)|}_{=:\,n}.$$

Sei wieder

$$E_{\Gamma'}(X) = \{X_1 \to \overline{X}_1, \ldots, X_m \to \overline{X}_m\}$$

und

$$E_\Gamma(X) = \{Y_1 \to \overline{Y}_1, \ldots, Y_n \to \overline{Y}_n\}.$$

Für alle Y_j existiert dann ein X_i mit $Y_j \overset{\bullet}{\to} X_i$ und wegen $m < n$ existieren i, j und k mit $j \neq k$ und $Y_j \overset{\bullet}{\to} X_i$ sowie $Y_k \overset{\bullet}{\to} X_i$. Weiter existiert ein h mit $X_i \overset{\bullet}{\to} Y_h$. Es ist entweder $h \neq j$ (Dann ist $Y_j \overset{\bullet}{\to} Y_h$ die gesuchte direkte Abhängigkeit) oder es ist $h \neq k$ und $Y_k \overset{\bullet}{\to} Y_h$ ist die gesuchte Abhängigkeit.

Algorithmus (zur Konstruktion einer minimalen Menge von funktionalen Abhängigkeiten, die zu Γ äquivalent ist)

1. Ersetze Γ durch äquivalentes nichtredundantes Γ'.
2. Bestimme alle Äquivalenzklassen von $\overline{E}_{\Gamma'}$.
3. Durchlaufe $\overline{E}_{\Gamma'}$ mit $E_{\Gamma'}(X)$
   Durchlaufe $E_{\Gamma'}(X)$ mit $Y \to U$
   Durchlaufe $E_{\Gamma'}(X)$ mit $Z \to V$
   Ist $(Y \to U) \neq (Z \to V)$ und $Y \overset{\bullet}{\to} Z$, so
   Ersetze $Y \to U$ und $Z \to V$ durch $Z \to UV$
4. Gebe das so veränderte Γ' aus.

zu 1.

```
Initialisiere Γ' := Γ
Für jedes X → Y ∈ Γ'
{
   Bilde (X')⁺ := Linclosure(X,Γ' \ (X → Y))
   Ist Y ⊆ (X')⁺, so Γ' := Γ' \ (X → Y)
}
```

Wir wollen nun den Zeitaufwand für diesen Schritt abschätzen. Die Länge der Eingabe sei n, damit hat `Linclosure()` den Aufwand $O(n)$. Diese Funktion wird für jedes $\gamma \in \Gamma$ aufgerufen, also haben wir hier einen Aufwand von $O(|\Gamma| \cdot n)$. Das Ersetzen der Abhängigkeiten hat einen Aufwand von $O(|\Gamma|)$. Wir erhalten also einen Gesamtaufwand für Schritt 1 von $O(|\Gamma| \cdot n)$.

zu 2.

Wir bilden eine Matrix M mit so vielen Zeilen, wie linke Seiten von Abhängigkeiten vorhanden sind, und so vielen Spalten, wie Abhängigkeiten vorhanden sind. Die Einträge $m(X, U \to V)$ sind folgendermaßen definiert

$$m(X, U \to V) := \begin{cases} 1, & \text{falls } U \subseteq X^+ \\ 0, & \text{sonst.} \end{cases}$$

Dann ist $X \equiv U$ äquivalent zu

$$\text{Es existieren } V, Z \text{ mit } m(X, U \to V) = 1 \text{ und } m(U, X \to Z) = 1.$$

In diesem Schritt muß zur Konstruktion der Matrix $|\Gamma|$-mal `Linclosure()` aufgerufen werden, der Aufwand beträgt daher $O(|\Gamma| \cdot n)$.

zu 3.

Wesentlicher Schritt ist hier der Test auf $Y \overset{\bullet}{\to} Z$

```
Betrachte jedes E_Γ(X) und jedes Y → U ∈ E_Γ(X)
   Bilde (Y'')⁺ := Linclosure(Y,Γ \ E_Γ(X))
```

Modifiziere dabei `Linclosure()` dahingehend, daß auch für die $\gamma \in E_\Gamma(X)$ die Funktion `count()` berechnet wird, also

$$\texttt{count}(\gamma) = 0 \Longrightarrow \text{Ist } \gamma = R \to S, \text{ so ist } R \subseteq (Y'')^+.$$

Ist dann $Z \subseteq (Y'')^+$ dann gilt $Y \overset{\bullet}{\to} Z$.

Es werden also nur Abhängigkeiten in $\Gamma \setminus E_\Gamma(X)$ benötigt, um $Y \to Z$ zu folgern, das heißt $Y \overset{\bullet}{\to} Z$.

Für jedes $Y \to U$ wird einmal der modifizierte `Linclosure()`-Algorithmus aufgerufen, damit ist der Aufwand $O(|\Gamma| \cdot n)$. Da der Aufwand für das Ersetzen wieder $O(|\Gamma|)$ ist, haben wir für den Schritt 3 einen Gesamtaufwand von $O(|\Gamma| \cdot n)$

Ist Γ minimal, so sind die Abhängigkeiten möglicherweise noch um einzelne Attribute rechts oder links zu verkürzen. Es ist jedesmal zu prüfen, ob die bisherige Menge von Abhängigkeiten äquivalent ist zu der, die durch das Ersetzen einer Abhänigkeit durch eine verkürzte entsteht. Für jedes A, das an irgendeiner Reduktion einer Abhängigkeit auftritt, ist jeweils `Linclosure()` zu rufen, damit beträgt dafür der Aufwand $O(n^2)$.

Ist Γ minimal und reduziert, so kann man manchmal trotzdem noch ein kürzeres $\Gamma' \equiv \Gamma$ finden.

Beispiel

Sei $\mathcal{A} = \{A, B, C, D, E\}$ und $\Gamma = \{A \to B, B \to A, AC \to DE, BD \to C\}$. Es ist $A \equiv B$ und wir erhalten

$$\mathcal{A}_1 = \{A, B\} \quad \text{und} \quad \Gamma_1 = \{A \to B, B \to A\}.$$

Weiter ist $AC \equiv BD$ und

$$\mathcal{A}_2 = \{A, B, C, D, E\} \quad \text{und} \quad \Gamma_2 = \{AC \to ABCDE, BD \to ABCDE\}.$$

Man kann hier B nicht aus $\mathcal{A}_2$ entfernen.

Wählt man jedoch das zu Γ äquivalente $\Gamma' = \{A \to B, B \to A, AC \to DE, AD \to C\}$, erhält man hier

$$\mathcal{A}'_2 = \{A, C, D, E\} \quad \text{und} \quad \Gamma'_2 = \{AC \to ACDE, AD \to ACDE\}.$$

Die Ursache hierfür ist, daß die Schlüssel von $S_2(\mathcal{A}_2, \Gamma_2)$ geschickt gewählt wurden.

Erinnerung

Ist $X \overset{\bullet}{\to} Z$, $X \equiv Z$, so läßt sich in $Z \to U$ das Z durch X ersetzen (Γ dabei minimal).

Algorithmus

Für jedes X_i ist $(X_i)^{\overset{\bullet}{+}}$ zu bilden, die größte Menge von Attributen, die von X_i direkt abhängig ist.

$$X_i \overset{\bullet}{\to} Z_i \iff Z_i \subseteq (X_i)^{\overset{\bullet}{+}}$$

Ist $Z_i \subseteq (X_i)^{\overset{\bullet}{+}}$, und $|Z_i|$ minimal, so ersetze X_i durch Z_i.

Definition

Seien $X_1, \ldots, X_k, Y \subseteq \mathcal{A}$ und R eine Relation auf $\mathcal{A}$. R erfüllt $(X_1, \ldots, X_k) \to Y$, wenn gilt

Für alle i, j erfüllt R die Abhängigkeit $X_i \to X_j$
Für alle i erfüllt R die Abhängigkeit $X_i \to Y$

$(X_1, \ldots, X_k) \to Y$ heißt *zusammengesetzte funktionale Abhängigkeit*.

Zu jeder Klasse $E_\Gamma(X) = \{X_1 \to Y_1, \ldots, X_k \to Y_k\}$ bilde

$$(X_1, \ldots, X_k) \to \bigcup_{i=1}^{k} Y_i,$$

dann bilden die zu einer Menge Γ von funktionalen Abhängigkeiten gebildeten zusammengesetzten funktionalen Abhängigkeiten eine *ringförmige Menge*.

Beispiel

Sei $\Gamma = \{A \to BC, B \to AD, AE \to I, BE \to IJ\}$, dann haben wir die ringförmige Menge

$$\{(A, B) \to ABCD, (AE, BE) \to IJ\}.$$

Für das zu Γ äquivalente $\Gamma' = \{A \to AB, AB \to B, B \to ACD, AE \to IJ\}$ erhalten wir

$$\{(A, AB, B) \to CD, (AE) \to IJ\}.$$

Hier ist die Abhängigkeit $AB \to B$ redundant, wir entfernen sie und erhalten

$$\{(A, B) \to CD, (AE) \to IJ\}.$$

Definition

Eine *ringförmige Menge* heißt *nichtredundant*, wenn keine Abhängigkeit mehr entfernt werden kann. Sie heißt reduziert, wenn kein Attribut auf der linken Seite auf eine rechte Seite verschoben werden kann und auf keiner rechten Seite mehr ein Attribut weggelassen werden kann.

Beispiel

Es sei

$$\begin{aligned}\Gamma = \{&B_1B_2 \to A, D_1D_2 \to B_1B_2, B_1 \to C_1, B_2 \to C_2,\\ &D_1 \to A, D_2 \to A, AB_1C_2 \to D_2, AB_2C_1 \to D_1\}\end{aligned}$$

Wegen $(B_1B_2)^+ = (D_1D_2)^+ = \mathcal{A}$ ist $B_1B_2 \equiv D_1D_2$. Wir erhalten die ringförmige Menge

$$\begin{aligned}&\{(B_1B_2, D_1D_2) \to A, (B_1) \to C_1, (B_2) \to C_2,\\ &(D_1) \to A, (D_2) \to A, (AB_1C_2) \to D_2, (AB_2C_1) \to D_1\}\end{aligned}$$

Wegen $(D_1) \to A$ kann man aus der Abhängigkeit $(B_1B_2, D_1D_2) \to A$ das Attribut A entfernen. Wir können also $(B_1B_2, D_1D_2) \to A$ durch $(B_1B_2, D_1D_2) \to \emptyset$ ersetzen.

