Datenbank-Systeme

Transaktionen und Normalisierung

Internationaler Frauenstudiengang Informatik

Renzo Kottmann

Wiederholung

Lesen und verstehn von SQL code des Music Collection Projekts

Transaktionen

Eine Transkation ist eine Menge von Operationen die atomic, consistent, isoliert und dauerhaft sind (ACID).

ACID

- **A**lle Änderungen mit
- **C**hirurgischer Integrität in einem von allen anderen
- **I**soliertem Vorgang zusammengefasst und mit
- **D**auerhaftem Ergebnis.

Die wichtigsten Eigenschaften im Einzelnen:

- Atomic

Eine Transaktion gruppiert mehrere Anweisungen in eine einzige atomare Operation in der "Alles oder Nichts” stattfindet.

Die wichtigsten Eigenschaften im Einzelnen:

- Atomic

Eine Transaktion gruppiert mehrere Anweisungen in eine einzige atomare Operation in der "Alles oder Nichts” stattfindet.

- Consistent

Eine Transaktion bringt eine Datenbank von einem konsistenten Zustand in den nächsten.

Die wichtigsten Eigenschaften im Einzelnen:

- Atomic

Eine Transaktion gruppiert mehrere Anweisungen in eine einzige atomare Operation in der "Alles oder Nichts” stattfindet.

- Consistent

Eine Transaktion bringt eine Datenbank von einem konsistenten Zustand in den nächsten.

- Isolated

Transaktion ist unabhängig, d.h. sie wird nicht durch konkurrierende Transaktionen beeinflusst.

Die wichtigsten Eigenschaften im Einzelnen:

- Atomic

Eine Transaktion gruppiert mehrere Anweisungen in eine einzige atomare Operation in der "Alles oder Nichts” stattfindet.

- Consistent

Eine Transaktion bringt eine Datenbank von einem konsistenten Zustand in den nächsten.

- Isolated

Transaktion ist unabhängig, d.h. sie wird nicht durch konkurrierende Transaktionen beeinflusst.

- Durable

Bei der erfolgreichen Beendigung einer Transaktion sind alle Änderungen dauerhaft gespeichert.

Konsistenz bei sehr vielen gleichzeitigen Änderungen

- Transaktionen garantieren, dass trotz gleichzeitiger Änderungen von vielen verschiedenen Benuztern, alle Änderungen - jede für sich - konsistent sind.
- Tatsächlich ist jede einzelne SQL-Anweisung bei den meisten Datenbankensystemen implizit eine einzelne Transaktion, d.h. eine Gruppe von Anweisungen mit nur einer Anweisung.

Transaktion mit mehreren Anweisungen

PostgreSQL Dokumentation für syntaktische Details und fortgeschrittene Eigenschaften

Ausgangslage: Datenbank ist in einem super gutem Zustand: Z1

```
-- Transaktion wird angefangen:
BEGIN;
SQL Anweisung 1;
SQL Anweisung 2;
...
SQL Anweisung N;
-- Bleibt offen bis:
COMMIT;
```

Wenn die Transaktion **gut läuft**, d.h. alle Anweisungen korrekt sind:

- Datenbank ist in einem super gutem Zustand: Z2

Wenn Transaktion **fehlschlägt**, d.h. wenn nur eine der Anweisungen zum Fehler führt:

Datenbank wird automatisch in den vorherigen Zustand zurück gesetzt (rollback): Z1

Gewollter Transaktionsabbruch

```
-- Transaktion wird angefangen:
BEGIN;
SQL Anweisung 1;
SQL Anweisung 2;
...
SQL Anweisung N;
-- Bleibt offen bis:
ROLLBACK;
```

Wenn Transaktion **fehlschlägt** oder **erfolgreich** ist:

Datenbank wird in den vorherigen Zustand zurück gesetzt (rollback): Z1

DDL Entwicklung

- PostgreSQL ist einer der wenigen DBMS, die auch DDL Anweisungen in Transaktionen ausfuhren kann

```
-- Transaktion wird angefangen:
BEGIN;
CREATE TABLE test (id integer PRIMARY KEY);
INSERT INTO test VALUES (1),(2),(3);
SELECT * from test;
UPDATE test SET id = id + 3;
SELECT * from test;
ROLLBACK;
```

