Chapter 5: Relational Database Definition

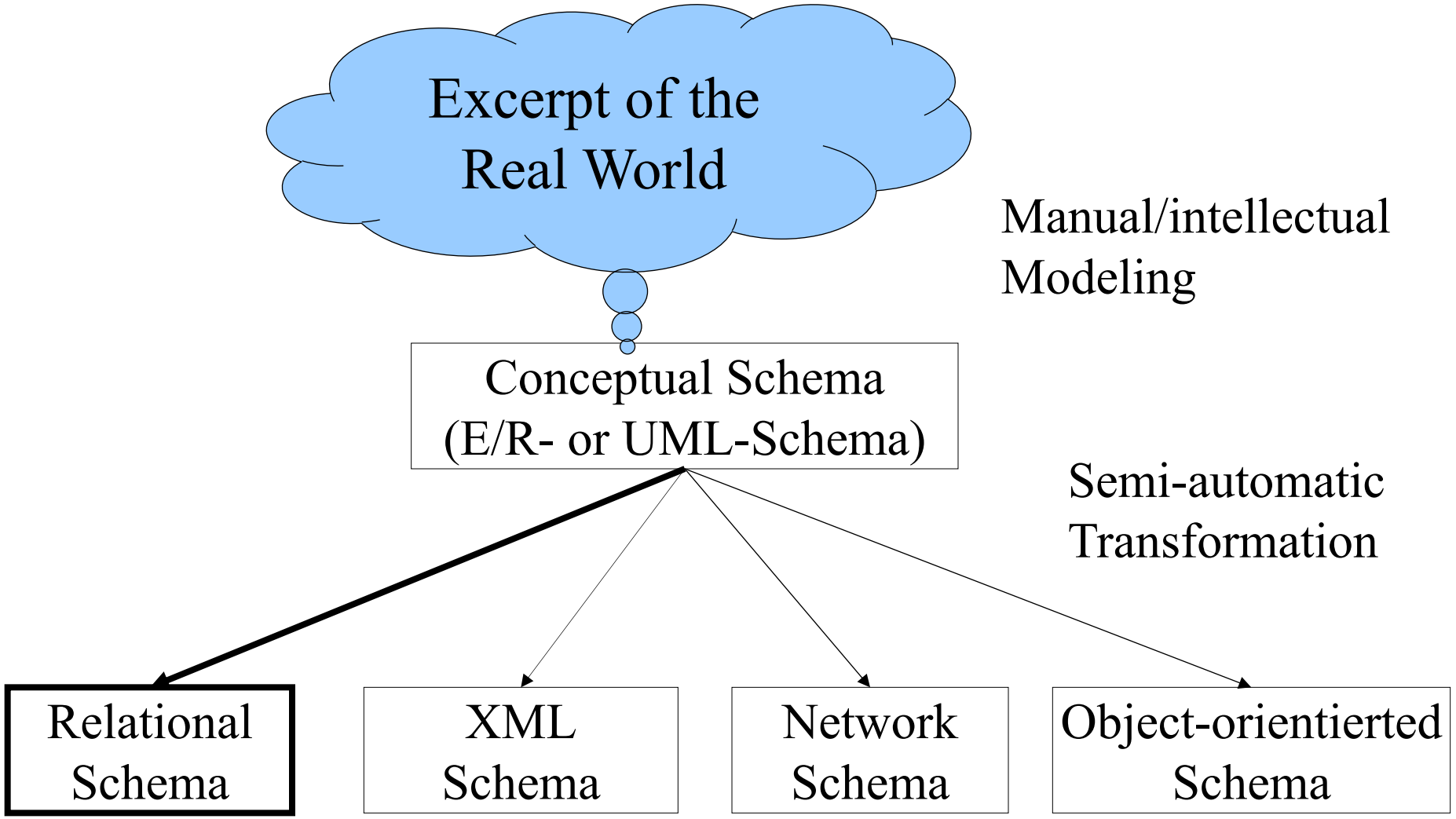
Content:

- How to transform the relational model into a database schema

Next:

- Query the database

Data modeling



SQL

SQL: Structured Query Language

Former name: SEQUEL

Standardized query language for relational DBMS:
SQL-89, **SQL-92**, SQL-99, SQL:2003 (XML-Extensions)

SQL is a **declarative query language**

Parts of SQL

Parts:

