# SQL: DML, DDL

# SQL

- Structured Query Language, basically a 30-year-old standard
- so far we have only talked about the SELECT statement, which is used to retrieve data from the database
- The SQL standard also includes
  - **DDL:** Data definition language (CREATE TABLE, ALTER TABLE, …)
  - **DML:** Data manipulation language (INSERT / UPDATE / DELETE)
  - **DCL:** Data control language (GRANT / REVOKE)
- the SQL standard defines the "basic" functionality that is more or less supported by every database system
- However, database systems often extend the standard with additional features
  - e.g. INSERT .... ON DUPLICATE KEY ... (MySQL), specific field types, functions, etc.

# DML

- **inserting records (rows):**
- INSERT INTO <table-name> (<col1>,<col2>,…) VALUES (…),(…)
- INSERT INTO <table-name> (<col1>,<col2>,…) SELECT … FROM …
  - http://www.postgresql.org/docs/current/static/sql-insert.html
- **Edit rows:**
- UPDATE <table-name> SET <col1>=<val1>, <col2>=<val2> WHERE <condition>
  - http://www.postgresql.org/docs/current/static/sql-update.html
- **Deleting rows:**
- DELETE FROM <table> WHERE <condition>
  - http://www.postgresql.org/docs/current/static/sql-delete.html

# INSERT

*purchase(id, buyer, seller, product, store)*

- INSERT INTO purchase (buyer, seller, product, store)
  VALUES ('Joe', 'Fred', 'espresso-machine', 'The Sharper')
- INSERT INTO purchase
  (1, 'Joe', 'Fred', 'espresso-machine', 'The Sharper'),
  (2, 'Joe', 'John', 'smart-tv', 'unknown');
- INSERT INTO purchase (buyer, seller, product, store)
  SELECT order_to, order_from, product, store FROM orders
  WHERE processed='1'

# UPDATE

- UPDATE product
  SET price = price/2
  WHERE productid > 20

- UPDATE product
  SET price = price / 2
  WHERE product.name IN
  (SELECT product
  FROM purchase
  WHERE date = ‘2015-11-09’);

# DELETE

- DELETE FROM purchase
  WHERE seller = ‘joe’ and product = ‘Brooklyn Bridge’

# DDL

- CREATE DATABASE <db-name>
- DROP DATABASE [IF EXISTS] <db-name>
  - Your cvika.dcs.fmph.uniba.sk account does not have permission to create and delete databases
- CREATE TABLE <table-name> (…)
  - Create a new table in the database
  - in parentheses are the names of the fields and their types (or other parameters)
- DROP TABLE [IF EXISTS] <table-name>
  - Deleting a table from the database
- ALTER TABLE <tablename> ADD/MODIFY/DROP [column] <colum name> …
  - change of attributes (columns), similarly indices are added / removed

# CREATE TABLE

```
CREATE TABLE person (
        personid SERIAL NOT NULL,
        firstname VARCHAR(50),
        lastname VARCHAR(50) NOT NULL,
        date_of_birth DATE,
        login_count INTEGER NOT NULL DEFAULT 0
);
```

- many different types of fields:
  - http://www.postgresql.org/docs/current/static/datatype.html
  - http://www.postgresql.org/docs/current/static/sql-createtable.html

# Data types

| **Text:** | |
|---|---|
| varchar(n) | string, variable-length with limit |
| text | string, variable unlimited length |
| **Number:** | |
| int/smallint/bigint | Integer number |
| real, numeric, double precision | Real number |
| decimal | Fixed number of decimals |
| **Date and time:** | |
| date | calendar date (year, month, day) |
| time | time of day (no time zone) |
| timestamp [with timezone] | date and time |

# Special data types

- serial / auto_increment – Automatic line numbering (suitable as primary key)
- JSON / JSONB – JavaScript Object Notation
  - {
    fistname: ‘Laco’,
    lastname: ‘Novák’,
    address: {
      street: ‘Štúrova’,
      number:’2/B’,
    }
    }
  - sometimes you can't say in advance what data / columns you will need, or they are variable (e.g. several individual items for each row)
  - rapid development, data taken from a NoSQL database or RESTful API, ...
  - http://www.postgresql.org/docs/current/static/datatype-json.html

# ALTER TABLE

- Changing the definition of a table without losing content
  - http://www.postgresql.org/docs/current/static/sql-altertable.html
- Add columns:
  - `ALTER TABLE <table-name> ADD COLUMN <colum-name> <type>`
- Delete columns
  - `ALTER TABLE <table-name> DROP COLUMN <colum-name>`
- Change columns
  - `ALTER TABLE <table-name> ALTER COLUMN <column-name> TYPE <type> ...`
- many other features, see documentation

# Indexes

- allow faster search for records
  - faster compared to the so-called full table scan (searching all rows of the table)
- for accelerated search, the system keeps a separate index
  - faster search --- O(1) or O(log n) versus O(n)
  - maybe slower insertion / adjustment --- O(log n) vs. O(1)
- Primary Key
  - A column or set of columns that uniquely identifies the other columns of a table
  - unique index
- Two basic types of indices:
  - Hash Tables
  - Search Trees (B/B+)
- http://www.postgresql.org/docs/current/static/indexes.html

# Indexes

- reation:
- CREATE INDEX <name> ON <table> USING btree (<columns>)
- CREATE TABLE <table> (
  firstname TEXT, lastname TEXT,
  UNIQUE (lastname, firstname),
  )
- The index can also be created on multiple columns and on an expression created from the values in individual columns, e.g. *lower(firstname)*

# Indexes

- The type of index also determines how we can search
- HASH indexes (USING hash) – works only for “=“
- B-Trees (USING btree) – works for “<“, “>”,”<=“, “>=“, “=“
  - we can search for prefixes in strings (e.g. <column> LIKE "prefix%" works)
- special indexes (GIST in postgres) allow you to search for e.g. the nearest neighbor on the map, etc.
- http://www.postgresql.org/docs/current/static/indexes-types.html

# Encodings and collations

- Incompatible string encodings are a common source of problems
- encoding – how individual characters of text are binary encoded
  - **UTF8**, WIN1250, LATIN1,…
  - If we want to compare strings in different encodings, we must first "recode" them into a uniform format
    - it is not always possible (different character sets)
  - beware of different types of client coding (e.g. console, .sql file) and server
    - SET CLIENT ENCODING utf8;
  - nowadays it is probably appropriate to use UTF8 everywhere
  - attention: utf8 in MySQL is not utf8, you need to use utf8mb4...
    https://medium.com/@adamhooper/in-mysql-never-use-utf8-use-utf8mb4-11761243e434

# Encodings and collations

- collation
  - Rules for arranging strings
  - Character classification
    - What is a letter?
    - what is the UPPER-CASE equivalent of the letter?
  - In some cases (e.g. MySQL) we can tell if we are case-sensitive when searching, etc.
    - CITEXT module in Postgres

# Encodings and collations

- Inconsistency between database systems, sometimes chaos,...
- postgres:
-
```
CREATE DATABASE <db-name>
        ENCODING='UTF8'
        LC_COLLATE="sk_SK.UTF8"
        LC_CTYPE="sk_SK.UTF8"

 CREATE TABLE <table-name> (
        description TEXT COLLATE 'sk_SK.UTF8'
 )
```
- http://www.postgresql.org/docs/current/static/multibyte.html
- http://www.postgresql.org/docs/current/static/collation.html