Zusammenfassung

Datenbanken

von Steve Göring
27.02.2010
Kontakt: stg7@gmx.de

Inhaltsverzeichnis

Kapitel 1: Datenbanken-Grundgedödel

Was ist eine Datenbank?

Daten= logisch gruppierte Informationseinheiten
Bank=Dienstleistungen für mehrere Kunden
Datenbank= Daten+Bank=langfristige Aufbewahrung von Daten

Datenredundanz

→ schwere Verwaltung der Datenbestände

→ ineffektive Speicherung / Verwaltung

→ Datenunabhängigkeit,Datenschutz,Datensicherheit ist nicht gewährleistet

→ → Probleme

Die Codd'schen Regeln

Integration; Operationen ;Katalog ; Benutzersichten ; Integritätssicherung

Datenschutz ; Transaktionen ; Synchronisationen ; Datensicherung

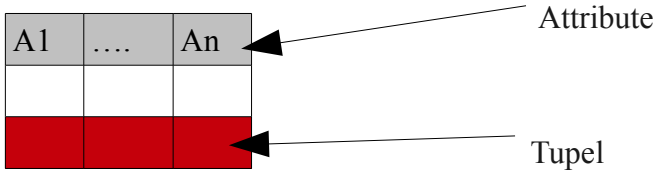
Kapitel 2: Relationale Datenbanken -Daten als Tabellen

Datenbank=Menge von Tabellen

Tabelle=Relation

Darstellung von Relationen/Begriffe

Relationenname



Integritätsbedingungen: Schlüssel

Attribute einer Spalte identifizieren das Tupel eindeutig=Schlüsseleigenschaft

auch Attributkombi möglich

Kennzeichnung: durch Unterstreichung

Fremdschlüssel=Verweis auf ein Tupel einer anderen Tabelle durch eindeutigen Schlüssel

Tabellen erzeugen

create table relationenname (Attribut Typ [not null],...);

primary key(Schlüssel) kennzeichnet Primärschlüssel

mögliche Typen: integer (int,integer4), smalint,float(genauigkeit),date, usw

foreign key(AttributnameName) references AndereTabelle(PrimärschlüsselDerTabelle)

Anfrageoperationen auf Tabellen (min. Relationenalgebra)

Kombinationen der Operationen sind möglich.

Selektion

Auswahl von Zeilen einer Relation anhand eines Selectionsprädikates

$\sigma_{prädikat}(Relation)$

Projektion

Auswahl von Spalten einer Relation durch Angabe der Attribute

$\pi_{Spalten}(Relation)$

→ entfernt doppelte Tupel (Mengenalgebra)

Natürlicher Verbund

Verknüpft Tabellen über gleichbenannte Attribute, „verschmelzen“ der beiden Tabellen

$Relation1 \bowtie Relation2$

Umbenennung

Atrributnamen können Umbenannt werden

$\beta_{neu \leftarrow alt}(Relation)$

Mengenoperationen

für alle Mengenoperationen: Attributmengen beider Relationen müssen gleich sein

Vereinigung

$Relation \cup Relation$

Differenz

$Relation1 - Relation2$

Durchschnitt

$Relation1 \cap Relation2$

SQL-Anfragen

Duplikate werden nicht automatisch entfernt, nur mit distinct

Selection

select *
from tabellenname
where Bedingungen;

Projektion

select Attribute
from tabellenname;

Verknüpfungen von Tabellen

crossjoin: select * from tab1,tab2
natural join: select * from tab1 natural join tab2
alternativ: select * from tab1,tab2 where tab1.attr=tab2.attr

Mengenoperationen

union, except (oder mit not in)

Änderungsoperationen

update tabellenname
set attribut1=wert,attribut2=wert2
[where ..];

insert into tabellenname
[(attribute)]
values (wert1,wert2,...) ;

delete from tabellenname [where..];

Kapitel 3: ER-Modelle

Datenbankmodelle

= System von Konzepten zur Beschreibung von Datenbanken , legt Syntax und Semantik von Datenbankbeschreibungen(=Datenbankschemata) für ein Datenbanksystem fest

legt fest: statische Eigenschaften(Objekte,Beziehungen, Datentypen)

dynamische Eigenschaften(Operationen, Beziehungen zwischen Ops)

