5. SQL - the Relational Database Language Standard

5.1 Introduction

Most relevant query languages

- development of special languages for relational DBMS, based on tuple relational calculus and relational algebra
- **SQL (Structured Query Language)** is the most popular database language
- also of practical importance: **QBE (Query by Example)**
- the language **Quel (Query Language)** was developed for the DBMS Ingres, did not prevail over SQL

SQL

- developed 1974 at IBM as language of the relational DBMS System R
- SQL can be regarded as a hybrid between an extended relational algebra and the relational calculus. SQL is a language standard now.
- versions: SQL1 (1985), SQL2 (1992, also denoted as SQL92), SQL3 (1999, also denoted as SQL:1999), in this chapter: excerpt from SQL2
Components of SQL

- **data definition language (DDL)**
  - creation and change of the data structures for the three levels of a database (external levels, conceptual level, physical level): definition of relation schemas, deletion of relations, creation of indexes, modification of relation schemas, creation of views
  - specification of integrity constraints
  - fixing of access rights (authorization)

- **data manipulation language (DML)**
  - insertion, change and deletion of data objects
  - interactive formulation of queries

- **embedded DML**
  - embedding of SQL-commands into an all-purpose programming language (host language) like e.g. Fortran, C, C++ or Java

- **transaction control**
  - commands for specifying the begin, abort or end of transactions, in some implementations explicit commands for locking data for concurrency control
5.2 Data definition language (DDL)

Data types

- primarily numbers, strings and date declarations as fundamental data types for attribute domains
- in detail:
  - `char(n)` character string of fixed length $n$, with user specified length $n$, synonym: `character(n)`
  - `varchar(n)` character string of variable length, with user specified maximum length $n$, synonym: `char varying(n)`, `character varying(n)`
  - `int` integer, value of a computer-dependent, finite subset of the whole numbers, synonym: `integer`
  - `smallint` small integer, a computer-dependent subset of the `int`-domain
  - `numeric(z, n)` fixed-point (decimal) number with user specified precision, $z =$ total number of digits, $n =$ number of the $z$ digits to the right of the decimal point, synonym: `decimal(z, n)`
  - `real` floating-point number with computer-dependent precision
<table>
<thead>
<tr>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>double precision</td>
<td>double-precision floating-point number with computer-dependent precision</td>
</tr>
<tr>
<td>float(n)</td>
<td>floating-point number with user specified precision of at least n digits</td>
</tr>
<tr>
<td>bit(n)</td>
<td>bit string of fixed length n</td>
</tr>
<tr>
<td>bit varying(n)</td>
<td>bit string of variable length with user specified maximum length n</td>
</tr>
<tr>
<td>blob</td>
<td>binary large object, byte sequence of variable length up to 4 GB, for the representation of extremely large objects (e.g. multimedia objects, video sequences, geo-objects)</td>
</tr>
<tr>
<td>date</td>
<td>calendar date with year (4 digits), month (2 digits), day (2 digits), format: YYYY-MM-DD</td>
</tr>
<tr>
<td>time</td>
<td>time of day, in hours, minutes, and seconds, format: HH:MM:SS</td>
</tr>
<tr>
<td>time with time zone</td>
<td>time difference to GMT (6 digits)</td>
</tr>
<tr>
<td>timestamp</td>
<td>value containing date and time of date</td>
</tr>
<tr>
<td>interval</td>
<td>relative value which can increment or decrement an absolute value of type date, time or timestamp, year/month-or day/hour-intervals</td>
</tr>
</tbody>
</table>
(restricted) declaration of a domain

- advantage: simple change of a data type for a domain which is used from several attributes in a schema
- form: `create domain < my type> as < type specification >`
  
  example: `create domain string as varchar(256)`

Specification of integrity constraints and default values

- Since SQL allows null values (`null`), an integrity constraint `not null` can be defined, if for a specific attribute a null value is `not` allowed.
- It is recommended to specify this condition for each primary key.
- definition of a default value for an attribute by attaching the clause `default < value >` to the attribute definition
- The default value is inserted into each new tuple, if an explicit value for this attribute is not specified. If a default clause is not defined, the default value is `null`.
- The clause `primary key` specifies one or more attributes that form the primary key of the relation.
- definition of a foreign key by the `foreign key` clause (referential integrity)