Lecture 3.1. DDL

What is SQL?

> SQL (Structured Query Language) - standard language for creating databases. Managing and manipulating data in relational DBMS (MySQL, PostgreSQL, Oracle, etc.)

Used for almost everything you do with RDBMS: queries, inserts, updates, schema changes, permissions, transactions, etc

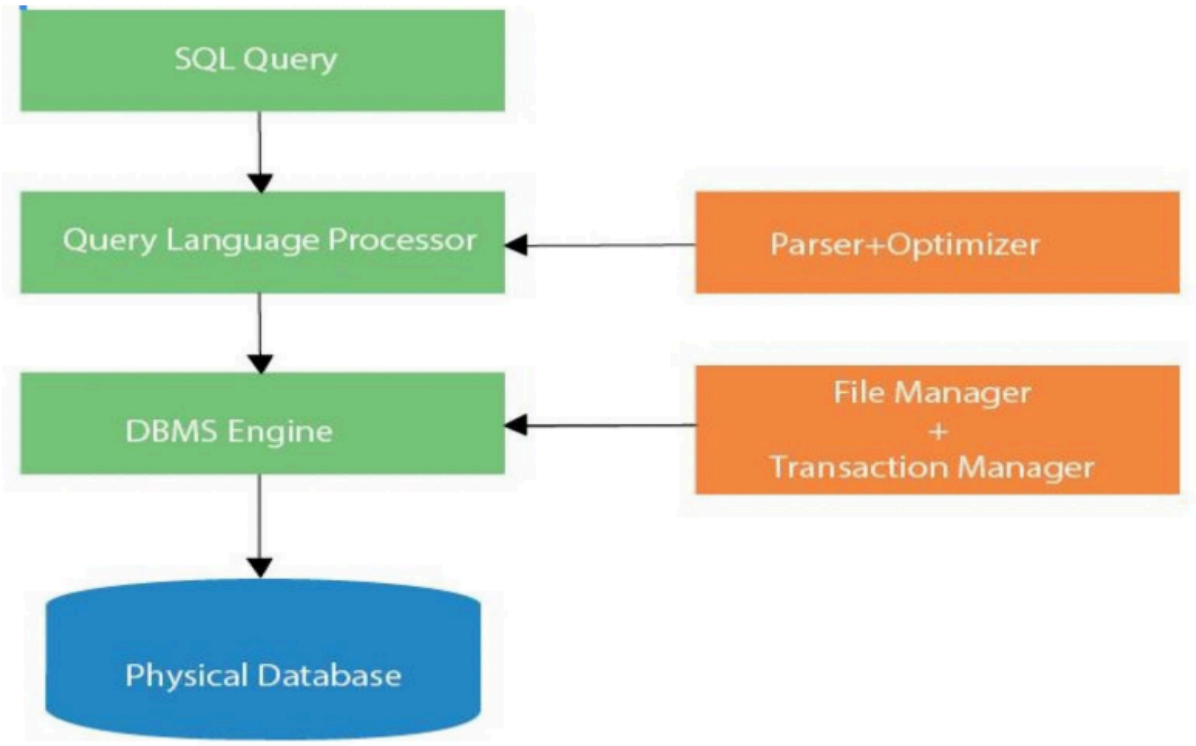
---

Basic properties / Rules of SQL

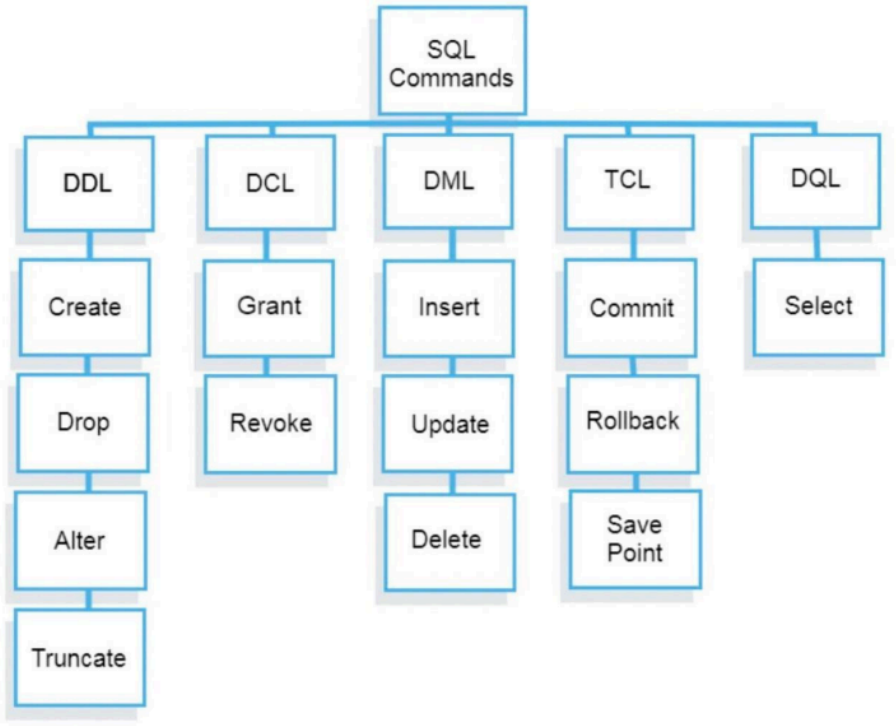
- **Not case-sensetive**:
  - `select`, `SELECT`, `SeLeCt` — all the same
  - Convention: keywords in UPPERCASE, identifiers in lowercase
- A single SQL statement can span multiple lines
- Almost all database actions can be done by sone SQL statement
- The language is based on relaational algebra and tuple relational calculus

---

SQL Process


SQL Query
Query Language Processor
Parser+Optimizer
DBMS Engine
File Manager
+
Transaction Manager
Physical Database

Types of SQL



Data Definition Language (DDL)

Defenition:

> DDL is the set of SQL commands used to define, modify, and remove database structures (tables, constraints, indexes, etc)

DDL is usally executed:

- in SQL console/browser
- or inside stored procedures

CREATE Statement

```
CREATE TABLE table_name(
    column_name TYPE column_constraint,
```

```
    table_constraint table_constraint
);
```

PostgreSQL column constraints:

- NOT NULL - the value of the column cannot be NULL
- UNIQUE
  - All non-NULL values in the column must be distinct
  - PostgreSQL only allow multiple NULLs in a UNIQUE column (each NULL is treated as 'different')
- PRIMARY KEY
  - Combination of NOT NULL + UNIQUE
