Datenbanksysteme 1

Dozent: S. Keller

Grauer Bernhard, Koeppel Damian

15. Januar 2013

Inhaltsverzeichnis

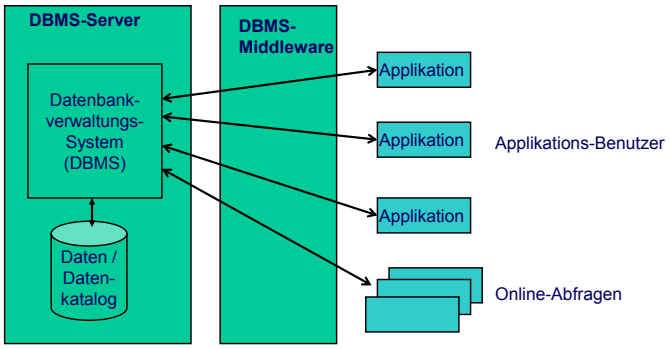
1 Basics

DBS = DBMS + (n*)DB

DBS Datenbanksystem

DBMS Datenbankmanagementsystem

DB Datenbasis



1.1 DBMS

1.1.1 Eigenschaften

- Verwaltet zentrale Datenbasis (Datenbank)
- Anwendungen greifen via DBMS auf die Daten zu
- Die Daten sind strukturiert und die Struktur ist im Datenkatalog beschrieben
- Client-Server Struktur
- Sichert Datenintegrität
- Stellt Datenpflege, Datenschutz und Datensicherheit sicher

1.1.2 Anforderungen

- Redundanzfreiheit
- Datenintegrität (Kosistenz, Sicherheit, Schutz)

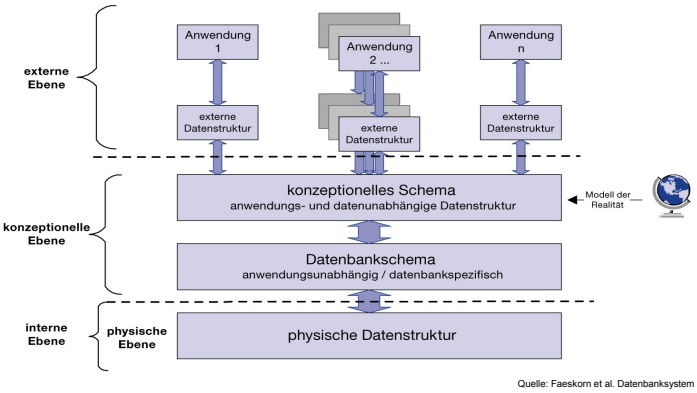
1.1.3 Funktionen

- Transaktionen
- Synchronisation paralleler Zugriffe (Mehrnutzerbetrieb)
- Sicherheit: Authentifizierung, Autorisierung
- Backup und Recovery
- Abfragesprache, Schnittstellen

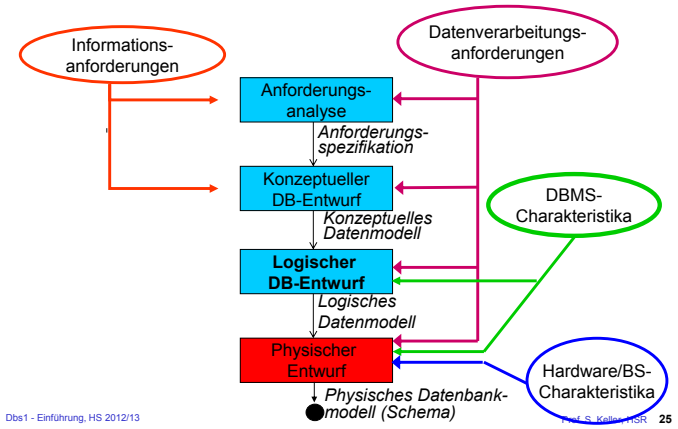
1.1.4 Modelle

- Hierarchisches DBM (XML)
- Netzwerk-DBM
- Relationales-DBM (SQL)
- Postrelationale-DBM (OODB)

