SFB441 Linguistische Datenstrukturen

Mini-Workshop „Relationale Datenbanken und SQL“

Dirk Wiebel
14.07.2003

1.1 Der Begriff „Datenbank“

"Eine Datenbank ist eine Sammlung von nicht-redundanten Daten, die von mehreren Applikationen benutzt werden" [Howe 83]

- Daten
- keine Redundanz
- standardisierte Schnittstelle

Warum nicht einfache (Text)Dateien als Datenspeicher?
--> Redundanzproblem (auch Adressierung/Verpointerung,...)

Wie wird die standardisierte Schnittstelle realisiert?
--> Datenbank-Managementsysteme (**DBMS**)

3-Ebenen-Modell:
- Externe Ebene: Anwender-, bzw. Programmschnittstellen (Eintragsmasken/Funktionen)
- Konzeptuelle Ebene: Bereitstellung der logischen Struktur (Einheiten, Datentypen, Beziehungen,...)
- Interne Ebene: Konkrete physikalische Speicherung und Datenverwaltung DBMS-intern

Weitere Vorteile von DBMS:
Integration von Backuplösungen, Zugriffsberechtigungen, Konkurrenzvermeidung durch Sperrung,...

==> Datenbank = Daten + DBMS

1.2 **Relationale** Datenbanken

Daten werden in **einfachen Tabellen** mit Zeilen und Spalten gespeichert/modelliert.

Zur Verwaltung werden Relationale Datenbankmanagementsysteme (**RDBMS**) eingesetzt, u.A. IBM DB2, Oracle, PostgreSQL, MySQL, Microsoft SQL Server, ...

1.3 Konzeption von Datenmodellen in RDBMS

„**Entity Relationship**“-Modell: Modellierung als Abbild einer (realen Mini-) Welt mit Hilfe folgender Typen von Einheiten und deren Beziehungen:

- **Objekttypen** (Objektabellen), **Attribute** (deren Spalten): Buchtitel, Lexikoneinträge, Quellen, Sprachen, Konzepte,...
- **Beziehungstypen**: x *steht in* y, a *ist Antezedens von* b

Bspp.: Tabellen mit Objekttypen (Bücher, Konzepte)

Id	Konzept
1	Kopf
2	Kapuze
3	Kapuziner
4	Caput
5	Kappenzipfel

Id	Titel
1	Buch1
2	Buch2
3	Buch3
4	Buch4
5	Buch5

Auch Beziehungstypen werden in RDBMS durch Tabellen realisiert (**Normalisierung**):
Bspp.: Buchstandorte (1:n), Antezedenstypen (n:m)

1:n-Beziehung

Id	Titel	Ort_id
1	Buch1	3
2	Buch2	3
3	Buch3	1
4	Buch4	3
5	Buch5	2

Ort_id	Ort
1	Bibliothek
2	Raum 2.04
3	Sekretariat
4	Projekt C1
5	Verschollen

Id	Lemma
1	Kopf
2	Kapuze
3	Kapuziner
4	Caput
5	Kappenzipfel

n:m-Beziehung

lem_id	ant_id
1	4
2	4
2	5
3	4
3	5
5	4
...	...

1.4 Art der Tabellen: **Datentypen** und **Schlüssel**

- minimaler Speicherbedarf --> Unterscheidung von Datentypen
- **Datenintegrität** --> Erstellung von Schlüsseln (Primäre S. und Fremdschlüssel)

Beispiele für Datentypen	SQL-Abkürzung	Bytes
Integer: Ganze Zahlen von -2147483647 bis 2147483647	INT	4
String: Feste Anzahl von Buchstaben/Zahlen	CHAR(x)	x
Variabler String: Variable Anzahl von Buchstaben/Zahlen	VARCHAR(x)	y+1
Große variable Strings: mehr als 255 Zeichen	TEXT	y+2
Binärdaten: Audiodaten, Bilddaten, kompilierte Programme	BLOB	y+2

Schlüssel:

- Ein primärer Schlüssel identifiziert einen Datensatz eindeutig. (Empfehlenswert: ID)
- Ein Fremdschlüssel „bindet“ einen Eintrag an einen Datensatz einer anderen Tabelle. Wenn beim Eintrag eines Datensatzes ein Fremdschlüssel definiert ist und der Datensatz in der anderen Tabelle nicht vorhanden ist (Beispiel Buch und Standort, 1:n), weist das DBMS den Eintragsversuch zurück. Duch die Restriktion werden fehlerhafte Einträge vermieden. Philosophie: Kein Eintrag und eine Warnung sind besser als ein falscher Eintrag.