  - Each table usually has one primary key
  - For single-column PK you can define it at column level
- REFERENCES
  - Defines a foreign key: value must exist in a column of another table.
  - Syntax: REFERENCES other_table(other_column)

PostgreSQL table constraints

- UNIQUE (col1, col2, ...)
  - Enforces uniqueness across combination of columns
- PRIMARY KEY (col1, col2, ...)
  - Composite primary key
- FOREIGN KEY (col) REFERENCES other_table(other_col)
  - Table-level FK, often used for composite or named constraints

ALTER statement

```
ALTER TABLE table_name action;
```

Purpose: change existing table structure

You can:

- **Add / drop / rename a column**
- **Change column type**
- **Set / drop default**
- **Add / drop constraints**
- **Rename a table**

DROP statement

```
DROP TABLE [IF EXISTS] table_name [CASCADE|RESTRICT]
```

Purpose: comletely remove a table definition and all its data

DROP is destructive:

- Deletes table structure
- Deletes data
- Delete constraints & indexes on that table

Options:

- CASCADE - also drop objects that depend on this table (views, FKs, etc)
- RESTRICT - refuse to drop if there are dependent objects

TRUNCATE statements

```
TRUNCATE TABLE table_name
```

To truncate multiple tables:

```
TRUNCATE TABLE table_name1, table_name2, ...;
```

Purpose: delete all data from a table that has a lot of data

TRUNCATE:

- is a DDL operation (in many DBMS, implicit COMMIT before/after).
- usually **cannot be rolled back** once committed (depends on DB & transaction mode).
- resets identity/serial counters in some DBs.