Zweiter Synthesealgorithmus

1. Ersetze die Menge Γ von funktionalen Abhängigkeiten durch eine reduzierte minimale ringförmige Menge, die zu Γ äquivalent ist.
2. Für alle $(X_1, \ldots, X_k) \to Y$ konstruiere $S_i = S(\mathcal{A}_i, \Gamma_i)$ wobei

   $$\mathcal{A}_i = \left(\bigcup_{j=1}^{k} X_j\right) \cup Y$$

   und

   $$\Gamma_i = \{X_1 \to \mathcal{A}_i, \ldots, X_k \to \mathcal{A}_i\},$$

   das heißt die X_i sind ausgezeichnete Schlüssel von S_i.
3. Falls kein $\mathcal{A}_i$ einen Schlüssel für $S(\mathcal{A}, \Gamma)$ enthält, so ist ein weiteres Schema für einen Schlüssel hinzuzunehmen, das heißt $S_m = S(X, \emptyset)$, wobei X Schlüssel ist.

(3. wird benötigt, um einen verlustlosen Verbund zu garantieren.)

Der Zeitaufwand für diesen Algorithmus beträgt $O(n^2)$, wobei n die Länge der Eingabe ist.

Satz

Durch den zweiten Synthesealgorithmus entsteht eine Zerlegung in Schemata S_i in 3NF.

Beweis

Es sei S_i zu $(X_1, \ldots, X_k) \to Y$ gebildet. Sei angenommen, daß dieses S_i nicht in 3NF ist. Sei also $X \to Z \to A$, X Schlüssel von S_i, $Z \not\to X$, $A \notin Z$ und $A \notin X$.

Bilde einen Ableitungsgraphen für $Z \to A$. Dieser verwendet kein $U \to V \in E_\Gamma(X)$, denn sonst $X \to Z \to U \to X$, also $Z \to X$ im Widerspruch zu unserer Annahme.

Bilde Γ', indem A aus Y entfernt wird. $Z \to A$ ist bezüglich Γ' ableitbar, da zur Ableitung von $Z \to A$ kein $X_i \to Y$ benutzt wurde. Da

$$Z \subseteq \left(\bigcup_{i=1}^{k} X_i\right) \cup Y$$

ist $X \to Z$ und $X_i \to Z$ für alle i ableitbar, damit ist in Γ' die Abhängigkeit $X_i \to A$ ableitbar.

Wir haben also $\Gamma' \equiv \Gamma$ gezeigt, also kann Γ nicht reduziert gewesen sein.

Bemerkung

Falls $Z \not\subset \left(\bigcup_{i=1}^{k} X_i\right) \cup Y$ ist der Schluß nicht möglich.

Beispiel

Sei $\mathcal{A} = \{A, B, C\}$ und $\Gamma = \{A \to B, B \to C, C \to B\}$. Wir zerlegen in

$$\mathcal{A}_1 = \{A, B\} \quad \Gamma_1 = \{A \to B\}$$
$$\mathcal{A}_2 = \{B, C\} \quad \Gamma_2 = \{B \to C, C \to B\}$$

Es gilt $A \to C \to B$ und $B \notin \{C, A\}$ sowie $C \not\to A$, also ist bei $S(\mathcal{A}_1, \Gamma_1)$ eine verdeckte transitive Abhängigkeit gegeben, das heißt eine transitive Abhängigkeit über verschiedene Tabellen.

Da jedoch $B \equiv C$, ist in den Schemata keine Redundanz vorhanden.

Um die Frage nach der Allgemeingültigkeit dieser letzten Feststellung beantworten zu können, brauchen wir die sogenannten Mehrwertigen Abhängigkeiten.

Mehrwertige Abhängigkeiten

Beispiel

Gegeben sei die Tabelle

Vorlesung	Lehrer	Stunde	Raum	Student
Informatik	Kohl	Mo 8-10	H1	Vogel
Informatik	Kohl	Di 8-10	H2	Vogel
Informatik	Kohl	Mi 18-19	H7	Vogel
Mathematik	Gauss	Di 10-11	H3	Hinz
Mathematik	Gauss	Mi 10-11	H3	Hinz

Hinz möchte aber auch Informatik hören. Alle Tupel bei Vogel können übernommen werden, wobei

Vogel durch Hinz zu ersetzen ist. Wir erhalten die Tabelle

Vorlesung	Lehrer	Stunde	Raum	Student
Informatik	Kohl	Mo 8-10	H1	Vogel
Informatik	Kohl	Di 8-10	H2	Vogel
Informatik	Kohl	Mi 18-19	H7	Vogel
Mathematik	Gauss	Di 10-11	H3	Hinz
Mathematik	Gauss	Mi 10-11	H3	Hinz
Informatik	Kohl	Mo 8-10	H1	Hinz
Informatik	Kohl	Di 8-10	H2	Hinz
Informatik	Kohl	Mi 18-19	H7	Hinz

Definition

Sei R eine Relation auf $\mathcal{A}$ und $X, Y \subseteq \mathcal{A}$. Dann erfüllt R die *mehrwertige Abhängigkeit* $X \twoheadrightarrow Y$ genau dann, wenn für alle $t, s \in R$ mit $s|_X = t|_X$ ein $u \in R$ existiert mit $u|_X = s|_X$, $u|_Y = t|_Y$ und $u|_{\mathcal{A}-(X \cup Y)} = s|_{\mathcal{A}-(X \cup Y)}$.

Im Beispiel ist $X \twoheadrightarrow Y$ mit $X = \{\texttt{Vorlesung}\}$ und $Y = \{\texttt{Stunde}, \texttt{Raum}\}$ erfüllt.

Bemerkung

Erfüllt R die mehrwertige Abhängigkeit $X \twoheadrightarrow Y$, so folgt unmittelbar aus der Definition auch $X \twoheadrightarrow \mathcal{A} - (X \cup Y)$.

Bemerkung

Erfüllt R die funktionale Abhängigkeit $X \to Y$, so auch die mehrwertige Abhängigkeit $X \twoheadrightarrow Y$.

Ist $Z \subseteq Y$ und erfüllt R die mehrwertige Abhängigkeit $X \twoheadrightarrow Y$, so braucht R nicht $X \twoheadrightarrow Z$ zu erfüllen.

Satz

Sei $\mathcal{A} = X \cup Y \cup Z$ mit X, Y, Z paarweise disjunkt. Dann erfüllt R die mehrwertige Abhängigkeit $X \twoheadrightarrow Y$ genau dann, wenn $R = \pi_{X \cup Y}(R) \bowtie \pi_{X \cup Z}(R)$, das heißt, wenn ein verlustloser Verbund bezüglich $(X \cup Y)$ und $(X \cup Z)$ vorliegt.

Beweis

"$\Rightarrow$" Per definition erfüllt.

"$\Leftarrow$" $\pi_{X \cup Y}(R)$ und $\pi_{X \cup Z}(R)$ haben die Attributmenge X als Verbundattribute. Es sind also nur Tupel zu betrachten, die auf X übereinstimmen.

Seien dazu $t, s \in R$ mit $s|_X = t|_X$. Die Menge $R = \pi_{X \cup Y}(\{s\}) \bowtie \pi_{X \cup Z}(\{t\})$ enthält u mit $u|_X = s|_X = t|_X$ und $u|_Y = s|_Y$ sowie $u|_Z = t|_Z$.

Wird also eine solche mehrwertige Abhängigkeit erkannt, so kann zerlegt werden!

Erinnerung (Test auf verlustlosen Verbund bei funktionalen Abhängigkeiten)

R erfüllt $R = \pi_{X \cup Y}(R) \bowtie \pi_{X \cup Z}(R)$ dann, wenn $X \to Y$ oder $X \to Z$ gilt.

Falls neben funktionalen Abhängigkeiten mehrwertige Abhängigkeiten erfüllt sein müssen, so gilt diese Aussage nicht mehr.

Beispiel

In der Tabelle

X	Y	Z
$a \ldots a$	$b \ldots b$	$a \ldots a$
$a \ldots a$	$a \ldots a$	$b \ldots b$

brauchen die mehrwertigen Abhängigkeiten nicht erfüllt zu sein.

Auch wenn durch funktionale Abhängigkeiten keine Zerlegung mehr möglich ist, kann durch mehrwertige Abhängigkeiten eine weitere Verfeinerung der Zerlegung möglich sein.

Definition 4NF

Sei Γ eine Menge von funktionalen und mehrwertigen Abhängigkeiten auf $\mathcal{A}$. Ein Relationenschema $S = S(\mathcal{A}, \Gamma)$ ist in 4NF genau dann, wenn aus $X \twoheadrightarrow Y \in \Gamma^+$ mit $Y \not\subseteq X$, $X \cup Y \neq \mathcal{A}$ und $Y \neq \emptyset$ folgt, daß X einen Schlüssel für S enthält.

Ziel des Folgenden wird es sein, einen Zerlegungsalgorithmus zu einer Zerlegung in 4NF-Schemata zu finden.

Satz

Ist S in 4NF, so auch in BCNF.

Beweis

Sei S in 4NF, aber nicht in BCNF. Dann existiert $X \to Y \in \Gamma^+$, wobei X keinen Schlüssel von S enthält. Dann erfüllt jedes $R \in S$ mit $X \to Y$ auch $X \twoheadrightarrow Y$ nach früherer Bemerkung. Da 4NF vorliegt, muß X einen Schlüssel enthalten im Widerspruch zur Annahme.

Beispiel

Sei $\mathcal{A} = \{A_1, A_2\}$ und $\Gamma = \{\emptyset \twoheadrightarrow \{A_1\}\}$. R erfüllt dann $\emptyset \twoheadrightarrow \{A_1\}$ genau dann, wenn

$$R = \pi_{\{A_1\}}(R) \bowtie \pi_{\{A_2\}}(R) = \pi_{\{A_1\}}(R) \times \pi_{\{A_2\}}(R).$$

Es gibt aber Schemata, die in BCNF aber nicht in 4NF sind.

Beispiel

Gegeben sei die Tabelle

Angestellter	Kind	Gebiet
Hilbert	Hilda	Mathematik
Hilbert	Hilda	Physik
Turing	Peter	Informatik
Turing	Peter	Mathematik
Pythagoras	Peter	Mathematik
Hilbert	Jochen	Mathematik
Hilbert	Jochen	Physik

in dem keine nichttriviale Abhängigkeit erfüllt wird. Das hierzu gebildete Schema ist also in BCNF, aber es liegt keine 4NF vor, da {Angestellter} $\twoheadrightarrow$ {Kind} gilt, und {Angestellter} kein Schlüssel ist.

Wir ergänzen nun den Zerlegungsalgorithmus in BCNF- zu einem Zerlegungsalgorithmus in 4NF-Schemata.