1.5 Modellierungsbeispiele und -herausforderungen

B6-Ausschnitt mit Objekten incl. Datentypen und Beziehungen (Skizze/Poster)

Problematik: Wieviel ist die Nicht-Redundanz wert?
Bspp.: Telefonbuch vs. Mitarbeiter-DB SFB441

Entity-Relationship-Modell --> DB-Konzept entwickeln

Normalisierungsprinzipien (stark vereinfacht!):
- Dateneinheiten atomisieren
- Redundanz vermeiden
- Konsistenz/Integrität sichern
- Abhängigkeiten durch Schlüssel definieren

2. **SQL** (Structured Query Language)

Beispiele hier aus **MySQL**, da frei verfuegbar und relativ einfach. Bei verschiedenen DBMS (PostgreSQL, DB2, Oracle, MSSQL,...) **immer** SQL-Referenz beachten.

2.1. Was kann SQL?

- Daten abfragen
  aber auch:
- Daten einfügen
- Daten manipulieren
  und auch:
- Tabellen erstellen
- Benutzerrechte setzen
- Daten exportieren und importieren
- Funktionen und Sichten erstellen
- ...

--> SQL wird als Schnittstelle vieler RDMBS verwendet. (Vorsicht: "Dialekte")

2.2. SELECT-Syntax

Einfaches Select ohne Konditionen:

```
# SELECT * FROM tabelle1;
# SELECT id,name,titel FROM tabelle1;
```

Select mit Konditionen:

```
# SELECT id,name,titel FROM tabelle1 WHERE id=1;
# SELECT id,name,vorname,titel FROM tabelle1 WHERE vorname LIKE "%laus";
(-> Ladislaus, Nikolaus, Klaus, Steinlaus)
```

Select über mehrere Tabellen (inner join):

```
# SELECT id,titel,verlag FROM tabelle1, tabelle2 WHERE tabelle1.verlags_id = tabelle2.id;
```

2.3. INSERT-Syntax

```
# INSERT INTO TABLE tabelle1 values (23,"Sauer", "Herrmann", "Relationale
Datenbanken. Theorie und Praxis. 5. Aufl.","2002",5);

# INSERT INTO TABLE tabelle2 values(5, "Addison-Wesley", "München, Boston, San
Francisco,...");
```

2.4. UPDATE-Syntax

UPDATE tabelle1 SET spalte1=“neuer Wert“ where id=15;

2.5. CREATE-Syntax

Erstellen einer neuen Datenbank:
CREATE DATABASE datenbank1;

CREATE TABLE tabelle1 (id INT, name VARCHAR(80), vorname VARCHAR(50), titel VARCHAR(80));

2.6. Weiteres

DELETE:
DELETE FROM tabelle1;
DELETE FROM tabelle1 where id=10;

DROP:
DROP TABLE tabelle1;

SHOW/DESCRIBE:
SHOW TABLES;
DESCRIBE tabelle1;

```
+-------------+--------------+------+-----+---------+----------------+
| Field       | Type         | Null | Key | Default | Extra          |
+-------------+--------------+------+-----+---------+----------------+
| id          | int(11)      |      | PRI | NULL    | auto_increment |
| title       | varchar(255) | YES  |     | NULL    |                |
| description | text         | YES  |     | NULL    |                |
+-------------+--------------+------+-----+---------+----------------+
```

3. Übungen auf MySQL

#ssh -lBENUTZERNAME barlach.sfb.uni-tuebingen.de
#PASSWORD
(Benutzername: workshopbenutzer, Passwort: *****)
#mysql -p
#PASSWORD

mysql> use db_workshop;

Einfachste Beispieltabelle: movies
mysql> DESCRIBE movies;

```
+-------------+--------------+------+-----+---------+----------------+
| Field       | Type         | Null | Key | Default | Extra          |
+-------------+--------------+------+-----+---------+----------------+
| id          | int(11)      |      | PRI | NULL    | auto_increment |
| title       | varchar(255) | YES  |     | NULL    |                |
| description | text         | YES  |     | NULL    |                |
+-------------+--------------+------+-----+---------+----------------+
```

Einfaches SELECT:

mysql> SELECT id,title from movies where id<5;
mysql> SELECT id,title from movies where title like „%Bond%“; *(%=Wildcard)*
mysql> SELECT title,description from movies where description like „%Bruce Willis%“;

