Lecture 3

SQL. Types of SQL. Data Definition Language. Managing Tables. Constraints.

Senior Lecturer: Sholpan Saimassayeva

SQL. Types of SQL. Data Definition Language. Managing Tables. Constraints:

PART I. Structured Query Language. History, Advantages, Types of SQL.
PART II. Data Definition Language. Managing Tables. Constraints.

PART I. Structured Query Language. History, Advantages, Types of SQL.

What is SQL?

Structured query language (SQL) is a standard language for database creation and manipulation.

SQL is the standard language for database management. All the RDBMS systems like MySQL, MS Access, Oracle, Sybase, Postgres, and SQL Server use SQL as their standard database language. SQL programming language uses various commands for different operations.

History:

- 1970 - Dr. E. F. "Ted" of IBM is known as the father of relational databases. He described a relational model for databases.
- 1974 - Structured Query Language appeared.
- 1978 - IBM worked to develop Codd's ideas and released a product named System/R.
- 1986 - IBM developed the first prototype of relational database and standardized by ANSI. The first relational database was released by Relational Software and its later becoming Oracle.

What are the SQL?

SQL follows the following rules:

•Structure query language is not case sensitive. Generally, keywords of SQL are written in uppercase.

•Statements of SQL are dependent on text lines. We can

use a single SQL statement on one or multiple text line.

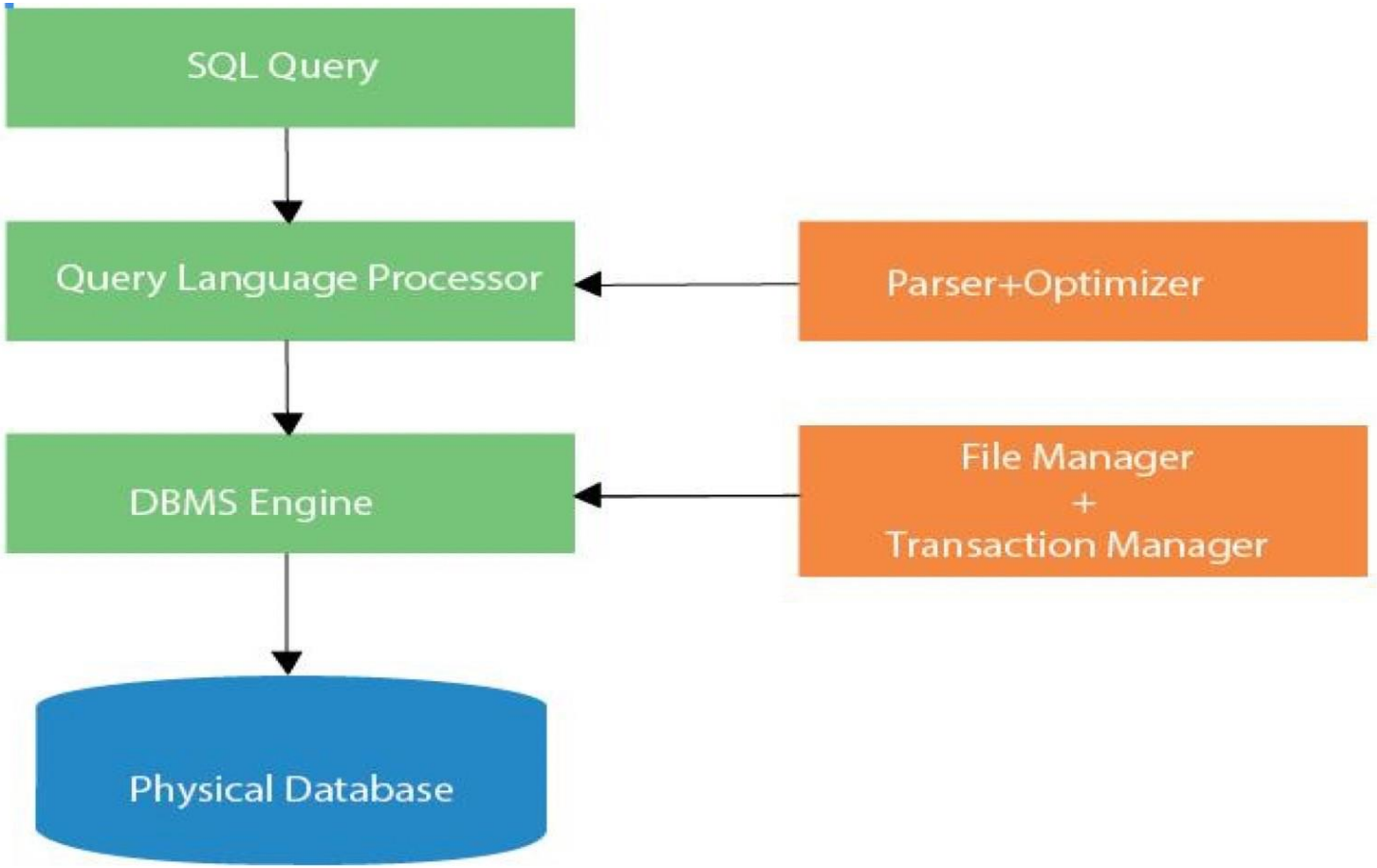
•Using the SQL statements, you can perform most of the actions in a database.

• SQL depends on tuple relational calculus and relational algebra.