Integritätsbedingung(an Objekten,Operationen)

Das ER-Modell

Entity: Objekt der Realen~ oder Vorstellungs~Welt

Relationship: beschreibt Beziehungen zwischen Entities

Attribut: repräsentiert Eigenschaft einer Beziehung/Entity

Werte= primitive Datenelemente die direkt darstellbar sind

Entities

=die in einer Datenbank zu repräsentierende Informationseinheit

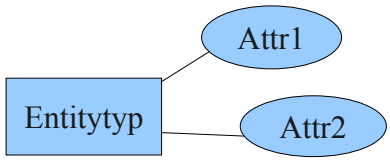
-im Gegensatz zu Werten nicht direkt darstellbar (durch Attribute beschrieben)

-eingeteilt in Entitytypen

Attribute

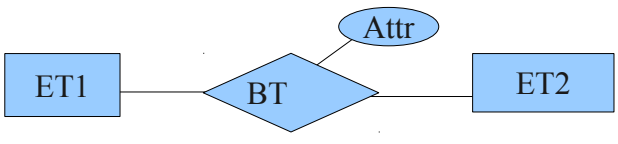
=modellieren Eigenschaften von Entities/Beziehungen, Unterstreichung = Primärschlüssel

werden für Entity Typen festgelegt



Beziehungstypen

Beziehungen zwischen Entitytypen werden durch Beziehungstypen festgelegt (haben eventuell auch Attribute)



Merkmale von Beziehungen

Stelligkeit (oder Grad)=Anzahl der beteiligten ET

Kardinalität(oder Funktionalität)=Anzahl der eingehenden Instanzen eines Ets (1:1,1:n,m:n)
=Integritätsbedingung

1:1 Beziehungen

jedem Entity e1 ist maximal ein Entity e2 zugeordnet und umgekehrt

(Mann ist verheiratet mit Frau (bei Monogamie))

1:N Beziehungen

jedem Entity e1 sind beliebig viele Entities e2 zugeordnet aber zu jedem e2 gibt es max 1 e1

(Mutter hat Kinder)

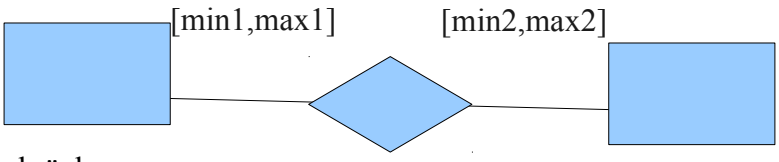
→ N:1 = inverse zu 1:n , auch funktionale Beziehung

M:N Beziehung

keine Einschränkungen

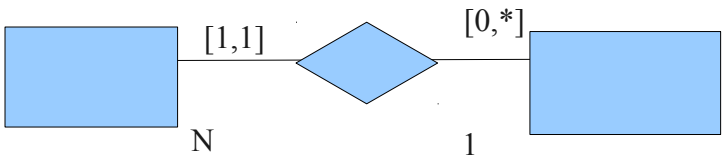
Min/Max Notation(Kardinalitätsangabe)

legt weitere Beschränkungen fest



[0,*]=keine Beschränkung

Alternative Kardinalitätsangabe



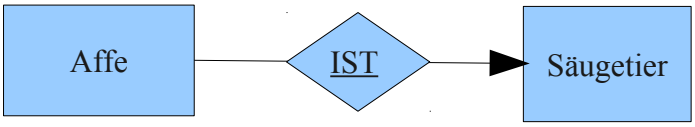
Abhängige Ets

bei N:1 Beziehung:

alternativ:

IST-Beziehung

=Spezialisierung/Generalisierung (wie bei OOP)



alternativ:

Kapitel 4: Datenbankentwurf

Entwurfsaufgabe

=Datenhaltung für mehrere Anwendungen und Jahre → besondere Bedeutung

Anforderung an Entwurf:

- Anwendungsdaten ableitbar
- nur benötigte Daten speichern
- nicht redundante Speicherung

