4. SQL - the Relational Database Language Standard

4.1 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) [→ chapter 5]
- the language **Quel (Query Language)** was developed for the DBMS Ingres, did not prevail over SQL [→ chapter 5]

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: excerpts 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
4.2 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
- **double precision**
  double-precision floating-point number with computer-dependent precision

- **float**\( (n) \)
  floating-point number with user specified precision of at least \( n \) digits

- **bit**\( (n) \)
  bit string of fixed length \( n \)

- **bit varying**\( (n) \)
  bit string of variable length with user specified maximum length \( n \)

- **blob**
  **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)

- **date**
  calendar date with year (4 digits), month (2 digits), day (2 digits), format: YYYY-MM-DD

- **time**
  time of day, in hours, minutes, and seconds, format: HH:MM:SS

- **time with time zone**
  time difference to GMT (6 digits)

- **timestamp**
  value containing date and time of date

- **interval**
  relative value which can increment or decrement an absolute value of type **date**, **time** or **timestamp**, year/month-or day/hour-intervals
(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)
unique expresses that this attribute is a candidate key. If a candidate key is formed by several attributes $A_1, ..., A_n$, this is specified by the integrity constraint $\text{unique}(A_1, ..., A_n)$.

Creation of a relation schema

- in SQL no relations but tables (duplicates allowed)
- creation of a schema with the aid of the clause

```
create table R(A_1 \text{ } D_1, A_2 \text{ } D_2, ..., A_n \text{ } D_n,
                 [< \text{integrity constraint}_1>, ..., < \text{integrity constraint}_k>])
```

$R$ relation name, $A_i$ name of an attribute in the schema of relation $R$, $D_i$ domain of $A_i$

- in BNF notation:

```
create table <relation name> (<relation comp> [, <relation comp>]*)

<relation comp> ::= <column definition> | <integrity constraint>
<column definition> ::= <attribute name> <type> [<default value> | not null | unique]
<default value> ::= [default <literal> | null]
```

The exact treatment of integrity constraints is discussed later.
integrity constraints

**primary key** \((A_{j1}, ..., A_{jm})\)

The attributes \(A_{j1}, ..., A_{jm}\) form the primary key of \(R\).

example: university schema (with incomplete integrity constraints)

```sql
create table students
    (reg-id int not null,
     name varchar(30) not null,
     sem int,
     **primary key** (reg-id))

create table professors
    (pers-id int not null,
     name varchar(30) not null,
     room int unique,
     rank char(2),
     **primary key** (pers-id))
```
create table assistants
  (pers-id int not null,
   name varchar(30) not null,
   room int unique,
   boss int,
   primary key (pers-id),
   foreign key (boss) references professors(pers-id))

create table lectures
  (id int not null,
   title varchar(30),
   credits int,
   held_by int,
   primary key (id),
   foreign key (held_by) references professors(pers-id))

create table attends
  (reg-id int not null,
   id int not null,
   primary key (reg-id, id),
   foreign key (reg-id) references students(reg-id),
   foreign key (id) references lectures(id))
create table is_precondition_of
(predecessor  int not null,
  successor  int not null,
  primary key (predecessor, successor),
  foreign key (predecessor) references lectures(id),
  foreign key (successor) references lectures(id))

create table tests
(reg-id  int not null,
  id  int not null,
  pers-id  int not null,
  grade  numeric(2,1),
  primary key (reg-id, id, pers-id),
  foreign key (reg-id) references students(reg-id),
  foreign key (id) references lectures(id),
  foreign key (pers-id) references professors(pers-id))