- DRL: Data Retrieval Language
- DML: Data Manipulation Language
- DDL: Data Definition Language
- DCL: Data Control Language
- TCL: Transaction Control Language

DRL: Data Retrieval Language

DRL contains statements for queries

Simple queries consist of the three parts:

select, **from** and **where**

select *list of attributes*
from *list of relations*
where *predicates*;

DML: Data Manipulation Language

DML contains statements to

- Insert data
- Delete data
- Change data

insert, delete, update

DDL: Data Definition Language

- With the DDL the schema of a database can be defined
- Also contains statements to control the access path to the database

  e.g. **create table, alter table, create view, create index**

- Correspondingly also delete statements **drop ..**

DCL: Data Control Language

- Contains statements mostly concerned with rights, permissions and other controls of the database system (authorization)

e.g. **grant, revoke**

TCL: Transaction Control Language

- Contains statements to control transactions
- A transaction is a set of interactions between application / user and the database
- Will be dealt with later in the section transaction management

e.g. **commit, rollback**

Different Ways of using SQL

- Interactively (command line application or GUI)
- Dynamic SQL: As a library in a programming language
- Embedded SQL: As an extension to a programming languages

Dynamic SQL

- Is used when queries are not yet known when the program is compiled
- Standardized Interfaces
  - ODBC (Open Database Connectivity)
  - JDBC (for Java)
- More flexible, but usually a bit slower than embedded SQL

Dynamic SQL: Example

```
String db_url = "jdbc:postgresql://localhost:5432/uni";
Connection connection = DriverManager.getConnection(db_url,
                                                    "alex",
                                                    "123456");
Statement statement = connection.createStatement();
ResultSet result = statement.executeQuery("select name, semester from
students");
while (result.next()) {
   String name = result.getString("name");
   int semester = result.getInt("semester")
   System.out.println(name + " " + semester);
}
```

Embedded SQL

- SQL statements are directly embedded in the corresponding host language (e.g. C, C++, Java, etc.)
- SQL statements are marked with a preceding **EXEC SQL**
- SQL statements are then replaced by a pre-processor by constructs of the corresponding language

(Embedded SQL: Example)

```
EXEC SQL CONNECT TO postgresql@localhost:5432 AS uni USER alex
PASSWORD 123456;

EXEC SQL BEGIN DECLARE SECTION
char name[50];
int semester;
EXEC SQL END DECLARE SECTION

EXEC SQL DECLARE C1 CURSOR FOR SELECT name, semester FROM
students;
EXEC SQL OPEN C1;

while (true) {
   EXEC SQL FETCH C1 INTO :name, :semester;
   printf("%s %i\n", name, semester);
}
```

SQL: Data Definition Language

DDL: Create Table Statement

Syntax diagram encompasses many pages!!

Simple form:

```
create table TableName (
        Attribute1 DataType1 [NOT NULL],
        …
        Attributen DataTypeN [NOT NULL]);
```

DDL: Example Create Table Statement

```
CREATE TABLE Professors
        (PersNr INTEGER NOT NULL,
        Name VARCHAR(30) NOT NULL,
        Level CHAR(2),
        Room INTEGER);

CREATE TABLE Lectures
        (LectureNr INTEGER NOT NULL,
        Title VARCHAR(30),
        WeeklyHours INTEGER,
        Given_by INTEGER);
```

DDL: Data types in SQL strings and numbers

- **VARCHAR** (n) variable length string, length maximal n Byte
- **CHAR[ACTER]** (n) fixed length string of n Byte
- **NUMERIC** [(p[, s])] signed number with p digits in total,
  s of them after the decimal place
  also **DEC[IMAL]** [(p,s)]
- **INT[EGER]** signed integer
- **SMALLINT** like INTEGER, smaller value range
- **FLOAT** [(p)] (rounded) floating point number (at least p Bits precision)
  REAL, DOUBLE PRECISION short cuts for FLOAT(p),
  values for p dependent on implementation

DDL: Data types in SQL
date and time

- **DATE** valid Date
- **TIME** time (from 00:00:00 bis 23:59:59)
- **TIMESTAMP** timestamp (combination of date and time)

(ORACLE only has DATE and uses this as timestamp)

DDL: Data types in SQL strings, binary data

- **LONG** variable string with up to 2 GB (**TEXT** SQL Server)
- **CLOB** string with up to 4 GB

- **RAW** (n) binary data of length n, n between 1 and 2000 Bytes
- **LONG RAW** binary data with up to 2 GB
- **BLOB** binary data with up to 4 GB

- **CFILE, BFILE** pointer to file (text, binary) (Oracle)
- **DATALINK** pointer to file (DB2)
- **MONEY / SMALLMONEY** (SQL Server)
- **…**

restricted operations on it!