Konzeptioneller Entwurf

1. formale Beschreibung des Problems

Sprachmittel: semantisches Datenmodell

Vorgehensweise: Modellierung von Sichten,Analyse der Sichten,Integration der Sichten in Gesamtschema

→ ergibt:konzeptionelles Gesamtschema (bspweise: ER-Diagramm)

Sichtenintegration

Analyse der Sichten auf Konflikte +Integration in Gesamtschema

Konflikte:

- Namenskonflike
- Typkonflikte=versch. Strukturen für gleiches Element
- Wertbereichskonflikte
- Bedingungskonflikte=versch. Schlüssel für ein Element
- Strukturkonflikte

Verteilungsentwurf

wenn Daten auf mehreren Rechnern verteilt gespeichert werden sollen

- horizontale Verteilung (1000 Datensätze dort, 3000 wo anders)
- vertikale Verteilung (Attr1...AttrN auf Rechner 1, AttrN.. AttrM auf Rechner 2, verbunden ü
  - Schlüssel)

ER-Abbildung auf Relationen

Entity-Typen + Beziehungen → Relationenschemata

Attribute → Attribute der Relationsschemata + Übernehmen der Schlüssel

Kardinalitäten → durch Wahl der Schlüssel eindeutig

in einigen Fällen → Verschmelzung der Relationenschemata von Entitytypen + Beziehungen

Abbildung von Beziehungstypen

neues Relationsschemata mit allen Attributen des Beziehungstypes

+Übernahme aller Primärschlüssel

m:n Beziehung → beide Primärschlüssel zusammen werden Primärschlüssel im Relationenschemata

1:n Beziehung → Primärschlüssel der n Seite wird neuer Primärschlüssel

1:1 Beziehung → beide Primärschlüssel werden je ein Primärschlüssel im neuen RS, PS wird dann ausgewählt (von den beiden)

Mögliche Verschmelzungen

1:n → die n Seite kann in das RS der Beziehung übernommen werden

1:1 → beide ET können in das RS der Beziehung übernommen werden

Übersicht über Transformation

ER-Konzept	wird abgebildet auf relationales Konzept
Entity-Typ E_i	Relationenschema R_i
Attribute von E_i	Attribute von R_i
Primärschlüssel P_i	Primärschlüssel P_i
Beziehungstyp	Relationenschema
	Attribute: P_1, P_2
dessen Attribute	weitere Attribute
$1:n$	P_2 wird Primärschlüssel der Beziehung
$1:1$	P_1 und P_2 werden Schlüssel der Beziehung
$m:n$	$P_1 \cup P_2$ wird Primärschlüssel der Beziehung
IST-Beziehung	R_1 erhält zusätzlichen Schlüssel P_2

E_1, E_2: an Beziehung beteiligte Entity-Typen,

P_1, P_2: deren Primärschlüssel,

$1:n$-Beziehung: E_2 ist n-Seite,

IST-Beziehung: E_1 ist speziellerer Entity-Typ

Kapitel 5: Relationale Entwurfstheorie

Begriffe

Begriff	Informale Bedeutung
Attribut	Spalte einer Tabelle
Wertebereich	mögliche Werte eines Attributs (auch Domäne)
Attributwert	Element eines Wertebereichs
Relationenschema	Menge von Attributen
Relation	Menge von Zeilen einer Tabelle
Tupel	Zeile einer Tabelle
Datenbankschema	Menge von Relationenschemata
Datenbank	Menge von Relationen (Basisrelationen)

Schlüssel	minimale Menge von Attributen, deren Werte ein Tupel einer Tabelle eindeutig identifizieren
Primärschlüssel	ein beim Datenbankentwurf ausgezeichneter Schlüssel
Fremdschlüssel	Attributmenge, die in einer anderen Relation Schlüssel ist
Fremdschlüsselbedingung	alle Attributwerte des Fremdschlüssels tauchen in der anderen Relation als Werte des Schlüssels auf

Integritätsbedingung

Primattribut=Element eines Schlüssels

Oberschlüssel (Superkey)=jede Obermenge eines Schlüssels

Fremdschlüssel= X(R1) → Y(R2)

Redundanzen

unerwünscht da:

unnötiger Speicherplatz wird belegt

Änderungsoperationen auf Basisrelationen schwer umzusetzen

Funktionale Abhängigkeit (FD)

