Übung Datenbanksysteme

Updates, Integritätsbedingungen, funktionale Abhängigkeiten

12.1.2004

Änderungsoperationen bei SQL (Daten)

- Einfügen neuer Tupel (schon bekannt)

  INSERT INTO Table (Spalte1, Spalte2)
  VALUES (Wert1, Wert2)

  – Abkürzung, falls **alle** Werte in **Schemareihenfolge** verwendet werden

  INSERT INTO Table VALUES (Wert1, Wert2)

  – Input aus einer Query (neu)

  INSERT INTO Table
  Select * FROM Othertable where ...

  Können auch mehrere Werte sein

Änderungsoperationen in SQL (2)

- Löschen:
  DELETE FROM TABLE
  (WHERE Bedingung)

  WHERE-Klausel ähnlich zu where-Klausel im SELECT, so z.B. auch Subqueries möglich

- Modifikation bestehender Tupel
  UPDATE Table SET Spalte = Wert
  (Where Bedingung)

Wichtig:
- Alle Änderungsoptionen arbeiten auf Mengen
- Alle Änderungsoperationen arbeiten in zwei Phasen – warum ?

Integritätsbedindungen

Problemstellung:

DBMS soll Gültigkeit der Daten bewahren

Welche Mittel:

- Eindeutigkeit in einer Tabelle
  - ⇒ *Primärschlüssel*
  - ⇒ *Schon bekannt*
- Schlüsselbeziehungen
  - =>Fremdschlüssel
- Wertbereiche von Attributen

Wir betrachten deklarative Integritätsbedingungen
⇔ Trigger (explizite Prozeduren bei bestimmten Ereignissen)

Was sind jeweils Vor- und Nachteile ?

Fremdschlüssel

Definition Fremdschlüssel: Verweis auf Einträge in anderer Tabelle
Problem: referenzierte Einträge müssen existieren
(sonst “Referenz ins Nichts”)
Vorgehen zur Lösung:

- „Markierung“ der Foreign Keys
- DBMS kümmert sich um den Rest

Syntax:

in der Tabellendefinition (auch nachträglich)

... , Spalte REFERENCES table, ...

Keine Spaltenangabe in der Zieltabelle nötig (Wieso ?)

⇒ Fremdschlüssel bezieht sich immer auf Primärschlüssel

Auswirkung:

- INSERT funktioniert nur noch, wenn referenzierter Wert existiert
- Was passiert bei DELETE und UPDATE ?

Fremdschlüsseloptionen bei Update und Delete

- Ergänzung der Deklaration mit
  - ON UPDATE Aktion
  - ON DELETE Aktion

Standardaktion
(wenn kein ON UPDATE/DELETE angegeben):

- NO ACTION

  Verbiete alle Änderung und Löschvorgänge,
  wenn es abhängige Tupel gibt

Andere Aktionen:

- SET NULL

  Setze den Fremdschlüssel der Abhängigen zu NULL

- CASCADE

  Propagiere die Wertänderung oder
  Lösche die abhängigen Tupel

Regeln zu SET NULL und CASCADE

Wann verwendet man welche Regel ?

NO ACTION ist ein guter Default
Überlegen, wann etwas anderes Sinn macht !

- ON DELETE CASCADE
  - Bei schwachen Entitäten
  - Bei n:m-Beziehungen

Vorsicht bei DELETE CASCADE – Lawineneffekt beim Löschen

- ON DELETE SET NULL:
  - Bei normalen 1-N-Beziehungen

Bei Updates sehr häufig applikationsspezifisch

Attributwerte

Problemstellung:

- Einträge einer Spalte sollen aus bestimmten Wertebereich kommen:
  Beispiel: Noten an der TUM
- Basisdatentyp erlaubt nicht genügend Einschränkungen z.B. TinyInt
- In Standard-SQL gibt es noch keine benutzerdefinierten Datentypen

Lösung:

- Explizite Festlegung der erlaubten Werte(menge)

Syntax (in Tabellendefinition):

Spalte CHECK (Bedingung)

- Typische Bedingungen:
  - Spalte BETWEEN Wert1 and Wert2
  - Spalte IN (Wert1, Wert2, Wert3)

Beispiel: ..., NOTE SMALLINT CHECK (NOTE IN (1.0, 1.3, 1.7, 2,0)), ...