1.2 3-Ebenen-Modell



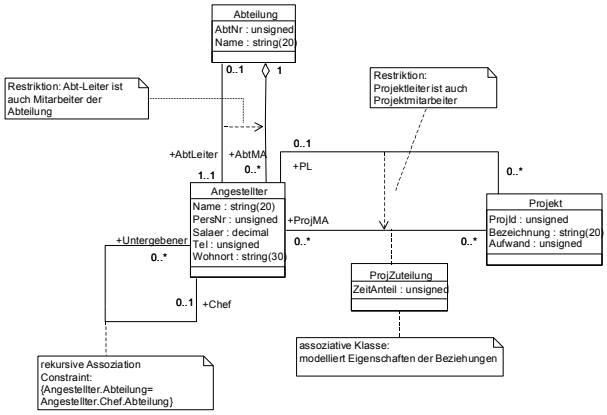
1.3 Entwurfsprozess



2 Datenmodellierung

2.1 ER-Modell

2.2 UML-Modell

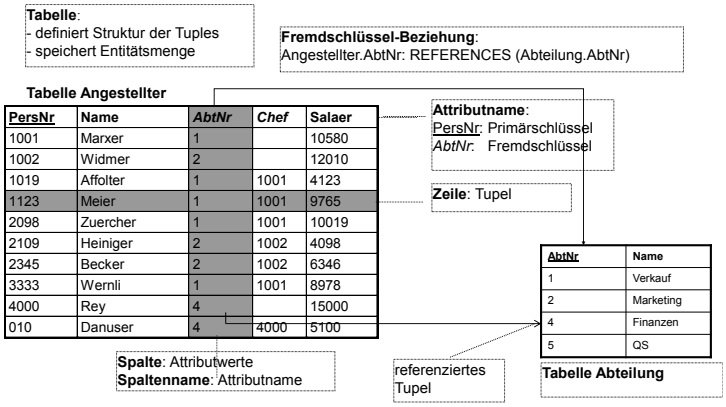


2.3 Relationales Modell

Entitität Individuelles Element (Objekt) des betrachteten Systems

Relation beschreibt eine Entitätsmenge

Tupel repräsentiert eine Entität



2.3.1 Schlüssel

Schlüssel Attribute oder eine Kombination von Attributen, die ein Tupel eindeutig identifizieren.

Primärschlüssel Der aus den möglichen Schlüsseln (= Schlüsselkandidaten) ausgewählte identifizierende Schlüssel. Wird für Fremdschlüsselbeziehungen verwendet.

Schlüsselkandidat Attribut oder Kombination davon, das/die ein Tupel identifiziert.

Surrogatschlüssel Künstlicher Schlüssel. Wird häufig eingeführt, wenn keiner der Schlüsselkandidaten obige Eigenschaften erfüllt.

2.4 Relationale Schreibweise

2.4.1 Tabellen

```
TableName (
    Attr1 [Type] [PK] [NOT NULL|NULL] [UNIQUE] [REFERENCES TabName(Attr)],
    ...,
```

```
    [PRIMARY KEY CONSTRAINT(Attr1, Attr2, ...)]
)
```

2.4.2 Attribut-Datentypen

- NUMBER, INT, INTEGER, DECIMAL[(10,2)]
- TEXT, STRING, VARCHAR
- TEXT(3), CHAR(3)
- DATE, TIME, DATETIME
- BOOLEAN
- ENUM, ENUM(ROT,GELB), DOMAIN

2.4.3 UML in relationales Modell

Abbildung von Klassen und Attributen

- pro Klasse eine Tabelle
- 1. Normalform, optionale Attibute mittels NULL
- mindestens einen Primärschüssel

Abbildung von Assoziationen und Aggregationen

Abbildung von Generalisierungen

3 Normalisierung

3.1 1. Normalform

- Wertebereiche der Attribute sind atomar.
- Strukturierte Werte (Mengen, Wiederholungsgruppen) sind nicht zugelassen.

3.2 2. Normalform

- Jedes Nichtschlüsselattribut von ist jedem Schlüsselkandidaten voll funktional abhängig.

3.3 3. Normalform

- Kein Nichtschlüsselattribut ist von irgendeinem Schlüssel transitiv abhängig.

3.4 Boyce-Codd-Normalform

- Jede Determinante ist Schlüsselkandidat.