- Sehr nützlich für iterative, evolutionäre Datenbankentwicklung

Normalisierung

Fiktive Teilnehmerinnen Tabelle

Name	Nachname	MatrikelNummer (Primary Key)	emails	Klausur	Aufgabe	Strasse	Ort	PLZ
Ricarda	Huch	123456	huch@stud.hs-bremen.de, ric@gmail.com	FALSE	Buchhandlung	S 1	HB	23
Greta	Garbo	234567	great@web.de, garbo@hollywood.com	FALSE	Filmsammlung	S 1	HH	44
Emma	Goldman	345678	libertat@gmx.de	FALSE	Buchhandlung	S 3	HB	44

- Diese Tabelle ist nicht **Normal**, voll von **Anomalien**.

Update-Anomalien

1. Modifikation: Die Datenbank muss an mehreren Stellen geändert werden, um Einträge zu ändern. Beispiel: wenn der Name der Aufgabe von Buchhandlung in Buchhandlungen geändert werden soll.

2. Einfügung: Um einen Eintrag in eine Tabelle vorzunehmen, müssen nicht reale Werte erfunden werden. Beispiel: Um eine neue Aufgabe einzufügen, muss eine `MatrikelNummer` erfunden werden, da MatrikelNummer Primärschlüssel ist.

3. Löschung: Um einen Eintrag zu löschen, müssen auch ungewollt andere Einträge gelöscht werden: Beispiel: Um die Aufgabe 'Filmsammlung' zu löschen, muss `MatrikelNummer` (und somit die gesamte Zeile) gelöscht werden.

Entwurfs-Ziel

Entwerfe ein relationales Datenbank-Model derart, dass keine Update-Anomalien vorkommen können.

Funktionale Abhängigkeit

- Begriff gehört zur Normalisierung

Definition: Funktionale Abhängigkeit

Seien X und Y beliebige Teilmengen der Attribute einer Tabelle T. Dann ist Y von X funktional abhängig, wenn gilt

> für jede Wertekombination von X gibt es – zu einem gegebenen Zeitpunkt – nur eine Wertekombination von Y.

Definition: Funktionale Abhängigkeit

Seien X und Y beliebige Teilmengen der Attribute einer Tabelle T. Dann ist Y von X funktional abhängig, wenn gilt

> für jede Wertekombination von X gibt es – zu einem gegebenen Zeitpunkt – nur eine Wertekombination von Y.

Wenn die Werte von X feststehen, stehen auch die Werte von Y fest.

Auch wenn Y noch nicht bekannt ist, kann es nur eine Wertekombination von Y zu X geben.

Anders: Es kann zu keinem Zeitpunkt zweimal dieselben Werte für X geben, aber jeweils verschiedene Werte für Y.

Das heißt:

> X determiniert Y funktional $X \rightarrow Y$

Beispiele

Sei A die Menge aller Attribute der Relation Teilnehmerinnen,

mit:

1. X = {MatrikelNummer},
   dann gilt X → A denn X ist Primärschlüssel

2. Y = {Name},
   dann **gilt nicht** Y → A. Denn es kann mehrere Teilnehmerinnen mit demselben Namen geben.

3. O = {Ort, Strasse}, P = {PLZ},
   dann kann O → P gelten, muss aber nicht

Funktionale Abhängigkeiten sind Kontext-Abhängig

Beispiel:

1. O = {Ort, Strasse}, P = {PLZ},
   dann kann O → P gelten, muss aber nicht

- Gilt evtl. nur in Bremen bzw. bestimmten Regionen
- Aber nicht allgemein in Deutschland
  (s. Unterstein und Matthiessen S. 241)

1. Normalform

- Eine Datenbank ist in 1. Normalform (1NF), wenn alle Attribute atomar sind.
  D.h. jedes Attribut enthält nur einen Wert der immer als unzerteilbare Einheit betrachtet wird.