Statt zu testen, ob $X \to Y$ existiert, wobei X keinen Schlüssel enthält, teste, ob $X \twoheadrightarrow Y$ existiert, wobei X keinen Schlüssel enthält.

Wenn ja, so kann das entsprechende Schema $S(\mathcal{A}_i, \Gamma_i)$ zerlegt werden in $S(\mathcal{A}_{i1}, \Gamma_{i1})$ und $S(\mathcal{A}_{i2}, \Gamma_{i2})$, wobei $\mathcal{A}_{i1} = X \cup Y$ und $\mathcal{A}_{i2} = \mathcal{A}_{i1} - Y$ gilt. Weiterhin ist $\Gamma_{ij} = \pi_{\mathcal{A}_{ij}}(\Gamma_i)$ für $j = 1, 2$.

Definition

Sei $S = S(\mathcal{A}, \Gamma)$ und $\mathcal{B} \subseteq \mathcal{A}$ mit einer Menge Γ von funktionalen und mehrwertigen Abhängigkeiten. Die Abhängigkeit $\gamma = X \twoheadrightarrow Y \in \Gamma$ vererbt auf $\mathcal{B}$ die Abhängigkeit

$$\pi_{\mathcal{B}}(\gamma) = \begin{cases} X \twoheadrightarrow Y \cap \mathcal{B}, & \text{falls } X \subseteq \mathcal{B} \\ \text{wahr} & \text{sonst.} \end{cases}$$

Weiter setzt man

$$\pi_{\mathcal{B}}(\Gamma) = \{\pi_{\mathcal{B}}(\gamma) \mid \gamma \in \Gamma^+\}.$$

Bemerkung

Erfüllt R die mehrwertige Abhängigkeit $\gamma = X \twoheadrightarrow Y$ und ist $X \subseteq \mathcal{B}$, so erfüllt $\pi_{\mathcal{B}}(R)$ auch $\pi_{\mathcal{B}}(\gamma)$.

In einer gegebenen Tabelle streicht man einfach die Spalten, die nicht zu $\mathcal{B}$ gehören.

Um entscheiden zu können, ob eine gegebene mehrwertige Abhängigkeit $X \twoheadrightarrow Y$ in Γ^+ liegt, ist eine ähnliche Konstruktion wie der zur Berechnung der Hülle bei funktionalen Abhängigkeiten mit dem Linclosure-Algorithmus erforderlich.

Beachte

Aus $X \twoheadrightarrow Y$ und $Z \subsetneq Y$ folgt nicht $X \twoheadrightarrow Z$.

Ähnlich wie die Armstrong-Axiome gibt es auch hier ein Axiomensystem, das es erlaubt, zu entscheiden, welche mehrwertigen Abhängigkeiten logisch impliziert werden.

Für alle $W, X, Y, Z \subseteq \mathcal{A}$ gilt

$$\text{Es gilt immer } X \twoheadrightarrow X. \tag{$M1$}$$

$$X \twoheadrightarrow Y \text{ impliziert } X \cup Z \twoheadrightarrow Y. \tag{$M2$}$$

$$X \twoheadrightarrow Y \text{ und } X \twoheadrightarrow Z \text{ impliziert } X \twoheadrightarrow Y \cup Z. \tag{$M3$}$$

$$X \twoheadrightarrow Y \text{ und } X \twoheadrightarrow Z \text{ impliziert } X \twoheadrightarrow Y \cap Z. \tag{$M4$}$$

$$X \twoheadrightarrow Y \text{ und } Y \twoheadrightarrow Z \text{ impliziert } X \twoheadrightarrow Z - Y. \tag{$M5$}$$

$$X \twoheadrightarrow Y \text{ und } Y \cup W \twoheadrightarrow Z \text{ impliziert } X \cup W \twoheadrightarrow Z - (Y \cup W). \tag{$M6$}$$

$$X \twoheadrightarrow Y \text{ und } Z = \mathcal{A} - \{X, Y\} \text{ impliziert } X \twoheadrightarrow Z. \tag{$M7$}$$

Weiterhin gelten noch die beiden sogenannten Koppelungsaxiome

$$X \to Y \text{ impliziert } X \twoheadrightarrow Y. \tag{$C1$}$$

$$X \twoheadrightarrow Y,\ Z \to W,\ W \subseteq Y \text{ und } Y \cap Z = \emptyset \text{ impliziert } X \to W. \tag{$C2$}$$

Satz

Diese Axiome sind korrekt und vollständig.

Beispiel

Sei $\mathcal{A} = \{A, B, C, D, E\}$ und $\Gamma = \{A \twoheadrightarrow BC, DE \twoheadrightarrow C\}$. Dann impliziert Γ die mehrwertige Abhängigkeit $AD \twoheadrightarrow BE$, denn aus $A \twoheadrightarrow BC$ folgt mittels (M7) $A \twoheadrightarrow DE$. Zusätzlich haben wir $DE \twoheadrightarrow C$, dies liefert mittels (M5) $A \twoheadrightarrow C$, hieraus folgt mit (M2) $AD \twoheadrightarrow C$ und mittels (M7) schließlich $AD \twoheadrightarrow BE$.

Es soll nun exemplarisch (M3) bewiesen werden

Sei R eine Relation die $X \twoheadrightarrow Y$ und $X \twoheadrightarrow Z$ erfüllt. Seien $t_1, t_2 \in R$ mit $t_1|_X = t_2|_X$.

Verwende $X \twoheadrightarrow Y$, das heißt es existiert ein $t_3 \in R$ mit $t_1|_X = t_2|_X = t_3|_X$, $t_3|_Y = t_1|_Y$ und $t_3|_V = t_1|_V$ mit $V := \mathcal{A} \setminus (X \cup Y)$.

Verwende nun $X \twoheadrightarrow Z$, das heißt zu t_1 und t_3 existiert ein $t_4 \in R$ mit $t_4|_X = t_1|_X$, $t_4|_Z = t_1|_Z$ und $t_4|_W = t_3|_W$ mit $V := \mathcal{A} \setminus (X \cup Z)$.

Es ist

$$t_1|_X = t_2|_X$$
$$t_4|_Z = t_1|_Z$$
$$t_4|_{Y\cap W} \underset{\text{Def. v. } t_4}{=} t_3|_{Y\cap W} \underset{t_3|_Y}{=} t_1|_Y \quad \text{also} \quad t_4|_{Y\cup Z} = t_1|_{Y\cup Z}$$

Schließlich haben wir noch

$$t_2|_U \underset{U\subseteq V}{=} t_3|_U \underset{U\subseteq W}{=} t_4|_U\,,$$

wobei $U = \mathcal{A} \setminus (Y \cup Z)$ ist.

Statt alle MVD's aufzulisten, stelle Bausteine zur Verfügung, aus denen diese kombiniert werden können.

Satz

Sei Γ eine Menge aus funktionalen und mehrwertigen Abhängigkeiten und $X, Y \subseteq \mathcal{A}$. Genau dann wird $X \twoheadrightarrow Y$ von Γ impliziert, wenn Y durch die Vereinigung ausgewählter Elemente der Abhängigkeitsbasis von X bezüglich Γ gebildet wird. Dabei wird die Abhängigkeitsbasis von X bezüglich Γ folgendermaßen erhalten.

1. Bilde eine Menge G, die aus allen Mengen Y besteht, die als rechte Seite einer Abhängigkeit $U \twoheadrightarrow Y$ auftreten, wenn man (M2) und (M7) auf Γ anwendet.
2. Solange G noch nicht aus disjunkten Teilmengen besteht, ersetze Z_1 und Z_2 mit $Z_1 \cap Z_2 \neq \emptyset$ durch $Z_1 \setminus Z_2$, $Z_2 \setminus Z_1$ und $Z_1 \cap Z_2$.

Beispiel zur Zerlegung in 4NF

Sei $\mathcal{A} = \{\texttt{Vorlesung}, \texttt{Lehrer}, \texttt{Raum}, \texttt{Student}, \texttt{Stunde}\}$ und

$$\begin{aligned}\Gamma = \{&\{\texttt{Vorlesung}\} \to \{\texttt{Lehrer}\}, \{\texttt{StundeRaum}\} \to \{\texttt{Vorlesung}\}\\ &\{\texttt{Stunde}, \texttt{Lehrer}\} \to \{\texttt{Raum}\}, \{\texttt{Stunde}, \texttt{Stundent}\} \to \{\texttt{Raum}\}\\ &\{\texttt{Vorlesung}\} \twoheadrightarrow \{\texttt{Stunde}, \texttt{Raum}\}\}\end{aligned}$$

Der einzige Schlüssel ist $\{\texttt{Stunde}, \texttt{Student}\}$. `Vorlesung` ist kein Schlüssel, daher ist die 4NF verletzt.

Wir zerlegen in

$$\mathcal{A}_1 = \{\texttt{Vorlesung}, \texttt{Stunde}, \texttt{Raum}\}$$
$$\mathcal{A}_2 = \{\texttt{Vorlesung}, \texttt{Lehrer}, \texttt{Student}\}$$

$\mathcal{A}_1$ ist in 4NF und Schlüssel sind $\{\texttt{Vorlesung}, \texttt{Stunde}\}$ und $\{\texttt{Stunde}, \texttt{Raum}\}$. Hingegen ist $\texttt{Vorlesung} \to \texttt{Lehrer} \in \Gamma_2$ und `Vorlesung` kein Schlüssel von $S_2(\mathcal{A}_2, \Gamma_2)$ und damit ist $\mathcal{A}_2$ nicht in 4NF. Wir zerlegen $\mathcal{A}_2$ weiter und erhalten

$$\mathcal{A}_{21} = \{\texttt{Vorlesung}, \texttt{Lehrer}\}$$
$$\mathcal{A}_{22} = \{\texttt{Vorlesung}, \texttt{Student}\}$$

Hier ist $\Gamma_{21} = \{\texttt{Vorlesung} \rightarrow \texttt{Lehrer}\}$ und $\Gamma_{22} = \emptyset$ (das heißt $\mathcal{A}_{22}$ ist Schlüssel) und beide Schemata sind in 4NF. $\mathcal{A}_{21}$ enthält ein Attribut weniger als im alten Beispiel.

Problem

Wie sieht ein Synthesealgorithmus für MVD's aus und auf welche Normalform sollen und können MVD's gebracht werden? Sind eventuelle Zerlegungen abhängigkeitserhaltend?

Strategie

Verwende Synthesealgorithmus für die funktionalen Abhängigkeiten. Bei der Schlüsselmenge prüfe, ob dort mehrwertige Abhängigkeiten vorliegen und zerlege damit weiter.

Idee

Die mehrwertigen Abhängigkeiten bestehen meist, wenn Informationen unabhängig voneinander sind und nur zufällig einige Attribute in mehreren Rollen auftreten.