Wenn in jedem Tupel der Relation der Attributwert unter den X-Komponenten den Attributwert unter den Y-Komponenten festlegt

kurz: X → Y

Schlüssel sind Spezialfall

Ableitungsregeln

F1: Reflexivität $X \supseteq Y \Rightarrow X \rightarrow Y$

F2: Augmentation $\{ \; X \rightarrow Y \; \} \Rightarrow XZ \rightarrow YZ$ sowie $XZ \rightarrow Y$

F3: Transitivität $\{ \; X \rightarrow Y \; , \; Y \rightarrow Z \; \} \Rightarrow X \rightarrow Z$

F4: Dekomposition $\{ \; X \rightarrow YZ \; \} \Rightarrow X \rightarrow Y$

F5: Vereinigung $\{ \; X \rightarrow Y \; , \; X \rightarrow Z \; \} \Rightarrow X \rightarrow YZ$

F6: Pseudotransitivität $\{ \; X \rightarrow Y \; , \; WY \rightarrow Z \; \} \Rightarrow WX \rightarrow Z$

F1-F3 = Armstrong Axiome

<u>Alternative Regelmege:</u>

B-Axiome oder RAP Regeln:

Reflexivität $\{\} \Rightarrow X \rightarrow X$

Akkumulation $\{ \; X \rightarrow ZY \; , \; Z \rightarrow AW \; \} \Rightarrow X \rightarrow YZA$

Projektivität $\{ \; X \rightarrow YZ \; \} \Rightarrow X \rightarrow Y$

Normalformen

legen Eigenschaften von Relationenschemata fest

verbieten bestimmt Kombi von Fds in Relationen

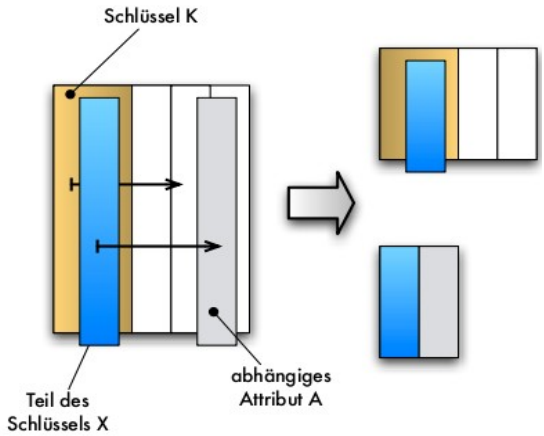
sollen Redundanzen und Anomalien vermeiden

Erste NF

erlaubt nur atomare Attribute in den Relationenschemata (Attribute sind Elemente von Standartdatentypen)

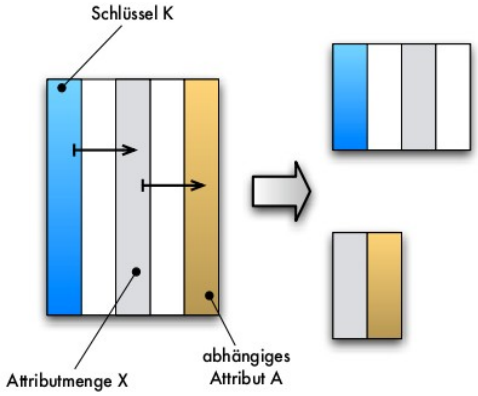
Zweite NF

Eliminierung von partiellen Abhängigkeiten bei Nichtschlüsselattributen



Dritte NF

Eliminierung von transitiven Abhängigkeiten (nur Nichtschlüssel als Endpunkte)



Boyce Codd NF

Eliminierung transitiver Abhängigkeiten auch zwischen Primattributen

Schemaeigenschaften

Kennung	Schemaeigenschaft	Kurzcharakteristik
	1NF	nur atomare Attribute
	2NF	keine partielle Abhängigkeit eines Nicht-Primattributes von einem Schlüssel
S 1	3NF	keine transitive Abhängigkeit eines Nicht-Primattributes von einem Schlüssel
	BCNF	keine transitive Abhängigkeit eines Attributes von einem Schlüssel
S 2	Minimalität	minimale Anzahl von Relationenschemata, die die anderen Eigenschaften erfüllt