4 SQL

4.1 Data Definition Language (DDL)

4.1.1 Basisdatentypen

BOOLEAN Boolscher Datentyp

SMALLINT Ganzzahl (2 Byte)

INT / INTEGER Ganzzahl (4 Byte)

BIGINT Ganzzahl (8 Byte)

REAL, FLOAT, DOUBLE Fliesskomma-Zahl (8 Byte)

NUMERIC (precision, scale), DECIMAL(precision, scale) Festkommazahl

CHAR(size), CHARACTER(size) String mit fixer Länge

VARCHAR(size) String mit variabler Länge

DATE Jahr, Monat, Tag

TIME Stunde, Minute, Sekunde

INTERVAL Zeitintervall

DATETIME DATE + TIME

BINARY, VARBINARY, LONGBINARY Binäre Datentypen

CLOB, BLOB Grosse Text- / Binärdaten

Spezelles bei PostgreSQL:

- kein FLOAT
- TEXT für Zeichenketten bel. Länge
- kein DATETIME, siehe TIMESTAMP

4.1.2 Create Table

```
CREATE TABLE table_name ( [
  { column_name data_type [ DEFAULT default_expr ] [ column_constraint [ ... ] ]
    | table_constraint }
    [, ... ]
] )

where column_constraint is:

[ CONSTRAINT constraint_name ]
{ NOT NULL |
  NULL |
  UNIQUE |
  PRIMARY KEY |
  CHECK (expression) |
  REFERENCES reftable [ ( refcolumn ) ]
    [ ON DELETE action ] [ ON UPDATE action ] }

and table_constraint is:
```

```
[ CONSTRAINT constraint_name ]
{ UNIQUE ( column_name [, ... ] ) |
  PRIMARY KEY ( column_name [, ... ] ) |
  CHECK ( expression ) |
  FOREIGN KEY ( column_name [, ... ] ) REFERENCES reftable [ ( refcolumn [, ... ] ) ]
    [ ON DELETE action ] [ ON UPDATE action ] }

and action is:

CASCADE | RESTRICT | SET NULL | SET DEFAULT
```

4.1.3 Alter Table

```
ALTER TABLE name [ * ]
    action [, ... ]
ALTER TABLE name [ * ]
    RENAME [ COLUMN ] column TO new_column
ALTER TABLE name
    RENAME TO new_name

where action is one of:

    ADD [ COLUMN ] column type [ column_constraint [ ... ] ]
    DROP [ COLUMN ] column [ RESTRICT | CASCADE ]
    ALTER [ COLUMN ] column TYPE type
    ALTER [ COLUMN ] column SET DEFAULT expression
    ALTER [ COLUMN ] column DROP DEFAULT
    ALTER [ COLUMN ] column { SET | DROP } NOT NULL
    ADD table_constraint
    DROP CONSTRAINT constraint_name [ RESTRICT | CASCADE ]
```

4.1.4 Drop Table

```
DROP TABLE name [, ...] [ CASCADE | RESTRICT ]
```

4.1.5 Create Index

```
CREATE [ UNIQUE ] INDEX name ON table
    ( column [, ...] )
```

4.1.6 Drop Index

```
DROP INDEX name [, ...] [ CASCADE | RESTRICT ]
```

4.2 Data Manipulation Language (DML)

4.2.1 Insert

```
INSERT INTO table [ ( column [, ...] ) ]
    { DEFAULT VALUES | VALUES ( { expression | DEFAULT } [, ...] ) }
```

4.2.2 Select

```
SELECT [ DISTINCT [ ON ( expression [, ...] ) ] ]
    * | expression [ AS output_name ] [, ...]
    [ FROM from_item [, ...] ]
    [ WHERE condition ]
    [ GROUP BY expression [, ...] ]
    [ HAVING condition [, ...] ]
    [ WINDOW window_name AS ( window_definition ) [, ...] ]
    [ ORDER BY expression [ ASC | DESC ] [, ...] ]
    [ LIMIT { count | ALL } ]
    [ OFFSET start ]
```



```
where from_item can be one of:

    table_name [ [ AS ] alias ]
    ( select ) [ AS ] alias
    from_item join_type from_item [ ON join_condition ]

where join_type is:
    { [ LEFT | RIGHT ] [ CROSS | INNER | OUTER | NATURAL ] JOIN }
```

4.2.3 Update

```
UPDATE table_name [ [ AS ] alias ]
    SET { column_name = { expression | DEFAULT } |
          ( column_name [, ...] ) = ( { expression | DEFAULT } [, ...] ) } [, ...]
    [ FROM from_list ]
    [ WHERE condition ]
```