Es kann vorkommen, daß auf $\mathcal{B} \subseteq \mathcal{A}$ eine mehrwertige Abhängigkeit gilt, obwohl auf $\mathcal{A}$ keine mehrwertigen Abhängigkeiten existieren. Solche mehrwertigen Abhängigkeiten heißen eingebettet.

Beispiel

BerichtNr	Projekt	Angestellter	Abteilung
1	Amalfi	Schmidt	Fertigung
2	Amalfi	Hinz	Fertigung
3	Silberpfeil	Schmidt	Fertigung
4	Silberpfeil	Hinz	Fertigung

Jeder Angestellte einer Abteilung ist an jedem Projekt der Abteilung beteiligt, das heißt es gilt $\texttt{Abteilung} \twoheadrightarrow \texttt{Angestellter}$ auf $\{\texttt{Projekt}, \texttt{Abteilung}, \texttt{Angestellter}\}$.

Wir hatten bisher, daß $R = \pi_{\mathcal{A}_1}(R) \bowtie \pi_{\mathcal{A}_2}(R)$ genau dann gilt, wenn die mehrwertige Abhängigkeit $\mathcal{A}_1 \cap \mathcal{A}_2 \twoheadrightarrow \mathcal{A}_1 \setminus \mathcal{A}_2$ gilt. Hingegen ist mit mehrwertigen Abhängigkeiten keine Aussage über die Gültigkeit von

$$R = \pi_{\mathcal{A}_1}(R) \bowtie \pi_{\mathcal{A}_2}(R) \bowtie \ldots \bowtie \pi_{\mathcal{A}_{k-1}}(R) \bowtie \pi_{\mathcal{A}_k}(R)$$

möglich. Wir definieren deshalb eine neue Form von Abhängigkeiten.

Definition

Sei $\mathcal{A} = \bigcup_{i=1}^{m} \mathcal{A}_i$ und R eine Relation auf $\mathcal{A}$. Dann erfüllt R die *Verbundabhängigkeit* $\overset{m}{\underset{i=1}{\bowtie}} \mathcal{A}_i$ genau dann, wenn gilt

$$R = \overset{m}{\underset{i=1}{\bowtie}} \pi_{\mathcal{A}_i}(R).$$

Beispiel

Sei R das folgende Schema

Vertreter	Produkt	Firma
Hinz	Staubsauger	AEG
Hinz	Mixer	AEG
Hinz	Föhn	Siemens
Kunz	Staubsauger	AEG
Kunz	Heizofen	IBM
Schmidt	Heizofen	Sonnenschein

Wir erhalten folgende Projektionen

R_1

Vertreter	Produkt
Hinz	Staubsauger
Hinz	Mixer
Hinz	Föhn
Kunz	Staubsauger
Kunz	Heizofen
Schmidt	Heizofen

R_2

Vertreter	Firma
Hinz	AEG
Hinz	AEG
Hinz	Siemenz
Kunz	AEG
Kunz	IBM
Schmidt	Sonnenschein

R_3

Produkt	Firma
Staubsauger	AEG
Mixer	AEG
Föhn	Siemenz
Staubsauger	AEG
Heizofen	IBM
Heizofen	Sonnenschein

Hier ist keiner der möglichen Verbunde zweier Schemata verlustlos, denn

$$R_1 \bowtie R_2 \ni (\texttt{Hinz}, \texttt{Staubsauger}, \texttt{Siemenz}) \notin R$$
$$R_1 \bowtie R_3 \ni (\texttt{Kunz}, \texttt{Heizofen}, \texttt{Sonnenschein}) \notin R$$
$$R_2 \bowtie R_3 \ni (\texttt{Kunz}, \texttt{Mixer}, \texttt{AEG}) \notin R,$$

jedoch gilt $R = R_1 \bowtie R_2 \bowtie R_3$. Dabei erfüllt R keine nichttriviale mehrwertige Abhängigkeit.

Wir können Regeln aufstellen, die sicherstellen, daß jedes R zu dieser Attributmenge, die Verbundabhängigkeit erfüllt

- Arbeitet ein Vertreter für zwei Firmen gleichzeitig, so dürfen diese Firmen kein Produkt gleichzeitig anbieten.

  Wir zeigen nun, daß diese Regel eine Verbundabhängigkeit impliziert. Sei also R in R_1, R_2 und R_3 zerlegt und erfülle die Regel. Sei $(a,b,c) \in R_1 \bowtie R_2 \bowtie R_3$, also $(a,b) \in R_1$, das heißt $(a,b,c') \in R$ mit einem c'. Weiter ist $(b,c) \in R_2$, das heißt $(a',b,c) \in R$ mit einem a' und schließlich $(a,c) \in R_3$, das heißt $(a,b',c) \in R$ mit einem b'. Es genügt nun wegen $R_1 \bowtie R_2 \bowtie R_3 \supseteq R$ zu zeigen, daß $(a,b,c) \in R$ gilt. Wir haben, daß b von c und c' hergestellt wird und, daß a für c und c' gleichzeitig arbeitet, dies liefert uns nach der Regel unmittelbar $c = c'$ und damit $(a,b,c) \in R$.
- Wenn ein Vertreter ein Produkt verkauft und der Vertreter für eine Firma arbeitet, die dieses Produkt herstellt, so verkauft er dieses Produkt für diese Firma.

Die mehrwertige Abhängigkeit $X \twoheadrightarrow Y$ ist Spezialfall einer Verbundabhängigkeit. Der Zerlegungsalgorithmus in 4NF ist zu erweitern auf alle vorkommenden Verbundabhängigkeiten.

Definition 5NF

Sei $S = S(\mathcal{A}, \Gamma)$ mit einer Menge Γ von funktionalen Abhängigkeiten und Verbundabhängigkeiten und seien $X_1, \ldots, X_r$ die Schlüssel von S.

Dann ist S in 5NF genau dann, wenn jede Verbundabhängigkeit in Γ^+ bereits von den Schlüsselabhängigkeiten $X_i \to \mathcal{A}$ impliziert wird.

Beispiel Beispieldaten wie oben

Das Beispielschema

$$\mathcal{A} = \{\texttt{Verteter}, \texttt{Produkt}, \texttt{Firma}\}$$
$$\mathcal{A}_1 = \{\texttt{Vertreter}, \texttt{Produkt}\}$$
$$\mathcal{A}_2 = \{\texttt{Vertreter}, \texttt{Firma}\}$$
$$\mathcal{A}_3 = \{\texttt{Produkt}, \texttt{Firma}\}$$

erfüllt die Verbundabhängigkeit $\overset{3}{\underset{i=1}{\bowtie}} \mathcal{A}_i$ und keine echte Teilmenge von $\mathcal{A}$ ist Schlüssel des Schemas.

Dies aber bedeutet, daß die Verbundabhängigkeit nicht von den Schlüsselabhängigkeiten impliziert wird, das aber heißt, daß die 5NF verletzt wird.

Wir versuchen nun durch eine Zerlegung bezüglich $\overset{3}{\underset{i=1}{\bowtie}} \mathcal{A}_i$ Redundanz zu vermieden.

Regel

Wenn ein Verteter für eine Firma arbeitet, so bietet er auch jedes Produkt dieser Firma an.

Einfügen von x neuen Vertretern für y neue Firmen mit je z Produkten

Es werden x beziehungsweise y beziehungsweise z neue Tupel in die Tabellen R_1, R_2 und R_3 eingefügt. In R ergeben sich dadurch maximal $x \cdot y + x \cdot z + y \cdot z$ neue Tupel.

Test auf Implikation einer Verbundabhängigkeit durch Schlüsselabhängigkeiten

$S(\mathcal{A}, \Gamma)$ habe die Schlüssel $X_1, \ldots, X_r$ und die Verbundabhängigkeit $\gamma = \bowtie_{i=1}^{n} \mathcal{A}_i$.

Test auf Implikation von γ durch $X_1 \to \mathcal{A}, \ldots, X_r \to \mathcal{A}$

Sei $T = \{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$.

Ersetze Y, Z durch $Y \cup Z$, falls für ein i X_i und $Y, Z \in T$ die Inklusion $X_i \subseteq Y \cap Z$ gilt und wiederhole diesen Schritt solange, bis sich T nicht mehr ändert.

Ist $T = \{\mathcal{A}\}$, dann wird γ logisch impliziert. (Es gilt sogar "genau dann, wenn").

Beweis für Korrektheit

Benutze den Test auf Verlustlosen Verbund.

Bilde dazu eine Tabelle T mit Zeilen $\mathcal{A}_1, \ldots, \mathcal{A}_m$ und Spalten $A_1, \ldots, A_n$ und mit Einträgen

$$T(\mathcal{A}_i, A_j) = \begin{cases} a & \text{falls } A_j \in \mathcal{A}_i \\ b_i & \text{sonst} \end{cases}$$

Ist $X_i \subseteq \mathcal{A}_l \cap \mathcal{A}_p$, so werden bei der Gleichsetzung der Einträge im Algorithmus für den Test auf verlostlosen Verbund die Einträge in den Spalten aus $\mathcal{A}_l \cup \mathcal{A}_p$ für $\mathcal{A}_l$ und $\mathcal{A}_p$ auf a gesetzt.

Falls durch Iteration eine Tafel mit nur a's in einer Zeile entsteht, liegt ein verlustloser Verbund vor, das heißt γ wird impliziert.

1. Bestimmung eines Schlüssels

   Setze $X = \mathcal{A}$.

   Durchlaufe X mit A und teste, ob $(X \setminus \{A\})^+ \supseteq X$.

   Falls ja, so entferne A, das heißt setze $X = X \setminus \{A\}$.

   Wiederhole dies, bis sich X nicht mehr ändert.

2. Sei $\mathcal{K}$ die Menge der bekannten Schlüssel.

   Ist $K \in \mathcal{K}$ und $L \to R \in \Gamma$, so bilde $Z = (K \setminus R) \cup L$

   Bestimme mit 1. in Z enthaltene Schlüssel.

   Bezogen auf die Anzahl der Schlüssel ist dies effizient.

Satz

Sei $\mathcal{B} \subseteq \mathcal{A}$ und $\gamma = \bowtie_{i=1}^{m} \mathcal{A}_i$ eine Verbundabhängigkeit auf $\mathcal{A}$.

Falls für jedes $A \in \mathcal{A} \setminus \mathcal{B}$ genau ein i existiert mit $A \in \mathcal{A}_i$, so impliziert γ auf $\mathcal{B}$ die Verbundabhängigkeit $\bowtie_{i=1}^{m} (\mathcal{A}_i \cap \mathcal{B})$.

