SQL language

Jaroslav Porubän, Miroslav Biňas,
Milan Nosáľ (c) 2011 - 2016

SQL - Structured Query Language

- SQL is a computer language for communicating with DBSM
- **Nonprocedural** (**declarative**) language
  - What we want, not how we want it (not a sequence of steps)
  - DBMS is responsible for retrieving data
- Based on relational calculus

SQL History

- 1970s – Structured English Query Language (SEQUEL or SEQL)
- 1980 – renamed to SQL
- 1986 – SQL-86 ANSI standard
- 1989 – SQL-89
- 1992 – SQL-92, bigger change
  - Data types for date and time
  - Set operations
  - Changes to `ALTER`, `DROP`
- SQL:1999, SQL:2003, SQL:2008, SQL:2011

Database servers

- Oracle
- MySQL
- Microsoft SQL Server
- Mongo DB (NoSQL)
- PostgreSQL
- IBM DB2
- Microsoft Access
- Apache Cassandra (NoSQL)
- SQLite
- Redis (NoSQL)

[1] http://db-engines.com/en/ranking (2016)

SQL parts

- **DDL** (Data Definition Language) – database structure definition language, including changes to structure – `CREATE`, `DROP`, `ALTER`, `TRUNCATE`
- **DML** (Data Manipulation Language) – data manipulation including retrieving data (queries) – `SELECT`, `INSERT`, `UPDATE`, `DELETE`
- **DCL** (Data Control Language) – language part for controlling access to data – `GRANT`, `REVOKE`

Data types

- **Data type** defines possible values in the column (attribute domain)
- Dedicated to describe domain integrity
- Several built-in types for working with
  - **Strings**
  - **Dumbers**
  - **Date** and **time**
  - **Multimedia contents** (`BLOB`)

Basic data types

- `INTEGER`
- `FLOAT` - floating point number
- `NUMBER(n,s)` - number with precision
- `CHAR(n)` - fixed size string
- `VARCHAR(n)` - variable size string
- `CLOB` - large (max 8TB) strings (texts)
- `DATE` - year, month, day, time, minutes, seconds
- `TIMESTAMP` - extends DATE time with fractional seconds

CREATE TABLE

- Creates table
- Requires to analyze the domain first!
- syntax:

```
CREATE TABLE table_name(
    name1 type1 [constraints],
    name2 type2 [constraints],
    ...
    nameN typeN [constraints]
);
```

CREATE TABLE - example

```
CREATE TABLE FBUser (
    username VARCHAR2(30),
    name VARCHAR2(30),
    surname VARCHAR2(40),
    birthday DATE,
    sex CHAR
);
```

DROP TABLE

- Drops (deletes) table
- syntax:

```
DROP TABLE table_name;
```

- example:

```
DROP TABLE FBUser;
```

Preventive table removing

- Removing previous version of table (view) before creating the new one
- Enforces name availability
- Used during development process (testing)
- syntax

```
DROP VIEW name;
CREATE VIEW name AS...;
```

or

```
CREATE OR REPLACE VIEW name
AS...;
```

Recommended steps for table creation

1. Identify columns' data types
2. Identify nullable columns (and those that cannot be null)
3. Identify unique columns
4. Identify primary and foreign keys (and their pairs that express relationships)
5. Identify default values
6. Identify constraints for columns (with respect to the domain)
7. Create table

Showing information about table

- Table description represents:
  - List of table columns (attributes),
  - Their data types, and
  - constraints (if any).
- syntax:

```
DESC table;
DESCRIBE table;
```

Table constraints

- Enforcing constraints during data inserting, updating, and removal
  - Mandatory value – `NOT NULL`
  - Unique values – `UNIQUE`
  - Primary key – `PRIMARY KEY`
  - Foreign key (link to primary key) – `FOREIGN KEY`
  - Custom checking – `CHECK`
  - Default value - `DEFAULT`

Example

```
CREATE TABLE osoba (
  username VARCHAR(30) PRIMARY KEY,
  name VARCHAR(30) NOT NULL,
  surname VARCHAR(40) NOT NULL,
  birthday DATE,
  sex CHAR(1) DEFAULT 'M'
);
```

Comparison of `NOT NULL`, `UNIQUE` a `PRIMARY KEY`

- **`NOT NULL`** column
  - **It is not possible to** insert `NULL` value
  - Values **can be repeated**
- **`UNIQUE`** column
  - `NULL` value is allowed
  - Values **cannot repeat**
- **`PRIMARY KEY`** column
  - `NULL` value is not allowed
  - Values **cannot repeat**

Primary key - Simple key (single column)

```
CREATE TABLE course (
    id INT PRIMARY KEY,
    title VARCHAR(32) NOT NULL
);

CREATE TABLE course (
    id INT,
    Title VARCHAR(32) NOT NULL,
    CONSTRAINT course_pk
        PRIMARY KEY (id)
);
```

Primary keys - Composed key (multiple keys)

```
CREATE TABLE course_enlist (
    id_course INT,
    id_student INT,
    PRIMARY KEY (
        id_course, id_student)
);
```

Foreign keys and referential integrity I.

- **Foreign key** represents relationship (link) between rows (tuples)
  - Table with foreign key: child table
  - Table with primary key: referenced (parent) table
- Keyword: `REFERENCES`
- syntax:

```
REFERENCES table(column)
```

Foreign keys and referential integrity II.

```
CREATE TABLE course_enlist (
  id_course INT
    REFERENCES course(id),
  id_student INT
    REFERENCES student(id),
  PRIMARY KEY (id_course,
    id_student)
);
```

Foreign keys and referential integrity III.

```
CREATE TABLE course_enlist (
  id_course INT,
  id_student INT,
  PRIMARY KEY (id_course,
    id_student),
  FOREIGN KEY(id_course)
    REFERENCES course(id),
  FOREIGN KEY(id_student)
    REFERENCES student(id)
);
```

CHECK constraint

- Checking contraints during insertion and update of table rows; there can be multiple constraints
- example:

```
CREATE TABLE student (
 ...
 class INT NOT NULL
 CHECK(class>=1 AND class<=5)

 -- CONSTRAINT class_constraint
 -- CHECK(class>=1
 --       AND class<=5)
);
```

ALTER TABLE

- Changes structure of existing table
  - **ADD COLUMN**,
  - **DROP COLUMN**, or
  - **MODIFY COLUMN**
- syntax:

```
ALTER TABLE table
operation column [type]
[constraints];
```

ALTER TABLE - example

- Add a column:

```
ALTER TABLE student
ADD sex CHAR;
```

- Remove column:

```
ALTER TABLE student
DROP COLUMN sex;
```

- Change existing column:

```
ALTER TABLE student
MODIFY name VARCHAR(20);
```

Questions?