1. Normalform(isierung)

Name	Nachname	MatrikelNummer (Primary Key)	emails	Klausur	Aufgabe	Strasse	Ort	PLZ
Ricarda	Huch	123456	huch@stud.hs-bremen.de, ric@gmail.com	FALSE	Buchhandlung	S 1	HB	23
Greta	Garbo	234567	great@web.de, garbo@hollywood.com	FALSE	Filmsammlung	S 1	HH	44
Emma	Goldman	345678	libertat@gmx.de	FALSE	Buchhandlung	S 3	HB	44

2. Normalform (2NF)

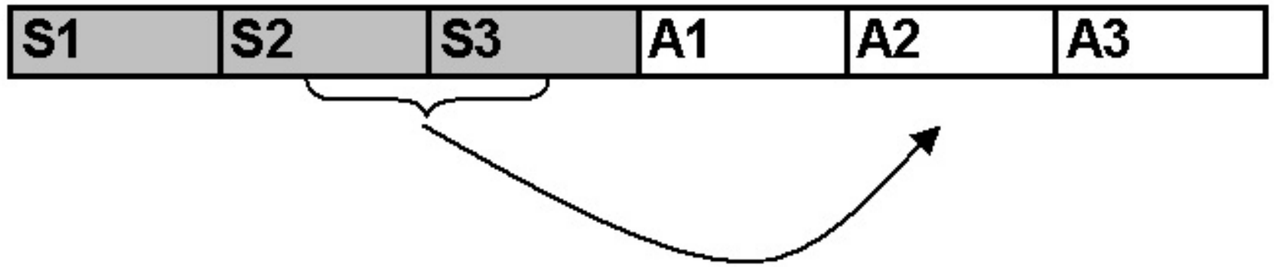
- Eine Relation ist in 2. Normalform (2NF),
  - wenn sie 1NF erfüllt und jedes Attribut, das nicht zum Primärschlüssel gehört, voll von diesem abhängig ist.

  - "Alle Attribute, die nicht Teil des Schlüssels sind, hängen voll funktional von diesem ab."

  Ist der Primärschlüssel einfach, so ist 2NF trivialerweise erfüllt.
  Daher ist die obige Tabelle schon in 2NF.

Verstoss gegen 2. Normalform

Folgendes schematisches Beispiel verdeutlicht einen Verstoss:



Verstoß gegen 2. Normalform

3. Normalform (3NF)

"Eine Relation ist in der dritten Normalform, wenn für jede nicht triviale funktionale Abhängigkeit T → A mit T Obermenge eines Schlüssels ist oder A (mindestens) ein Primattribut enthält."

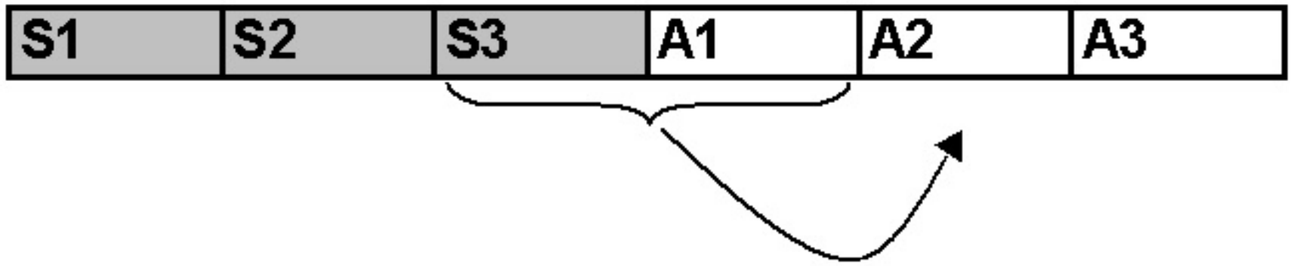
- Aus 2NF und Minimalität des Schlüssels ergibt sich, dass T mindestens ein Nichtschlüsselattribut enthalten muss, um gegen 3NF zu verstoßen.