Das bedeutet, daß der Zerlegungsalgorithmus in 4NF sich zu einem Zerlegungsalgorithmus in 5NF erweitern läßt.

Bemerkung

Ist S in 5NF, so auch in 4NF.

Beweis

Sei S in 5NF und $\Gamma \equiv \Gamma' = \{X_1 \to \mathcal{A}, \ldots, X_r \to \mathcal{A}\}$. Sei $X \twoheadrightarrow Y \in \Gamma^+$ mit $X \twoheadrightarrow Y$ nichttrivial, so wird $X \twoheadrightarrow Y$ auch von Γ' impliziert.

Berechne zu X die Abhängigkeitsbasis bezüglich Γ'.

Da Γ' nur funktionale Abhängigkeiten enthält, brauchen nur die Armstrong-Axiome verwendet werden. Also wird $X^+ = \texttt{Linclosure}(X, \Gamma')$ gebildet. Wir wissen, daß $Y \subseteq X^+$ ist.

$X = X^{(0)} \subseteq X^{(1)} \subseteq \ldots \subseteq X^+$ wird durch Verwendung von Abhängigkeiten $U \to V \in \Gamma'$ gebildet. Es existiert folglich eine funktionale Abhängigkeit $U \to V \in \Gamma$ mit $U \subseteq X$. Da $U \to V \in \Gamma'$ liegt, ist U Schlüssel. Es liegt also 4NF vor.

Der Chase-Prozeß

Definition

Sei $\mathcal{A} = \{A_1, \ldots, A_n\}$ und $\Sigma = \{\, a_i, b_i \mid i \in I\!N \,\}$ ein Alphabet. Ein Tableau T ist eine Menge von Tupeln $t\colon \mathcal{A} \to \Sigma$ mit folgenden Eigenschaften

Falls $t(A_i) = a_j$ für alle $t \in T$ gilt, so ist $i = j$, das heißt die i-te Spalte enthält höchstens a_i's, und $t_1(A_i) \neq t_2(A_j)$ für alle $t_1, t_2 \in T$, falls $i \neq j$, das heißt kein Symbol aus Σ tritt in verschiedenen Spalten auf. Die Menge

$$\texttt{Bild}(T) = \{\, t(A) \mid A \in \mathcal{A}, t \in T \,\}$$

heißt die *Menge der auftretenden Symbole*. Jedes

$$\rho\colon \texttt{Bild}(T) \to \bigcup_{i=1}^{m} D(A_i)$$

mit $\rho(t(A)) \in D(A)$ für alle A mit $t(A) \in \texttt{Bild}(T)$ definiert eine Relation $\rho(T)$ über $\mathcal{A}$.

Es folgt $t(A) \in D(A)$. Jedes $c \in \texttt{Bild}(T)$ wird durch $\rho(c)$ ersetzt.

Definition

Sei R eine Relation über $\mathcal{A}$. Wir definieren die Menge $T(R)$ durch

$$T(R) = \{(\rho(a_1), \ldots, \rho(a_n)) \mid \rho(T) \subseteq R\}.$$

Beispiel

Gegeben sei die Relation R durch

R

A_1	A_2	A_3	A_4
1	4	5	8
2	3	5	7
1	4	5	7
2	3	6	7

und das Tableau T durch

T

A_1	A_2	A_3	A_4
a_1	b_1	a_3	b_2
b_3	a_2	a_3	b_4
a_1	b_5	a_3	a_4

Sei

ρ

a_1	a_2	a_3	a_4	b_1	b_2	b_3	b_4	b_5
1	3	5	7	4	8	2	7	4

dann erhalten wir

$\rho(T)$

A_1	A_2	A_3	A_4
1	4	5	8
2	3	5	7
1	4	5	7

und damit $\rho(T) \subseteq R$. Betrachte nun

ρ'

a_1	a_2	a_3	a_4	b_1	b_2	b_3	b_4	b_5
2	4	5	7	3	7	1	8	3

damit erhalten wir

$\rho'(T)$

A_1	A_2	A_3	A_4
2	3	5	7
1	4	5	8
2	3	5	7

das heißt $\rho'(T) \subseteq R$. Schließlich erhält man

$T(R)$

A_1	A_2	A_3	A_4
1	3	5	7
2	4	5	7
1	4	5	8
1	4	5	7
1	3	5	8
2	3	5	7
2	3	6	7

Definition

Sei $\mathcal{A} = \bigcup_{i=1}^{m} \mathcal{A}_i$. Dann bezeichnet $T_{\{\mathcal{A}_1,\ldots,\mathcal{A}_m\}}$ die Starttafel für den Test auf verlustlosen Verbund.

Satz

$T_{\{\mathcal{A}_1,\ldots,\mathcal{A}_m\}}(R)$ und $\overset{m}{\underset{i=1}{\bowtie}} \pi_{\mathcal{A}_i}(R)$ definieren dieselbe Abbildung.

Beweis

Siehe Test auf verlustlosen Verbund.

Definition

Seien T_1 und T_2 Tableaus über $\mathcal{A}$.

Dann ist $T_1 \geq T_2$ genau dann, wenn $T_1(R) \supseteq T_2(R)$ für alle R über $\mathcal{A}$ ist.

Es ist $T_1 \equiv T_2$ genau dann, wenn $T_1 \geq T_2$ und $T_2 \geq T_1$ ist.

Satz

Sei $\mathcal{A} = \bigcup_{i=1}^{m} \mathcal{A}_i = \bigcup_{j=1}^{k} \mathcal{B}_j$

Dann sind äquivalent

(i) $\bowtie_{i=1}^{m} \pi_{\mathcal{A}_i}(R) \supseteq \bowtie_{j=1}^{k} \pi_{\mathcal{B}_j}(R)$ für alle R über $\mathcal{A}$

(ii) $T_{\{\mathcal{A}_1,\ldots,\mathcal{A}_m\}} \geq T_{\{\mathcal{B}_1,\ldots,\mathcal{B}_k\}}$

(iii) Für alle i existiert j mit $\mathcal{A}_i \subseteq \mathcal{B}_j$ oder anders geschrieben $\{\mathcal{A}_1,\ldots,\mathcal{A}_m\} \leq \{\mathcal{B}_1,\ldots,\mathcal{B}_k\}$.

(iv) Aus $R = \bowtie_{i=1}^{m} \pi_{\mathcal{A}_i}(R)$ folgt $R = \bowtie_{j=1}^{k} \pi_{\mathcal{B}_i}(R)$

Beweis

(i) $\Leftrightarrow$ (ii) Siehe voriger Satz.

(i) $\Rightarrow$ (iv) Sei $R = \bowtie_{i=1}^{m} \pi_{\mathcal{A}_i}(R)$. Dann ist $R \supseteq \bowtie_{j=1}^{k} \pi_{\mathcal{B}_j}(R)$. "$\subseteq$" gilt immer.

(iv) $\Rightarrow$ (i) Sei

$$R' := \bowtie_{j=1}^{m} \pi_{\mathcal{A}_j}(R),$$

dann ist

$$\begin{aligned} \bowtie_{i=1}^{m} \pi_{\mathcal{A}_i}(R) &\overset{\text{ÜA}}{=} \bowtie_{i=1}^{m} \pi_{\mathcal{A}_i}\Big(\bowtie_{j=1}^{m} \pi_{\mathcal{A}_j}(R)\Big) \\ &= \bowtie_{i=1}^{m} \pi_{\mathcal{B}_i}(R') \\ &= R', \end{aligned}$$

also erfüllt R' die Voraussetzungen von (iv). Somit ist

$$R \subseteq \bowtie_{i=1}^{m} \pi_{\mathcal{A}_i}(R) = R' = \bowtie_{i=1}^{k} \pi_{\mathcal{B}_i}(R').$$

Wir wenden $\bowtie_{i=1}^{k} \pi_{\mathcal{B}_i}$ an. Dies liefert

$$\bowtie_{i=1}^{k} \pi_{\mathcal{B}_i}(R) \subseteq \bowtie_{i=1}^{k} \pi_{\mathcal{B}_i}\Big(\bowtie_{j=1}^{k} \pi_{\mathcal{B}_j}(R')\Big).$$

Schließlich erhalten wir

$$\bowtie_{i=1}^{k} \pi_{\mathcal{B}_i}(R) \subseteq \bowtie_{i=1}^{k} \pi_{\mathcal{B}_i}(R') = R' = \bowtie_{j=1}^{m} \pi_{\mathcal{A}_j}(R).$$

(i) $\Rightarrow$ (iii) Für alle R gelte

$$\bowtie_{i=1}^{m} \pi_{\mathcal{A}_i}(R) \supseteq \bowtie_{i=1}^{k} \pi_{\mathcal{B}_i}(R).$$

Wir konstruieren nun ein R, das für den Beweis nützlich ist. Seien dazu $t_1,\ldots,t_k$ Tupel mit

$$t_i(A) = \begin{cases} 0 & \text{falls } A \in \mathcal{B}_i \\ 1 & \text{sonst.} \end{cases}$$

Dann gilt nach der Definition des Verbundes

$$t_0 := (0,\ldots,0) \in \bowtie_{i=1}^{k} \pi_{\mathcal{B}_i}(R).$$

Nach der Voraussetzung gilt somit auch

$$t_0 \in \bowtie_{i=1}^{m} \pi_{\mathcal{A}_i}(R).$$

Damit existiert für alle $\mathcal{A}_i$ ein $t_j \in R$ mit

$$\pi_{\mathcal{A}_i}(t_j) = \pi_{\mathcal{A}_i}(t_0) = (0, \ldots, 0)_{\mathcal{A}_i}.$$

Somit haben wir $\mathcal{A}_i \subseteq \mathcal{B}_j$, da R nur aus $t_1, \ldots, t_k$ besteht.

(iii) $\Rightarrow$ (i) Exitiere nun für alle i ein j mit $\mathcal{A}_i \subseteq \mathcal{B}_j$. Sei R eine Relation über $\mathcal{A}$ und $t \in \bowtie_{j=1}^{k} \pi_{\mathcal{B}_j}(R)$. Dann existieren $t_1, \ldots, t_k \in R$ mit

$$\pi_{\mathcal{B}_j}(t) = \pi_{\mathcal{B}_j}(t_j) \quad \text{für alle } j.$$

Da zu jedem i ein j existiert mit $\mathcal{A}_i \subseteq \mathcal{B}_j$, haben wir also, daß zu jedem i ein $t_j \in R$ existiert mit

$$\pi_{\mathcal{A}_i}(t) = \pi_{\mathcal{A}_i}(t_j).$$

Dann ist aber

$$t \in \bowtie_{i=1}^{m} \pi_{\mathcal{A}_i}(R).$$

Beispiel

Seien $\mathcal{A}_1 = \{A_1, A_2\}$, $\mathcal{A}_2 = \{A_2, A_3\}$, $\mathcal{A}_3 = \{A_3, A_4\}$, $\mathcal{B}_1 = \{A_1, A_2, A_3\}$ und $\mathcal{B}_2 = \{A_3, A_4\}$ gegeben.