4.2.4 Delete

```
DELETE FROM table_name [ [ AS ] alias ]
    [ WHERE condition]
```

4.2.5 Copy

TODO: Unterbafragen (NOT) IN, ANY, EXIST, ALL
TODO: Mengenoperatoren UNION, MINUS; INTERSECT
TODO: Window Fönktschens
TODO: CTE

4.2.6 Create View

```
CREATE VIEW name [ ( column_name [, ...] ) ]
    AS query
```

4.2.7 Drop

```
DROP VIEW name [, ...]
```

5 Transaction

TODO: ACID (Atomicity, Consistency, Isolation, Durability)

5.1 SQL

5.1.1 Transaction

```
BEGIN [ TRANSACTION ] [ transaction_mode [, ...] ]

where transaction_mode is one of:

    ISOLATION LEVEL { SERIALIZABLE | REPEATABLE READ | READ COMMITTED | READ UNCOMMITTED }
    READ WRITE | READ ONLY
    [ NOT ] DEFERRABLE

COMMIT [ TRANSACTION ]

ROLLBACK [ TRANSACTION ]
```


5.1.2 Savepoint

```
SAVEPOINT savepoint_name

ROLLBACK TO [ SAVEPOINT ] savepoint_name

RELEASE [ SAVEPOINT ] savepoint_name
```

TODO: Serialisierbarkeit, Isolation Levels
TODO: Isolationsverfahren (Transaktionen_Fortsetzung.pdf , S. 43)

6 JDBC

6.1 Type Mapping

Datenbanktyp	Java Datentyp
CHAR, VARCHAR, LONGVARCHAR	String
BIT, BOOLEAN	boolean
INTEGER	int
BIGINT	long
REAL	float
FLOAT, DOUBLE	double

6.2 Transaction Levels

Level	Wert	Beschreibung
TRANSCATION_NONE	0	Es werden keine Sperren in der DB gesetzt
TRANSACTION_READ_UNCOMMITTED	1	Lesende Transaktionen verursachen keine Sperren.
TRANSACTION_READ_COMMITTED	2	Lesende Transaktionen verursachen Sperren
TRANSACTION_SERIALIZABLE	3	Transaktionen werden geblockt und hintereinander ausgeführt.

6.3 Prepared Statement

```
1    PreparedStatement updateOrt;
2
3    String updateString = "UPDATE Dozent SET ort = ?";
4    updateOrt = connection.prepareStatement(updateString);
5
6    updateOrt.setString("1, Zuerich");
```

6.4 Metadaten

Metadaten geben den JDBC Benutzer Informationen über die Datenbank die im DBMS Schema gespeichert sind.

```
1    DatabaseMetaData dbmd = connection.getMetaData();
2    String URL = dbmd.getURL;
```

6.5 Code Beispiel

```
1  public class A1_Transaktion {
2          public static void main(String[] args) {
3                  final String user = "anguser";
4                  final String password = "angproj";
5                  final String database = "jdbc:postgresql://localhost/angproj";
6                  try (Connection connection = DriverManager.getConnection(database,user,
                       password)) {
7                       try(Statement stmt = connection.createStatement()) {
8                               connection.setAutoCommit(false);
```

```
9                                   connection.setTransactionIsolation(Connection.
                                        TRANSACTION_SERIALIZABLE);
10                                  stmt.execute("INSERT INTO ang VALUES (42, 'Meier, Max', 42)");
11                                  stmt.execute("UPDATE ang SET chef=42 WHERE chef=11");
12                                  stmt.executeUpdate("DROP TABLE Wegdamit"); //Also valid for
                                        CREATE INSERT DELETE
13                                  ResultSet rs = stmt.executeQuery("SELECT * FROM Nochda");
14                                  while(rs.next()){
15                                          String sbuff = rs.getString(1);
16                                          System.out.println(sbuff);
17                                  }
18                                  //Batch Upddate
19                                  stmt.addBatch("UPDATE projekt SET leiter=42 WHERE leiter=11");
20                                  stmt.addBatch("DELETE FROM zuteilung WHERE persnr=11");
21                                  stmt.executeBatch();
22                                  connection.commit();
23                          } catch (SQLException ex) {
24                                  System.err.println(ex.getMessage() +"\nRollback...");
25                                  connection.rollback();
26                          }
27                  } catch (SQLException ex) {
28                          System.err.println("SQLException: " + ex.getMessage());
29                  }
30          }
31
32  }
```