- Auch komplexere Bedingungen möglich
- CHECK kann noch mehr ...

Aufgabe 1.1

Tabelle	Attribute	Parent
Kunde	Bezeichnung	Kategorie
Festgeldkonto	Kontonr	Konto
Girokonto	Kontonr	Konto
Kontobewegung	Kontonr	Konto
Kontobewegung	Vorname, Name, Gebdat	Kunde

Aufgabe 1.1 (2)

Tabelle	Attribute	Parent
Kontoauszug	Kontonr	Konto
Besitzt	Kontonr	Konto
Besitzt	Vorname, Name, Gebdat	Kunde
Listet	Erstelldat,Konto nr	Kontoauszug
Listet	Kontonr, Datum	Kontobewegung
Listet	Vorname, Name, Gebdat	Kunde

Aufgabe 1.3

- CREATE TABLE mytable.Kunde (VN VARCHAR(30) NOT NULL, NN VARCHAR(30) NOT NULL, Gebdat DATE NOT NULL,

  Bezeichnung VARCHAR(15) REFERENCES mytable.Kategorie ON DELETE SET NULL);

Aufgabe 1.4

- CREATE TABLE Kontobewegung (Datum DATE NOT NULL, VN VARCHAR(30) NOT NULL, NN VARCHAR(30) NOT NULL, Gebdat DATE NOT NULL, Kontonr INTEGER NOT NULL, Betrag NUMERIC(10,2), Bewegungsart VARCHAR(15),

  PRIMARY KEY (Datum, VN, NN, Gebdat, Kontonr), CONSTRAINT kto FOREIGN KEY (Kontonr) REFERENCES Konto ON DELETE CASCADE ON UPDATE RESTRICT,

  CONSTRAINT kd FOREIGN KEY (VN, NN, Gebdat) REFERENCES Kunde ON DELETE CASCADE ON UPDATE RESTRICT, CONSTRAINT Bewegung_korrekt CHECK (Bewegungsart in ('ein', 'aus')))

Aufgabe 1.5

- Änderungen Kategoriebezeichnung:

ON UPDATE CASCADE

-> Kunden behalten auf jedenfall Kategorie

Ansonsten geht auch ON UPDATE NO ACTION

- Änderungen Kundenidentifikation:

ON UPDATE CASCADE (sonst ist Zugriff auf Konten, bzw. Kontobewegungen nicht mehr möglich)

Aufgabe 1.6

- ALTER TABLE Kontobewegung ADD CONSTRAINT Bewegung_korrekt CHECK (Bewegungsart in (´ein´, ´aus´))

Aufgabe 2

Aufgabe 2.1

UPDATE Konto SET Gebühr = 0 WHERE Kontonr in (SELECT Kontonr FROM Girokonto where telebanking = ´t´);

Aufgabe 2.2

DELETE FROM Konto

WHERE Kontonr = 174266

Wegen ON DELETE CASCADE werden Löschungen werden propagiert (auch in Kontobewegung und listet).

Sinnvollerweise muß auch Betrag=0 gelten (das erfordert u.U. weitere Arbeit).

Aufgabe 2.3

- UPDATE Kunde SET Name = ´Mueller´
  WHERE Vorname = ´Erwin´ AND
  Name = ´Müller´ and Gebdat = 1935-05-17
- Änderungen auch in besitzt, listet und Kontobewegung.