Dann gilt $\{\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3\} \leq \{\mathcal{B}_1, \mathcal{B}_2\}$.

Bemerkung

Die Aussagen des Satzes gelten auch wenn "$\leq$" beziehungsweise "$\subseteq$" durch "$\equiv$" beziehungsweise "$=$" ersetzt werden.

Beispiel

Seien

$$\begin{array}{lll} \mathcal{A}_1 = \{A_1, A_2, A_3\} & \mathcal{A}_2 = \{A_1, A_4\} & \mathcal{A}_3 = \{A_1, A_3, A_4\} \\ \mathcal{B}_1 = \{A_1, A_2, A_3\} & \mathcal{B}_2 = \{A_3, A_4\} & \mathcal{B}_3 = \{A_1, A_3, A_4\}. \end{array}$$

Dann gilt $\{\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3\} \equiv \{\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3\}$.

Wir haben dann

$T_{\{\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3\}}$

A_1	A_2	A_3	A_4
a_1	a_2	a_3	b_1
a_1	b_2	b_3	a_4
a_1	b_4	a_3	a_4

$\neq$

$T_{\{\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3\}}$

A_1	A_2	A_3	A_4
a_1	a_2	a_3	b_1
b_2	b_3	a_3	a_4
a_1	b_4	a_3	a_4

Nach dem Satz definieren die Tableaux aber dieselbe Abbildung $T(R)$.

Jeweils das zweite Tupel hat nur dort Einträge a, bei denen auch bereits ein anderes Tupel ebenfalls Einträge a besitzt, das heißt $\mathcal{A}_2 \subseteq \mathcal{A}_3$ beziehungsweise $\mathcal{B}_2 \subseteq \mathcal{B}_3$. Alle Informationen zu Tupeln über $\mathcal{A}_2$ sind bereits bei Tupeln, die über $\mathcal{A}_3$ definiert sind vorhanden. Damit sind diese Tupel entbehrlich.

Definition

$t_1 \in T$ *subsummiert* $t_2 \in T$ genau dann, wenn für alle A gilt

$$t_2(A) = a \Longrightarrow t_1(A) = a.$$

Weiter definieren wir

$$\mathrm{sub}(T) := \Big\{ t \in T \;\Big|\; \text{es exisiert kein } t' \in T, \text{ das } t \text{ subsummiert} \Big\}.$$

Es werden also die subsummierten Tupel gestrichen.

Schließlich definieren wir $T_{\{\mathcal{A}_1,\ldots,\mathcal{A}_m\}} \equiv T_{\{\mathcal{B}_1,\ldots,\mathcal{B}_k\}}$ genau dann wenn

$$\mathrm{sub}(T_{\{\mathcal{A}_1,\ldots,\mathcal{A}_m\}}) = \mathrm{sub}(T_{\{\mathcal{B}_1,\ldots,\mathcal{B}_k\}})$$

gilt.

Definition

Sei V die Menge der Variablen von T und V' die Menge der Variablen von T'. T' enthält T genau dann, wenn eine Funktion $\psi: V \to V'$ mit folgenden Eigenschaften existiert

a) Ist $\psi(v) = \delta$ so gilt für alle Spalten A von T bzw. T': Steht v in der Spalte A von T, so steht auch das δ in der Spalte A von T'.

b) Ist $v = a_i$, so ist auch $\delta = a_i$.

c) Es ist $\psi(T) \subseteq T'$.

Es werden also den Variablen von T statt fester Werte die Variablen von T' zugeordnet.

Beispiel

Sei T das Tableau

A_1	A_2	A_3	A_4
a_1	a_2	b_1	b_2
b_3	a_2	a_3	b_2
b_4	b_5	a_3	a_4

Sei ψ eine Abbildung mit $\psi(a_i) = a_i$ für alle i und $\psi(b_1) = a_3$, $\psi(b_2) = b_1$, $\psi(b_3) = a_1$ $\psi(b_4) = b_2$ und $\psi(b_5) = a_2$. Dann ist $\psi(T)$ das Tableau

A_1	A_2	A_3	A_4	
a_1	a_2	a_3	b_1	
~~a_1~~	~~a_2~~	~~a_3~~	~~b_1~~	gestrichen, da doppelt
b_2	a_2	a_3	a_4	

Satz

T' enthält T genau dann, wenn $T > T'$ gilt.

Es werden nun zusätzliche Abhängigkeiten berücksichtigt. $R \subseteq S(\mathcal{A}, \Gamma)$ werde betrachtet, das heißt alle betrachteten R erfüllen Γ.

Definition

Wir definieren $T_1 \equiv_\Gamma T_2$ genau dann wenn für alle $R \in S(\mathcal{A}, \Gamma)$ $T_1(R) = T_2(R)$ gilt.

Tableaux werden durch äquivalente ersetzt, wobei berücksichtigt wird, welche Abhängigkeiten gelten.

Der Test auf verlustlosen Verbund startet mit einem Tableau und ersetzt dieses schrittweise durch neue Tableaux, die bei Betrachtung der vorliegenden Abhängigkeiten dazu äquivalent sind.

Regeln für das Ersetzen von Tableaux durch äquivalente unter Berücksichtigung von Abhängigkeiten

1. Eine Regel $X \to Y$ erlaubt bei Tupeln, die auf X übereinstimmen, die Einträge auf Y gleichzusetzen. Dabei werden wenn möglich b's durch a's ersetzt. Ist b_i mit b_j gleichzusetzen, so wähle $b_{\min(i,j)}$ als neuen Eintrag.
2. Eine Regel $\bowtie_{i=1}^{m} \mathcal{A}_i$ erlaubt Tupel zu erzeugen. Sind $t_1, \ldots, t_m \in T$ und ist für i und j $t_i(\mathcal{A}_i \cap \mathcal{A}_j) = t_j(\mathcal{A}_i \cap \mathcal{A}_j)$, so füge in T das Tupel t ein, das für alle i $t(\mathcal{A}_i) = t_i(\mathcal{A}_i)$ erfüllt. (Sind die t_i verbindbar, so muß das durch den Verbund entstehende Tupel in der Relation liegen.)

Es gilt: Erfüllt R eine Abhängigkeit $\gamma \in \Gamma$, so ist $T(R) = T'(R)$, wobei T' aus T durch Anwenden der zugehörigen Regel 1. oder 2. entsteht.

Beispiel für Verbundregel

Wir wollen für das Tableau

	A_1	A_2	A_3	A_4
$t_1 \to$	a_1	b_1	b_2	a_4
$t_2 \to$	a_1	a_2	b_3	b_4
$t_3 \to$	b_3	a_2	b_3	a_4

$\bowtie\{A_1A_2, A_2A_3, A_3A_4\}$ anwenden.

Die Tupel t_2, t_2 und t_3 sind verbindbar, daher müssen wir das Tupel t_4 einfügen. Die Tupel t_3, t_3 und t_2 wären somit natürlich auch verbindbar. Wir erhalten das Tableau

	A_1	A_2	A_3	A_4
$t_1 \to$	a_1	b_1	b_2	a_4
$t_2 \to$	a_1	a_2	b_3	b_4
$t_3 \to$	b_3	a_2	b_3	a_4
$t_4 \to$	a_1	a_2	b_3	a_4

Das Tupel t_4 ist korrekt, da $t_4|_{A_1A_2} = t_2|_{A_1A_2}$, $t_4|_{A_2A_3} = t_2|_{A_2A_3}$ und $t_4|_{A_3A_4} = t_3|_{A_3A_4}$ ist.

Bilde nun $\bowtie\{A_1A_2A_4, A_1A_3A_4\}$, dann sind die Tupel t_1 und t_4 verbindbar, und wir fügen das Tupel t_5 ein

	A_1	A_2	A_3	A_4
$t_1 \to$	a_1	b_1	b_2	a_4
$t_2 \to$	a_1	a_2	b_3	b_4
$t_3 \to$	b_3	a_2	b_3	a_4
$t_4 \to$	a_1	a_2	b_3	a_4
$t_5 \to$	a_1	b_1	b_3	a_4

Es gilt

$$t_5|_{A_1A_2A_4} = t_1|_{A_1A_2A_4} \quad \text{und} \quad t_5|_{A_1A_3A_4} = t_4|_{A_1A_3A_4}\,.$$

Der Chase-Prozeß

Sei Γ eine Menge von funktionalen und Verbundabhängigkeiten. Sei T ein Tableau über $\mathcal{A}$. Eine Folge $T = (T_0, T_1, \ldots, T_r)$ von Tableaux mit folgenden Eingenschaften

1. Für alle i gilt $T_i \neq T_{i+1}$
2. Für alle i entsteht T_{i+1} aus T_i durch Anwenden einer Regel 1. oder 2. für das Ersetzen von Tableaux.
3. T_r läßt sich durch Anwenden keiner γ-Regel für $\gamma \in \Gamma$ mehr verändern.

heißt *Chase-Folge*.

Satz (ohne Beweis)

Es gilt

- Jede Chase-Folge ist endlich.
- Jede Chase-Folge zum gleichen T und Γ endet beim gleichen T_r. (Church-Rosser-Eigenschaft)

Test auf $X \to A_j \in \Gamma^+$ (Γ enthält neben funktionalen auch Verbundabhängigkeiten)

Als Starttableau wird

$T_X =$

		$\overbrace{\quad\quad}^{X}$						
A_1	$\cdots$	A_i	$\cdots$	A_{i+k}	$\cdots$	A_j	$\cdots$	A_n
a_1	$\cdots$	a_i	$\cdots$	a_{i+k}	$\cdots$	a_j	$\cdots$	a_n
b_1	$\cdots$	a_i	$\cdots$	a_{i+k}	$\cdots$	b_l	$\cdots$	b_r

verwendet. Wende dann auf T_X den Chase-Prozeß an. Sei T_X^* das letzte Folge-Glied. Genau dann gilt $X \to A_j \in \Gamma^+$, wenn in der Spalte von A_j in T_X^* nur a_j steht.

Der Beweis hierzu ist trivial.

Beispiel

Sei $\mathcal{A} = \{A, B, C, D\}$ und $\Gamma = \{A \to D, \bowtie \{ABC, CD\}\}$. Wird dann mit $X = \{B, C\}$ und $Y = \{D\}$ die funktionale Abhängigkeit $X \to Y$ impliziert?