Abhängigkeitstreue

→ alle vorhergehenden Abhängigkeiten können „rekonstruiert“ werden

Verbundtreue

→ alter Verbund kann „rekonstruiert“ werden

Kennung	Transformationseigenschaft	Kurzcharakteristik
T1	Abhängigkeitstreue	alle gegebenen Abhängigkeiten sind durch Schlüssel repräsentiert
T2	Verbundtreue	Originalrelationen können durch den Verbund der Basisrelationen wiedergewonnen werden

Kapitel 6: Die relationale Anfragesprache SQL

SFW-Block

select [distinct] projektionsliste
from relationenliste
[where bedingungen]

projektionsliste bei mehreren gleich benannten Attributen Zugriff über Präfix (Tabellenname.Attr)

Kartesisches Produkt

select * from tab1,tab2
oder
select * from tab1 cross join tab2

Tupelvariablen für Mehrfachzugriff

select * from tab1 Name1, tab1 Name2

Natürlicher Verbund

select * from tab1, tab2, where tab1.Attr=tab2.Attr
oder
select * from tab1 natural join tab2
oder
select * from tab1 join tab2 on tab1.attr=tab2.attr
oder
select * from tab1 join tab2 using(attr)

„Unterabfragen“

select * from (select * from tab1) as Erg;

Where Klausel

Bedingungen können die Form haben

- Vergleich mit Konstante/Attribute (>=, <= , >,<,<>)
- logische Verknüpfungen von Vergleichen (and , or , not)

Like = Ungewissheitsselektion (bei Zeichenketten)

- % Platzhalter (mehrere Zeichen), _ (ein Zeichen)

Mengenoperationen

select * from tab1

OP

select * from tab2

wobei Attribute von tab1 und tab2 identisch vom Typ und Anzahl

OP= union, intersect , except

Schachtelung

→ Operatoren all,any, in, not in

Skalare Ausdrücke

+,-,*,/ auf numerische Typen

char_length, substring bei Strings

now(), current_date,current_time +,-,* bei Datumstypen/Zeit~

Typkonvertierung : cast(Attribut as Typ)

Aggregatfunktionen

count, sum, avg, max,min

eventuell mit distinct „verbunden“ select count(distinct Attr) from tab;

→ bei Gruppierungen beziehen sich die Aggregatfnkt immer auf innere „Gruppe“

Rekursive Anfragen

SQL2003

```
  with recursive TOUR(Abfahrt, Ankunft) as (
--- Basisfall
   select Abfahrt, Ankunft
   from BUSLINIE
   where Abfahrt = ’Nuriootpa’
      union all
--- Rekursionsschritt
   select T.Abfahrt, B.Ankunft
   from TOUR T, BUSLINIE B
   where T.Ankunft = B.Abfahrt
)
select distinct * from TOUR --- Ausgabe
```

Kapitel 7: Algebra&Kalkül

Begriffe

Anfrage= Folge von Operationen die aus Basisrelationen Ergebnisrelation liefert

Sicht= Folge von Operationen die als Name abgespeichert später als „Relation“ verwendet werden kann

Snapshot=Ergebnisrelation einer Anfrage die nie ein zweites mal berechnet wird

Kriterien für Anfragesprachen

Ad-Hoc Formulierung: Anfrage formulierbar ohne vollständiges Programm zu schreiben

Deskriptivität: direkte Formulierung der Anfragen

Mengenorientiertheit

Abgeschlossenheit: Ergebnis ist wieder Relation und kann weiterverwendet werden

Adäquatheit: alle Konstrukte des zu Grunde liegenden Datenmodells werden unterstüzt

Orthogonalität: Sprachkonstrukte in ähnlichen Situationen ähnlich anwendbar

Effizienz: jede Operation effizient ausführbar

Sicherheit

Eingeschränktheit: Anfragesprache darf keine komplette Programmiersprache sein

Vollständigkeit

Relationenalgebra

Projektion (entfernt auch Doppelte Einträge)

Selektion (Kommutativ, Distributiv bezüglich Vereinigung,Durchschnitt,Differenz)

Join (Kreuzprodukt durch Umbenennung)