DDL: Integrity constraints

- One of the tasks of a DBMS:
  guarantee the consistency of the data
- Semantical integrity constraints describe the properties of the mini world modelled
- DBMS can automatically check these constraints – once formulated

Primary Key Constraint

Value of an attribute or a combination of attributes does not occur twice in any instance of the data base

DDL: Primary key constraint

Value of an attribute or a combination of attributes does not occur twice in any instance of the data base

```
CREATE TABLE Table_Name (
        Attribute_1 Data_Type_1 [NOT NULL],
        …
        Attribute_n Data_Type_n [NOT NULL],

        [CONSTRAINT constraint_name_pk] PRIMARY KEY
                (Attribute_i, …,Attribute_p));
```

DDL: Example primary key

```
CREATE TABLE Professors (
      persNr INTEGER NOT NULL,
      name VARCHAR(30) NOT NULL,
      level CHAR(2),
      room INTEGER,
      PRIMARY KEY (persNr)
);
CREATE TABLE Lectures (
      lectureNr INTEGER NOT NULL,
      title VARCHAR(30),
      weeklyHours INTEGER,
      given_by INTEGER,
      PRIMARY KEY (lectureNr)
);
```

DDL: Primary Key Constraints (short form)

```
CREATE TABLE Professors (
      persNr INTEGER NOT NULL PRIMARY KEY,
      name VARCHAR(30) NOT NULL,
      level CHAR(2),
      room INTEGER
);

CREATE TABLE Lectures (
      lectureNr INTEGER NOT NULL PRIMARY KEY,
      title VARCHAR(30),
      weeklyHours INTEGER
);
```

DDL: Further integrity constraints

Besides primary keys there are some other integrity constraints, such as:

- NOT NULL
- Unique
- Check clauses

DDL: NOT NULL

- Enforces defined attribute values when inserting tuples
- Mandatory for primary keys
- Possible to give default value

```
CREATE TABLE defaults (
      id INTEGER NOT NULL PRIMARY KEY,
      location VARCHAR(80) DEFAULT 'GARCHING',
      vat SMALLINT DEFAULT 19,
      age SMALLINT DEFAULT 20,
      height SMALLINT NOT NULL
);
```

DDL: UNIQUE

Enforces key property (for candidate key)

```
CREATE TABLE Professors (
      persNr INTEGER PRIMARY KEY,
      name VARCHAR(30) NOT NULL,
      level CHAR(2) CHECK (Rang IN ('C2','C3','C4')),
      room INTEGER NOT NULL UNIQUE
);
```

DDL: Check clauses

With check clauses the range of values for an attribute can be restricted

Example:

```
CREATE TABLE Professors (
      PersNr INTEGER NOT NULL PRIMARY KEY,
      Name VARCHAR(80) NOT NULL,
      Level CHAR(2) CHECK (Level IN
                  ('C2','C3','C4','W1','W2','W3')),
      Room INTEGER CHECK (Room > 0 AND Room < 9999)
);
```

Referential Integrity

- Let *R* and *S* be two relations with schema R resp. S
- k is primary key of *R*
- Then f ϵ *S* is foreign key, if for all tuples *s* ϵ *S* holds:
  - *s.f* either only holds only null values or only values not null
  - If *s.f* has no null values, then there exists a tuple r ϵ R with *s.f* = *r.k*
- The fulfillment of these properties is called "*referential integrity*"

DDL: Example Referential Integrity

Lectures			
Lecture Nr	**Title**	**Weekly Hours**	**Given_by**
5001	Grundzüge	4	2137
5041	Ethik	4	2125
5043	Erkenntnistheorie	3	2126
5049	Mäeutik	2	2125
4052	Logik	4	2125
5052	Wissenschaftstheorie	3	2126
5216	Bioethik	2	2126
5259	Der Wiener Kreis	2	2133
5022	Glaube und Wissen	2	2134
4630	Die 3 Kritiken	4	2137

Professors			
PersNr	**Name**	**Level**	**Room**
2125	Sokrates	C4	226
2126	Russel	C4	232
2127	Kopernikus	C3	310
2133	Popper	C3	52
2134	Augustinus	C3	309
2136	Curie	C4	36
2137	Kant	C4	7