3. Normalform (3NF) informal

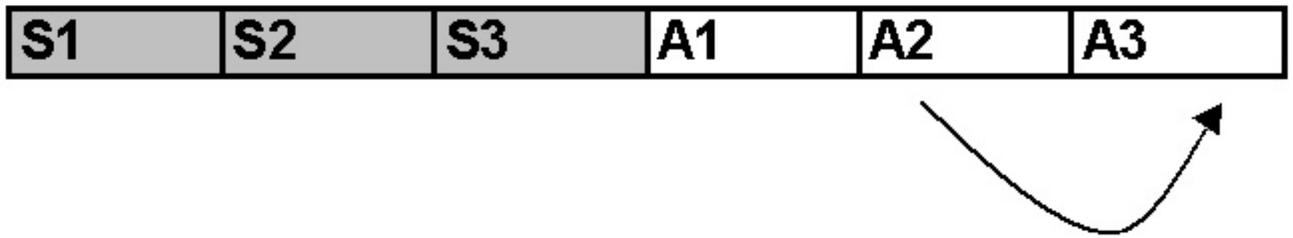
Eine Relation ist 3NF genau dann, wenn die nicht Schlüsselattribute (nicht Teil des Primary Key) beide Eigenschaften haben:

1. voneinander unabhängig
2. unwiderruflich abhängig vom Primary Key

3. Normalform (3NF) informal



Verstoß gegen 3. Normalform



Verstoß gegen 3. Normalform, einfacher Fall

Boyce-Codd Normalform (BCNF)

"Eine Relation ist in der Boyce-Codd Normalform, wenn für jede nicht triviale funktionale Abhängigkeit T → A mit T Obermenge eines Schlüssels ist oder A (mindestens) ein Primattribut enthält."

- Es gibt keine transitiven funktionalen Abhängigkeiten. Aus S → A darf sich nicht A → T ableiten lassen.
- Alle funktionalen Abhängigkeiten sind vom Primary Key abhängig

Meine Vorgehensweise

- Alles was, gemäss den Anforderungen, eigenständig und unabhängig von existierenden Entitäten gemanaged werden muss, ist eine neue eigenständige Entität oder neue Beziehung zwischen Entitäten.
- Managen heisst: Alles was zu unterschiedlichen Zeiten erst ins System kommt und eigene `insert`, `update`, und `delete` Regeln hat.

Normalisierung: Weitere Vorgehensweisen

- 5 Regeln:
  http://www.databaseanswers.org/downloads/Marc_Rettig_5_Rules_o
- J. Celko's 13 Regeln:
  - Wichtigste: If you have to change more than one row to update, insert or delete a simple fact, then the table is not normalized.

- Does the table model either a set of one and only one kind of entity or one and only one relationship? This is what I call disallowing a “Automobiles, Squids and Lady GaGa” table. Following this rule will prevent ‘Multi-valued dependencies’ (MVD) problems and it is the basis for the other heuristics.

- Does the entity table make sense? Can you express the idea of the table in a simple collective or plural noun? To be is to be something in particular; to be everything in general or nothing in particular is to be nothing at all (this is known as the Law of Identity in Greek logic). This is why EAV does not work – it is everything and anything.

- Do you have all the attributes that describe the thing in the table? In each row? The most important leg on a three-legged stool is the leg that is missing.

- Are all the columns scalar? Or is a column serving more than one purpose? Did you actually put hat size and shoe size in one column? Or store a CSV list in it?

- Do not store computed values, such as (unit_price * order_qty). You can compute these things in VIEWs or computed columns.

- Does the relationship table make sense? Can you express the idea of the table in a simple sentence, or even better, a name for the relationship? The relationship is “marriage” and not “person_person_legal_thing”

- Did you check to see if the relationship is 1:1, 1:m or n:m? Does the relationship have attributes of its own? A marriage has a date and a license number that does not belong to either of the people involved. This is why we don't mind tables that model 1:1 relationships.

- Does the entity or relationship have a natural key? If it does, then you absolutely have to model it as the PRIMARY KEY or a UNIQUE constraint. Is there a standard industry identifier for it? Let someone else do all that work for you.

- If you have a lot of NULL-able columns, the table is probably not normalized.

- The NULLs could be non-related entities or relationships.

- Do the NULLs have one and only one meaning in each column?

- If you have to change more than one row to update, insert or delete a simple fact, then the table is not normalized.

- Did you confuse attributes, entities and values? Would you split the Personnel table into “Male_Personnel” and ”Female_Personnel” by splitting out the sex code? No, sex is an attribute and not an entity. Would you have a column for each shoe size? No, a shoe size is a value and not an attribute.

Danke für die Zusammenarbeit