Vereinigung

Differenz

Umbenennung

Erweiterungen

Verbundvarianten(equi-join,theta-join,semi-join,outer-join)

Äußere Verbunde:

- voller äußerer Verbund: übernimmt alle Tupel beider Operanden
- linker äußerer Verbund: übernimmt alle Tupel des linken Operanden
- rechter äußerer Verbund: ~rechten ~

Division:

$r_1(R_1)$ und $r_2(R_2)$ gegeben mit $R_2 \subseteq R_1$, $R' = R_1 - R_2$. Dann ist

$$\begin{aligned} r'(R') &= \{t \mid \forall t_2 \in r_2 \exists t_1 \in r_1 : t_1(R') = t \wedge t_1(R_2) = t_2\} \\ &= r_1 \div r_2 \end{aligned}$$

Division von r_1 durch r_2

$$r_1 \div r_2 = \pi_{R'}(r_1) - \pi_{R'}((\pi_{R'}(r_1) \bowtie r_2) - r_1)$$

Gruppierung

Relationale Kalküle

Bereichskalkül: Variablen nehmen Werte elementarer Datentypen an

Tupelkalkül: Variablen variieren über Tupelwerte (Zeilen)

Tupelkalkül

Bsp: $\{ \quad x \quad | \quad x \in TEST \wedge x.farbe = 'rot' \quad \}$

Bereichskalkül

Bsp: $\{ \quad x \quad | \quad TEST(x, y, z) \wedge z = 'Bier' \quad \}$

→ streng relational vollsändig

Umsetzung:

Geg.: Relationenschemata $R(A_1, \ldots, A_n)$ und $S(B_1, \ldots, B_m)$

- Vereinigung (für $n = m$)

$$R \cup S \hat{=} \{x_1 \ldots x_n \mid R(x_1, \ldots, x_n) \vee S(x_1, \ldots, x_n)\}$$

- Differenz (für $n = m$)

$$R - S \hat{=} \{x_1 \ldots x_n \mid R(x_1, \ldots, x_n) \wedge \neg S(x_1, \ldots, x_n)\}$$

- Natürlicher Verbund

$$R \bowtie S \hat{=} \{x_1 \ldots x_n x_{n+1} \ldots x_{n+m-i} \mid R(x_1, \ldots, x_n) \wedge S(x_1, \ldots, x_i, x_{n+1}, \ldots, x_{n+m-i})\}$$

Annahme: die ersten i Attribute von R und S sind die Verbundattribute, also $A_j = B_j$ für $j = 1 \ldots i$

Kapitel 8: Weitere Datenbanksprachen

Anbindung der Datenbank an die Programmiersprache über bekanntes Cursorprinzip

QBE

→ ganz am Anfang : QBE stinkt

Kern ist relational vollständig

P.= Ausgabe sonstige Variablen groß geschrieben

Tabelle hat Aufbau:

Relationenname	| Spalte1	|...|SpalteN
	| Bed1	|...|BedN

Algebra	**QBE**
Projektion	mit `P.` markierte Spalten
Selektion	1. Vergleiche als Spalteneinträge 2. `Condition Box`
Umbenennung	explizite Ausgabetabelle
Verbund	Verbindung zweier Tabellen mittels Beispielelementen (Bereichsvariablen)

QBE gibt's in ACESS → stinkt noch mehr

Kapitel 9: Transaktion, Integrität und Trigger

Integrität

~Bedingungen=Bedingungen für Zuverlässigkeit/Korrektheit

Klassifikation

Bedingungsklasse		zeitlicher Kontext
statisch		Datenbankzustand
dynamisch	transitional	Zustandsübergang
	temporal	Zustandsfolge

Inhärente Integritätsbedingungen

Typintegrität

Schlüsselintegrität= Angabe eines Schlüssels einer Relation

Referntielle Integrität=Angabe von Fremdschlüssel

Transaktionen

= Folge von Operationen(Aktionen), die die DB von einem konsistenten Zustand in einen konsistenten (eventuell veränderten) Zustand überführt, wobei das ACID Prinzip eingehalten werden muss