DDL: Referential integrity

CREATE TABLE Table_name (
Attribute_1 Data_type_1 [NOT NULL],
…
Attribute_n Data_type_n [NOT NULL],

[CONSTRAINT constraint_name_pk] PRIMARY KEY
(Attribute_i, …,Attribute_p),

CONSTRAINT constraint_name_fk FOREIGN KEY
(Attribute_j, …, Attribute_l) REFERENCES
Parent_table_name (Attribute_t, …, Attribute_v));

DDL: Example referential key

```
CREATE TABLE Professors (
      persNr INTEGER NOT NULL,
      name VARCHAR(30) NOT NULL,
      level CHAR(2),
      room INTEGER,
      PRIMARY KEY (persNr)
);

CREATE TABLE Lectures (
      lectureNr INTEGER NOT NULL,
      title VARCHAR(30),
      weeklyHours INTEGER,
      given_by INTEGER,
      PRIMARY KEY(lectureNr),
      FOREIGN KEY(given_by) REFERENCES Professors (persNr)
);
```

DDL: Example referential key (short form)

```
CREATE TABLE Professors (
      PersNr INTEGER NOT NULL PRIMARY KEY,
      Name VARCHAR(30) NOT NULL,
      Level CHAR(2),
      Room INTEGER
);

CREATE TABLE Lectures (
      LectureNr INTEGER NOT NULL PRIMARY KEY,
      Title VARCHAR(30),
      WeeklyHours INTEGER,
      Given_by INTEGER REFERENCES Professors
);
```

DDL: Foreign key variants

Changes to key attributes can automatically be propagated:

- set null: all foreign keys values which reference a key which was altered or deleted are set NULL
- cascade: all foreign keys values which reference a key which was altered or deleted are likewise altered (to the new value) resp. Deleted
- restrict: the update or delete operation is aborted

DDL: Example foreign key variants

Lectures			
Lecture Nr	**Title**	**Weekly Hours**	**Given_by**
5001	Grundzüge	4	2137
5041	Ethik	4	2125
5043	Erkenntnistheorie	3	2126
5049	Mäeutik	2	2125
4052	Logik	4	2125
5052	Wissenschaftstheorie	3	2126
5216	Bioethik	2	2126
5259	Der Wiener Kreis	2	2133
5022	Glaube und Wissen	2	2134
4630	Die 3 Kritiken	4	2137

Professors			
PersNr	**Name**	**Level**	**Room**
2125	Sokrates	C4	226
2126	Russel	C4	232
2127	Kopernikus	C3	310
2133	Popper	C3	52
2134	Augustinus	C3	309
2136	Curie	C4	36
2137	Kant	C4	7

DDL: Example foreign key variants

```
CREATE TABLE Lectures (
      LectureNr INTEGER NOT NULL PRIMARY KEY,
      Title VARCHAR(30),
      WeeklyHours INTEGER,
      Given_by INTEGER REFERENCES Professors
                  ON DELETE SET NULL);

CREATE TABLE attend (
      StudNr INTEGER REFERENCES Students
                  ON DELETE CASCADE,
      LectureNr INTEGER REFERENCES Lectures
                  ON DELETE CASCADE,
      PRIMARY KEY (StudNr, LectureNr));
```

Generated Values

Artificial values, surrogates, no semantic, mostly as keys:

- Directly in the table definition:

```
create table dept (
deptno serial primary key,
deptname varchar(50) not null);
```

insert with:

```
insert into dept values(default, 'I3')or
insert into dept values('I3');
```

Sequences to share

CREATE [TEMPORARY | TEMP] SEQUENCE name
[INCREMENT [BY] increment]
[MINVALUE minvalue | NO MINVALUE] [MAXVALUE maxvalue | NO MAXVALUE]
[START [WITH] start] [CACHE cache]
[[NO] CYCLE]

```
CREATE SEQUENCE artificial_key START 101;

  CREATE TABLE Dept (deptno INT DEFAULT
  nextval('artificial_key') NOT NULL,...)

INSERT INTO dept VALUES (default, 'I3');
```

“Homework” until next lecture

Play the game SQL Island,
http://www.sql-island.de/

Try the W3Schools SQL Quiz,
http://www.w3schools.com/quiztest/quiztest.asp?qtest=SQL