Die Tabelle

T_X

A	B	C	D
a_1	a_2	a_3	a_4
b_1	a_2	a_3	b_2

wird durch Anwenden des Join-Operators zu

A	B	C	D
a_1	a_2	a_3	a_4
b_1	a_2	a_3	b_2
a_1	a_2	a_3	b_2
b_1	a_2	a_3	a_4

und wegen der funktionalen Abhängigkeit $A \to D$ ergibt sich mit Regel 1. $a_4 = b_2$ und damit die Tabelle

A	B	C	D
a_1	a_2	a_3	a_4
b_1	a_2	a_3	a_4

Γ impliziert also $\{B, C\} \to \{D\}$.

Folgerung

Sei Γ eine Menge von Verbundabhängigkeiten und $X \to Y$ werde von Γ impliziert. Dann ist $Y \subseteq X$.

Beweis

Chase startet mit T_X, wobei für $A \in Y$ und $A \notin X$ in der Spalte von A in der zweiten Zeile b steht.

Verbundabhängigkeiten liefern aber nur neue Tupel, verursachen aber keine Gleichsetzungen, was den Widerspruch liefert.

Berechnung einer Abhängigkeitsbasis für mehrwertige Abhängigkeiten $X \twoheadrightarrow Y$

Sei Γ eine Menge von funktionalen Abhängigkeiten und Verbundabhängigkeiten. Sei $X, Y \subseteq \mathcal{A}$ und $X^+ \cap Y = \emptyset$. Dann gilt:

Γ impliziert $X \twoheadrightarrow Y$ genau dann, wenn $\texttt{chase}_\Gamma(T_X)$ eine Zeile t_Y mit a-Einträgen genau auf $X^+ \cup Y$ enthält.

Beweis

"$\Rightarrow$" Um zu zeigen, daß $X \twoheadrightarrow Y$ gilt, verwende den Test auf verlustlosen Verbund.

Setze $Z = \mathcal{A} - (X^+ \cup Y)$ und wähle $\mathcal{A}_1 = X^+ \cup Z$ und $\mathcal{A}_2 = X \cup Y$.

Es ergibt sich

$T_{\{\mathcal{A}_1, \mathcal{A}_2\}}$

	X^+	Y	Z
$t_{\mathcal{A}_1}$	$a_1 \ldots a_i$	$b_1 \ldots b_r$	$a_{i+r+1} \ldots a_n$
$t_{\mathcal{A}_2}$	$a_1 \ldots a_i$	$a_{i+1} \ldots a_{i+r}$	$b_{r+1} \ldots b_s$

und durch den Chase-Prozeß

	X^+	Y	Z
$t^*_{\mathcal{A}_1}$	$a_1 \ldots a_i$	?	$a_{i+r+1} \ldots a_n$
$t^*_{\mathcal{A}_2}$	$a_1 \ldots a_i$	$a_{i+1} \ldots a_{i+r}$	?

Behauptung

$\texttt{chase}(T_{\{\mathcal{A}_1, \mathcal{A}_2\}})$ enthält eine Zeile, die nur aus a's besteht.

Beweis

Verwende die Voraussetzung, daß

T_X

X^+	$\mathcal{A} - X^+$
$a_1 \ldots a_i$	$a_{i+1} \ldots a_n$
$a_1 \ldots a_i$	$c_1 \ldots c_l$

durch den Chase-Prozeß zu

	X^+	Y	Z
t_Y	$a_1 \ldots a_i$	$a_{i+1} \ldots a_{i+k}$	$c_k \ldots c_l$

wird.

Wenn in einer Relation zwei Tupel existieren, die auf X übereinstimmen, so enthält diese Relation, falls sie Γ genügt, auch das Tupel t_Y.

Deshalb kann man durch eine geeignete Abbildung δ:$\texttt{Var}(T_X) \rightarrow \texttt{Var}(T_{\{\mathcal{A}_1, \mathcal{A}_2\}})$ mit $\delta(t_a) = t^*_{\mathcal{A}_2}$ und $\delta(t_X) = t^*_{\mathcal{A}_2}$ die a's und c's von T_X so abbilden, daß geeignete Tupel einer Relation entstehen, die T genügt.

Da $\texttt{chase}(T_{\{\mathcal{A}_1, \mathcal{A}_2\}}$ den Abhängigkeiten aus Γ genügt, muß auch $\delta(t_Y)$ darin enthalten sein.

$\delta(t_Y)$ besteht aber nur aus a's, deshalb enthält es auch auf X^+ nur a's.

Da $\delta(t_a) = t^*_{\mathcal{A}_2}$ ist $\delta(a_j) = a_j$ für $A_j \in Y$, was bedeutet, daß $\delta(t_Y)$ auf Y auch nur a's enthält.

Wegen $\delta(t_X) = t^*_{\mathcal{A}_2}$ werden die c-Einträge ebenfalls zu a's.

Dies aber bedeutet, daß $\delta(t_Y)$ auf Z und somit auf ganz $\mathcal{A}$ nur a's enthält.

"$\Leftarrow$" Man nimmt an, Γ impliziere $X \twoheadrightarrow Y$.

Sei dann ohne Beschränkung der Allgemeinheit $\bowtie \{X^+ \cup Y, X \cup Z\} \in \Gamma$.

Betrachte $\texttt{chase}(T_X)$, wobei wir zuerst $\bowtie \{X^+ \cup Y, X \cup Z\}$ anwenden und so die neue Tabelle

$T_X =$

	X		Y	Z
t_a	$a \ldots a$	a	$a \ldots a$	$a \ldots a$
t_X	$a \ldots a$	b	$b \ldots b$	$b \ldots b$
t	$a \ldots a$	a	$a \ldots a$	$b \ldots b$

erhalten.

Auf Z können durch funktionale Abhängigkeiten keine b's durch a's ersetzt werden, da Verbundabhängigkeiten nur neue Zeilen erzeugen.

Das aber heißt, daß t unverändert bleibt.

Folgerung

Zu X erhält man aus $\texttt{chase}_\Gamma(T_X)$ sofort alle mehrwertigen Abhängigkeiten der Form $X \twoheadrightarrow Y$. Bilde Abhängigkeitsbasis durch den bereits früher beschriebenen Algorithmus.

Beispiel

Sei $\Gamma = \{\{B\} \to \{C\}, \bowtie \{ABC, CDE\}\}$ und $X = \{B\}$. Die Tabelle

T_X

A	B	C	D	E
a_1	a_2	a_3	a_4	a_5
b_1	a_2	b_2	b_3	b_4

wird wegen der funktionalen Abhängigkeit $B \to C$ zu

A	B	C	D	E
a_1	a_2	a_3	a_4	a_5
b_1	a_2	a_3	b_3	b_4

und diese wird durch Anwenden des Join-Operators zu

A	B	C	D	E
a_1	a_2	a_3	a_4	a_5
b_1	a_2	a_3	b_3	b_4
a_1	a_2	a_3	b_3	b_4
b_1	a_2	a_3	a_4	a_5

mit $X^+ = \{B, C\}$. Es ergeben sich schließlich die Abhängigkeiten $X \to X^+$, $\{B\} \twoheadrightarrow \{A\}$ und $\{B\} \twoheadrightarrow \{D, E\}$.

Definition Nullwerte

Nullwerte sind nicht existierende oder nicht bekannte Werte.

Wir haben bereits früher den "Platzhalter" δ betrachtet, der vom äußeren Verbund benutzt wird. Das Problem dabei war bisher, daß δ nicht assoziativ ist. Als Ausweg führt man nun induzierte Nullwerte δ_i ein. Wie verwendet man nun diese Nullwerte?

Man führt eine sogenannte Universalrelation $\pi: R \to (R_1, \ldots, R_m)$ mit $R_i = \pi_{\mathcal{A}_i}(R)$ ein, speichert die R_i lokal, macht alle Updates lokal und hofft, daß kein Widerspruch auftritt.

Man versucht, eine abhängigkeitserhaltende Zerlegung und einen verlustlosen Verbund zu erhalten.

Wenn die Universalrelationsbedingung gilt, so ist die Projektion π bijektiv. Bei Nullwerten ist die Universalrelationsbedingung verletzt, da zum Beispiel für

R_1

A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

und

T_X

A	B	D	E
a_1	b_1	d_1	e_1

$\pi_{\{A,B,C\}}(R_1 \bowtie R_2) \neq R_1$ gilt, das heißt, es gehen Informationen verloren.

Allgemein wird bei zusätzlichen Tupeln durch die Verbundbildung Information vernichtet.

Welche Informationen sind aus einer Datenbank ableitbar, wenn unvollständige Daten gespeichert sind?

Ersetze in dieser Situation den Verbund durch den Chase-Prozeß.

1. Setze jedes Tupel eines jeden R_i durch paarweise verschiedene Nullwerte auf die gesamte Attributmenge fort.
2. Interpretiere die echten Werte als a's und die Nullwerte als b's und führe den Chase-Prozeß durch.
3. Entferne subsumierte Tupel.

Folgerung

Alle Tupel in der entstehenden Tafel, die keine Nullwerte enthalten, bilden exakt den Verbund der R_i.

Ist $X \subseteq \mathcal{A}$, so sei $\pi^t_X(T)$ die Menge der Tupel aus T, die auf X keinen Nullwert haben, das heißt, die als Abbildung auf X eingeschränkt, total sind. Das hochgestellte t steht dabei für total.

$\pi^t_X(T)$ enthält die über X vorhandenen Informationen.

Beispiel

Gegeben seien die Tabellen

A	B	C	D
1	1	1	2

C	G	D	E	F
1	1	1	1	1

D	E	F	B
1	1	1	1

und

B	C	F

sowie die funktionalen Abhängigkeiten $DEF \to B$ und $BC \to F$.

Es resultiert die Tabelle

T

A	B	C	D	E	F	G
1	1	1	2	δ_1	δ_2	δ_3
δ_4	δ_5	1	1	1	1	1
δ_6	1	δ_7	1	1	1	δ_8

mit $\delta_5 = 1$ wegen $DEF \to B$ und $\delta_2 = 1$ wegen $BC \to F$.

Es ergibt sich also

$\pi^t_{ABCDF}(T)$

A	B	C	D	F
1	1	1	2	1

Da δ_4 und δ_6 nicht ersetzt wurden, erhält man nur die erste Zeile als totales Tupel, was man durch den Verbund nicht erreicht hätte!