ACID Eigenschaften

Atomicity: Transaktion wird ganz oder gar nicht ausgeführt

Consistency: vor / nach Transaktion DB in konsistentem Zustand

Isolation: Nutzer soll Eindruck haben → er allein Arbeitet mit DB

Durability (Dauerhaftigkeit): Ergebnis soll nach erfolgreicher Transaktion dauerhaft in DB gespeichert sein

Kommandos

BOT = Begin of Transaktion (in SQL implizit)

commit = Transaktion soll erfolgreich beendet werden

abort= Transaktion soll abgebrochen werden

Assertion Klausel

Prädikat, dass eine Bedingung ausdrückt, die immer von der DB erfüllt sein soll

create assertion NAME check (Prädikat)

Trigger

= Anweisung / Prozedur die beim eintreten eines bestimmten Ereignis automatisch vom DBMS ausgeführt wird

hier nur SQL2003er Syntax:

- Syntax:

```
create trigger <Name: >
after | before <Ereignis>
on <Relation>
[ when <Bedingung> ]
begin atomic < SQL-Anweisungen > end
```

- Ereignis:
  - **insert**
  - **update** [**of** <Liste von Attributen>]
  - **delete**

- *Kein Kundenkonto darf unter 0 absinken:*

```
create trigger bad_account
after update of Kto on KUNDE
referencing new as INSERTED
when (exists
   (select * from INSERTED where Kto < 0)
)
begin atomic
   rollback;
end
```

⇝ ähnlicher Trigger für **insert**

Kapitel 10: Sichten und Zugriffskontrollen

Sichten

=virtuelle Relation

Vorteile

vereinfachen von Anfragen für bestimmte Benutzer, da komplexere Abfragen als „Sicht“ verkauft werden können

Beschränkung von Zugriffen auf die DB

Probleme

automatische Anfragetransformation

Änderungen auf Sichten

Definition in SQL:

create view NAME [SchemaDeklaration]

as SQLAnfrage

[with check option]

Kriterien für Änderungen auf Sichten

- **Effektkonformität**
  Benutzer sieht Effekt als wäre die Änderung auf der Sichtrelation direkt ausgeführt worden
- **Minimalität**
  Basisdatenbank sollte nur minimal geändert werden, um den erwähnten Effekt zu erhalten
- **Konsistenzerhaltung**
  Änderung einer Sicht darf zu keinen Integritätsverletzungen der Basisdatenbank führen
- **Respektierung des Datenschutzes**
  Wird die Sicht aus Datenschutzgründen eingeführt, darf der bewußt ausgeblendete Teil der Basisdatenbank von Änderungen der Sicht nicht betroffen werden

Klassifikation der Problembereiche

1. Verletzung der Schemadefinition (z.B. Einfügen von Nullwerten bei Projektionssichten)
2. Datenschutz: Seiteneffekte auf nicht-sichtbaren Teil der Datenbank vermeiden (Tupelmigration, Selektionssichten)
3. nicht immer eindeutige Transformation: Auswahlproblem
4. Aggregierungssichten (u.a.): keine sinnvolle Transformation möglich
5. elementare Sichtänderung soll genau einer atomaren Änderung auf Basisrelation entsprechen: 1:1-Beziehung zwischen Sichttupeln und Tupeln der Basisrelation (kein Herausprojizieren von Schlüsseln)

In SQL

Integritätsverletztende Sichten nicht erlaubt

Sichten mit nicht eindeutiger Transformation

Einschränkungen bei Sichtänderungen:

- änderbar nur Selektions- und Projektionssichten (Verbund und Mengenoperationen nicht erlaubt)
- 1:1-Zuordnung von Sichttupeln zu Basistupeln: kein **`distinct`** in Projektionssichten
- Arithmetik und Aggregatfunktionen im **`select`**-Teil sind verboten
- genau eine Referenz auf einen Relationsnamen im **`from`**-Teil erlaubt (auch kein Selbstverbund)
- keine Unteranfragen mit „Selbstbezug“ im **`where`**-Teil erlaubt (Relationsname im obersten SFW-Block nicht in **`from`**-Teilen von Unteranfragen verwenden)
- **`group by`** und **`having`** verboten

Kapitel 11: Anwendungsprogrammierung

→ denke eher unwichtig