Aufgabe 2.3 (2)

- On update cascade: die entsprechenden Tupel in besitzt, listet und Kontobewegung werden ebenfalls geändert
- On update restrict: verweist die Relation (z.B. besitzt) nicht mehr auf **exakt** das selbe Tupel wie vorher, wird die Update-Relation zurückgewiesen.
- On update no action:verweist die Relation (z.B. besitzt) weiterhin auf **irgend ein** Tupel in Kunde, wird die Änderung erlaubt, sonst zurückgewiesen.

Funktionale Abhängigkeiten

- Ab jetzt wieder recht theoretisch
- Teil der Normalformentheorie (mehr dazu im nächsten Übungsblatt)
- Formal:

  $\alpha \rightarrow \beta$ α, β Mengen von Spalten
- Anschaulich:

  Wenn die Werte von α gleich sind,
  dann müssen auch die Werte von β gleich sein
- Wie findet man fkt. Abhängigkeiten:
  - Verständnis der Daten
  - systematisches Suchen
- Wichtig:
- Funktionale Abhängigkeiten sind nicht abhängig von den konkret vorhandenen Tupeln

Beispiel für funktionale Abhängigkeiten

Nach gleichen Werten suchen !

A	B	C	D
A1	B1	C1	D1
A1	B1	C2	D2
A2	B2	C4	D3
A3	B2	C4	D3

Im Beispiel:
A -> B
BC->D
BD->C
…

Definition 1. bis 3. NF

- Definition 1 NF

„Alle Attribute haben atomare Werte“

- Im relationalen Modell immer erfüllt
- Definition 2 NF:
  - 1 NF
  - Nicht-Schlüsselattribute sind
    voll funktional abhängig
    von allen Kandidatenschlüsseln
- Voll funktional abhängig:

  Funktionale Abhängigkeit kann nicht weiter verkleinert werden (durch Herausnahme von Attr.)

Definition 3. Normalform

- R ist in 3NF, wenn
  - R in 2 NF
  - Keine transitiven fkt. Abhängigkeiten in R

Aufgabe 3:

- Relation Auftrag
  (ProduktNr, ProduktName,
  KundenNr, KundenName,
  Datum, Stückpreis, Anzahl,
  Nettopreis, Mehrwertsteuersatz, Bruttopreis)
- Suche nach fkt. Abh:
- Vorgehen nach „common sense“
- Durch Armstrongregeln können redundante Abhängigkeiten eliminiert werden (nächste Woche)

Afg 3: „Gefundene“ Fkt. Abh.

- {Stückpreis, Anzahl} → {Nettopreis}
- {Nettopreis, Mehrwertsteuersatz} → {Bruttopreis}
  entspricht Erfahrungswerten der Realität
- {KundenNr} → {KundenName}
- {KundenName} → {KundenNr}
  Namen und Nummern von Kunden sind gegenseitig abhängig
- {ProduktName} → {ProduktNr}
- {ProduktNr} → {ProduktName, Stückpreis, Mehrwertsteuersatz}
  Ein Produkt hat Preis pro Stück und Mehrwertsteuersatz
- {ProduktNr, KundenNr, Datum} → {Anzahl}
  Pro Datum gibt es genau eine Bestellung

Schlüssel und fkt. Abhängigkeiten

Funktionale Abhängigkeit unterstützen Suche nach Schlüssel – warum ?

- Schlüssel: Alle Werte eines Tupel sind durch diesen Wert eindeutig bestimmt
- Fkt Abh: Eine Menge Spalten bestimmt eine andere eindeutig

Genauer Zusammenhang: nächste Übung

- Für jetzt: Aus den fkt. Abhängigkeiten Schlüssel zusammensuchen

Afg 3b)

- {ProduktNr, KundenNr, Datum}
- {ProduktName, KundenName, Datum}

Aufgabe 3:

- Normalform der Relation
- 2 NF: nein, da bei 1. SK z.B. Kundenname nicht voll funktional abhängig (nur von KundenNR)
- Weitere NF: nein
- Kanonische Überdeckung:

=> Erhält alle fkt. Abhängigkeiten