Komplexeres SELECT:

```
mysql>  DESCRIBE actors
+----------+--------------+------+-----+---------+----------------+
| Field    | Type         | Null | Key | Default | Extra          |
+----------+--------------+------+-----+---------+----------------+
| actor_id | int(11)      |      | PRI | NULL    | auto_increment |
| actor    | varchar(100) | YES  |     | NULL    |                |
+----------+--------------+------+-----+---------+----------------+
mysql> DESCRIBE actmov
+----------+---------+------+-----+---------+-------+
| Field    | Type    | Null | Key | Default | Extra |
+----------+---------+------+-----+---------+-------+
| movie_id | int(11) | YES  |     | NULL    |       |
| actor_id | int(11) | YES  |     | NULL    |       |
+----------+---------+------+-----+---------+-------+
```

mysql> SELECT actors.actor, movies.title FROM movies, actors, actmov WHERE (actmov.movie_id=movies.id AND actmov.actor_id=actors.actor_id) AND actors.actor LIKE "%Sean%";

oder:

mysql> SELECT movies.title FROM movies
LEFT JOIN actmov ON actmov.movie_id=movies.id
LEFT JOIN actors USING (actor_id)
WHERE actors.actor LIKE "%Diaz%";

3.1. Eine eigene Datenbank: Die wichtigsten Datentypen in MySQL:

TINYINT:	-128 bis 127	1 Byte
SMALLINT:	-32768 bis 32767	2 Byte
INT:	-2147483648 bis 2147483647	4 Byte
VARCHAR(x):	x Zeichen, x ≤ 255	y+1 Byte (y=tats. Zeichen)
CHAR(x):	x Zeichen, x ≤ 255	x Bytes
TEXT	≤ 2^{16} Zeichen	y+1 Bytes (y=tats. Zeichen)
YEAR:	vierstellige Jahreszahl	1 Byte

Extra: Der NULL-Wert und der Parameter "NOT NULL"
Ein nicht gegebener Wert wird mit „NULL“ realisiert. Die Möglichkeit eines NULL-Wertes kann unterbunden werden.

CREATE TABLE tabelle1 (id INT NOT NULL, name VARCHAR(80), vorname VARCHAR(50), titel VARCHAR(80));

Extra: Der Parameter auto_increment bei MySQL
Um nicht bei jedem Eintrag den Stand des Primärschlüssels nachlesen zu müssen, kann dieser automatisch um 1 erhöht werden:

CREATE TABLE tabelle1 (id INT **AUTO_INCREMENT** PRIMARY KEY, name VARCHAR(80), vorname VARCHAR(50), titel VARCHAR(80));

Bei DB2:
CREATE TABLE tabelle1 (id INT **GENERATED ALWAYS AS IDENTITY (START WITH 1, INCREMENT BY 1)**, name VARCHAR(80), vorname VARCHAR(50), titel VARCHAR(80));

3.2. Vorschlag: Konzeption einer Buchdatenbank mit:

Eindeutiger Identifikationsnummer,
Titel,
Autor,
Jahr,
Auflage,
Verlag,
Signatur,
Standort

Ist es möglich/notwendig, mit Relationen zu arbeiten? Wenn ja, welche Spalte sollte aus Redundanzgründen „ausgelagert“ werden? Welche Datentypen sind sinnvoll?

3.2. Übersetzung der Tabellenstruktur(en) in eine CREATE-Query

3.3. Zwei oder mehr Einträge in die Tabelle/n mit einer INSERT-Query

3.4. Abfrage von Autor, Titel und Standort eines Buchs mit einer SELECT-Query

4. Literaturhinweise:

Sauer, Herrmann (2002): Relationale Datenbanken - Theorie und Praxis. Mit einem Beitrag zu SQL-3 von Klaus Grieger. 5., aktualisierte und erweiterte Auflage. Addison-Wesley, München u.a. UB-Signatur: inf N 5401.

Elmasri, Ramez & Shamkant B. Navathe (1994): Fundamentals of Database Design. Second Edition. The Benjamin/Cummmings, Redwood City.

Praktisch und DBMS-bezogen:
Mysql-Manual auf www.mysql.com
DB2 Administration Guides (Print-Versionen bei mir)
DB2 SQL-Reference (PDF-Version bei mir)

Sonstiges:
Howe, D.R. (1983): Data Analysis for Data Base Design. Edward Arnold.