Verfügbare Systeme

- Es gibt keine Speicherung indizierter Nullwerte.
- Es gibt keinen Chase-Prozeß.
- Es werden zu den einzelnen Schemata Primärschlüssel eingegeben.
- Auf Schlüsseln sind Nullwerte verboten.
- Es gibt keine Prüfung, ob eine Universalrelation existiert.

Ziel

Die R_i sollen möglichst unabhängig voneinander geändert werden können.

Frage

Wo werden durch eine Änderung eines Tupels Folgeänderungen nötig?

Γ enthalte stets nur funktionale und Verbundabhängigkeiten. $X \subseteq \mathcal{A}$. Änderung betreffe Einträge auf X. Der Chase-Prozeß liefert Gleichsetzungen, das heißt Abänderung von Einträgen und evtl. Konflikte nur auf X^+.

Fremdschlüsselbedingung

Die Forderung

$$R_i \subseteq \bowtie_{\mathcal{A}_j \subseteq \mathcal{A}_i^+} R_j$$

heißt Fremdschlüsselbedingung. Sie wird von einigen verfügbaren Systemen unterstützt.

Versicherungbeispiel

Gegeben seien die folgenden funktionalen Abhängigkeiten

$$\{\texttt{SchadensNr}\} \to \{\texttt{VersNr}, \texttt{Schadenshöhe}\}$$
$$\{\texttt{VersNr}\} \to \{\texttt{KundenNr}, \texttt{Sparte}\}$$
$$\{\texttt{KundenNr}\} \to \{\texttt{Name}, \texttt{Anschrift}, \texttt{Agent}\}$$
$$\{\texttt{Agent}\} \to \{\texttt{Bezirk}, \texttt{Chef}, \texttt{Status}\},$$

aus denen man die Schemata $S_i(\mathcal{A}_i, \Gamma_i)$ mit $i = 1, 2, 3, 4$ erhält.

Fremdschlüsselbedingung

Ein neuer Agent darf eingetragen werden ohne Nebenbedingung. Ein Kunde darf nur eingetragen werden, wenn ein zugehöriger Agent existiert ($\mathcal{A}_4 \subseteq \mathcal{A}_3^+$). Eine Versicherungsnummer darf nur eingetragen werden, wenn Kundennummer und Agent existieren. Ein Schaden darf nur eingetragen werden, wenn Versicherungsnummer, Kundennummer und Agent existieren.

Beantwortung von Anfragen

Zu $X \subseteq \mathcal{A}$ werden alle Tupel aus $\pi_X^t(T)$ gebraucht, wobei T durch den Chase-Prozeß entstand. T habe genau auf X keine Nullwerte. Dann gilt

$$t \in \pi_X^t\Big(\bowtie_{\mathcal{A}_j \subseteq X^+} R_j\Big)$$

Beweis

t wurde vom Chase-Prozeß aus Tupeln $t_i \in R_i$ gebildet. Außerhalb von X^+ hat t nur Nullwerte. Daher hat jedes t_i außerhalb von X^+ nur Nullwerte. Es ist $\mathcal{A}_i \subseteq X^+$. Da auf X^+ die darauf totalen Tupel durch den Verbund erhalten werden folgt die Behauptung.

Die *allgemeine* Anfrage wird beantwortet durch

$$\pi_X^t\Big(\bigcup_{X \subseteq Y}\Big(\bowtie_{\mathcal{A}_j \subseteq Y^+} R_j\Big)\Big).$$

Definition

Der Verbund

$$(\cdots((R_1 \bowtie R_2) \bowtie R_3) \bowtie \cdots \bowtie R_{n-1}) \bowtie R_n,$$

wobei $\bigcup_{i=1}^{j-1} \mathcal{A}_i$ einen Schlüssel von $S_j(\mathcal{A}_j, \Gamma_j)$ enthält, heißt *Erweiterungsverbund*.

Beim Erweiterungverbund werden die Zwischenergebnisse von der Tupelzahl her nie größer.

Beispiel

Seien die folgenden beiden Tabellen gegeben

Geschichte

Angestellter	Arbeitgeber	früherer Job	früheres Gehalt
Otto	Post	Briefträger	20000
Max	Müllabführ	Straßenreiniger	30000

Fehlzeiten

Angestellter	Krankheit	Sonderurlaub
Ferdinand	12	2
Otto	10	7

Jedesmal ist `Angestellter` Schlüssel. Hier ist die Fremdschlüsselbedingung sinnlos.

Definition

Sei $\mathcal{A} = \bigcup_{j=1}^{n} \mathcal{A}_i$. Es gibe in Γ genau eine Verbundabhängigkeit $\bowtie_{j=1}^{n} \mathcal{A}_i$ und auf jedem $\mathcal{A}_i$ eine funktionale Abhängigkeit $X_i \to \mathcal{A}_i$ mit

1. $(R_1, \ldots, R_n) \in S_1(\mathcal{A}_1, \Gamma_1) \times \cdots \times S_m(\mathcal{A}_m, \Gamma_m) \Longrightarrow \text{chase}_{\{\bowtie_{i=1}^{n} \mathcal{A}_i\}}$ erfüllt Γ.
2. Für alle X ist $\pi_X^t = \bigcup_{i=1}^{l} \pi_X(E_i)$, wobei jedes E_i ein Erweiterungsverbund ist.

Gelten diese Bedingungen, so heißen $S_1(\mathcal{A}_1, \Gamma_1), \ldots, S_m(\mathcal{A}_m, \Gamma_m)$ *unabhängig*.

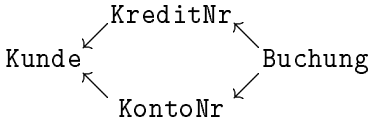
Definition

$(S_1, \ldots, S_m)$ erfüllt die *Eindeutigkeitsbedingung*, wenn folgendes gilt: Ist $A \in \mathcal{A}_i^+$, so gibt es genau ein $\mathcal{A}_j$ mit der Eigenschaft, daß $\mathcal{A}_i$ einen Schlüssel K von S_j enthält und $A \in \mathcal{A}_j \setminus K$ ist. S_j fügt dem $\mathcal{A}_j$ das Attribut A hinzu. Wichtig: Das Attribut kann nur auf eine Weise zu dem Schema hinzugefügt werden.

Satz

Falls die Eindeutigkeitsbedingung gilt, sind die Schemata unabhängig.

Beispiel



Ausweg

Rollenattribute: Kreditkunde, Sparkunde.

Sei $X \subseteq \mathcal{A}_1 \cup \mathcal{A}_2$ $(R_1 \bowtie R_2)$ Es existiere ein $\mathcal{A}_3 \subseteq \mathcal{A}_1^+$. Falls in $R_3 \in S_3(\mathcal{A}_3, \Gamma_3)$ keine Tupel sind, die mit einem Tupel aus $R_1 \bowtie R_2$ verbindbar wären, wird dieses Tupel durch einen weiteren Verbund mit R_3 vernichtet.

Bilde minimalen Erweiterungsverbund, das heißt minimal mit der Eigenschaft, daß die bereits vorhandenen Attribute X umfassen.

Um eine Anfrage zu X zu beantworten geht man nun wie folgt vor: Projiziere alle minimalen Erweiterungsverbunde, die zu X gebildet sind, auf X und gebe deren Vereinigung aus.

Algorithmus zur Berechnung von $\pi_X^t\left(\text{chase}_{\{\bowtie_{i=1}^{n} \mathcal{A}_i\}}\left(T_{\{\mathcal{A}_1,\ldots,\mathcal{A}_n\}}(R)\right)\right)$ (das heißt X-totale Projektion der Datenbank)

(1) Sei $V = \{\mathcal{A}_i \mid X \subseteq \mathcal{A}_i^+\}$

(2) Solange $\mathcal{A}_i, \mathcal{A}_j \in V$ existieren, so daß die Attributmenge zum minimalen Erweiterungsverbund von $\mathcal{A}_j$ bezüglich X das $\mathcal{A}_i$ enthält, entferne $\mathcal{A}_i$ aus V.

(3) Bilde zu jedem $\mathcal{A}_j \in V$ den minimalen Erweiterungsverbund E_j von $\mathcal{A}_j$ bezüglich X.

(4) Berechne $\bigcup_{\mathcal{A}_j \in V} \pi_X^t(E_j)$

Beispiel

Gegeben seien die Attributmengen

$$\mathcal{A}_1 = \{\texttt{Ort}, \texttt{Abteilung}, \texttt{Projekt}\}$$
$$\mathcal{A}_2 = \{\texttt{Abteilung}, \texttt{Projekt}, \texttt{Chef}\}$$
$$\mathcal{A}_3 = \{\texttt{Ort}, \texttt{Chef}, \texttt{Gruppenleiter}\}$$

mit den Abhängigkeiten

$$\{\texttt{Ort}, \texttt{Abteilung}\} \to \{\texttt{Projekt}\}$$
$$\{\texttt{Abteilung}, \texttt{Projekt}\} \to \{\texttt{Chef}\}$$
$$\{\texttt{Ort}, \texttt{Chef}\} \to \{\texttt{Gruppenleiter}\}.$$

Sei $X = \{\texttt{Ort}, \texttt{Chef}\}$. Gesucht ist π_X^t.

Der minimale Erweiterungverbund von $\mathcal{A}_1$, der X umfasst ist $\mathcal{A}_1 \bowtie \mathcal{A}_2$, der minimale Erweiterungsverbund von $\mathcal{A}_3$, der X umfasst ist $\mathcal{A}_3$ und $X \subseteq \mathcal{A}_3^+$. Bilde also $\pi_X^t(R_1 \bowtie R_2) \cup \pi_X^t(R_3)$. Dies ist das gesuchte Ergebnis.

A Literatur zur Vorlesung Informatik III

[1] T. Härder, *Implemtierung von Datenbanksystemen*, Hanser (1978), ISBN 3-446-12549-3.

[2] D. Maier, *The theory of relational databases*, Pitman (1983), ISBN 0-914894-42-0.

[3] K. Mehlhorn *Sorting and Searching*, Springer 1984.

[4] G. Schlageter/W. Stucky *Datenbanksysteme: Konzepte und Modelle*, Teubner Stuttgart (1983), ISBN 3-519-12339-8.

[5] J. Ullmann *Principles of Databases and Knowledge-Base Systems I,II*, Computer Science Press 1988/89.

[6] G.Vossen *Datenmodelle, Datenbanksprachen und Datenbankmanagementsysteme*, Addison-Wesley (1987), ISBN 3-925118-64-0.

[7] C.C. Yang *Relational databases*, Prentice-Hall, Englewood Cliffs (1986), ISBN 0-13-771858-6-025.

[8] R.Heuer *Objektorientierte Datenbanken.*