What is SQL Process?

- When an SQL command is executing for any RDBMS, then the system figure out the best way to carry out the request and the SQL engine determines that how to interpret the task.
- In the process, various components are included. These components can be optimization Engine, Query engine, Query dispatcher, classic, etc.
- All the non-SQL queries are handled by the classic query engine, but SQL query engine won't handle logical files.

What is SQL Process?



Why Use SQL?

- It helps users to access data in the RDBMS system.
- It helps you to describe the data.
- It allows you to define the data in a database and manipulate that specific data.
- With the help of SQL commands in DBMS, you can create and drop databases and tables.
- SQL offers you to use the function in a database, create a view, and stored procedure.
- You can set permissions on tables, procedures, and views.

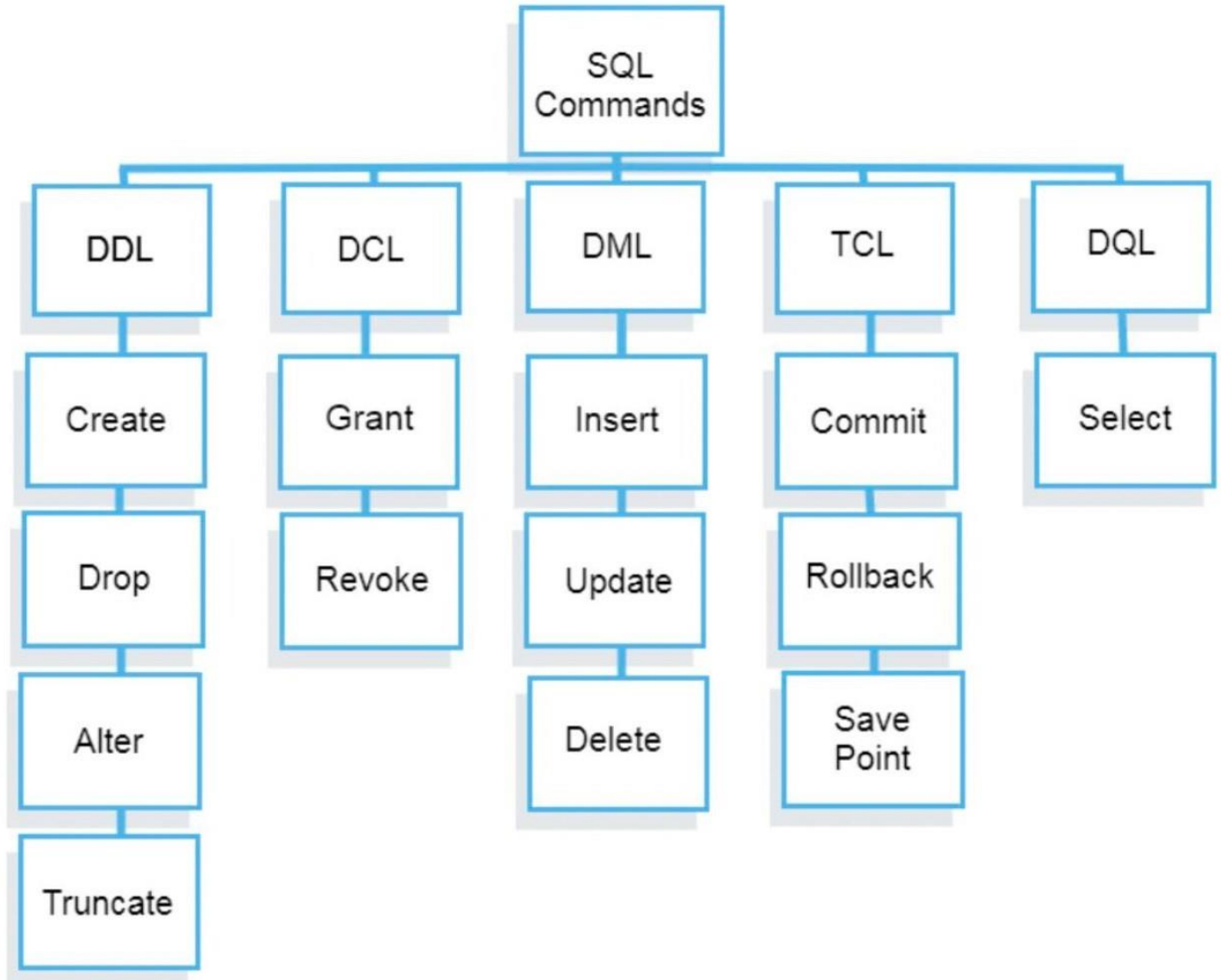
What are the Advantages of SQL?

- High speed
- Well defined standards
- Portability
- Interactive language
- Multiple data view

Types of SQL

Here are five types of widely used SQL queries:

- Data Definition Language (DDL)
- Data Manipulation Language (DML)
- Data Control Language(DCL)
- Transaction Control Language(TCL)
- Data Query Language (DQL)


SQL Commands
DDL
DCL
DML
TCL
DQL
Create
Drop
Alter
Truncate
Grant
Revoke
Insert
Update
Delete
Commit
Rollback
Save Point
Select

PART II. Data Definition Language. Managing Tables. Constraints.

Data Definition Language

Data definition language (DDL) refers to the set of SQL commands that can create and manipulate the structures of a database.

Using DDL statements, you can perform powerful commands in your database such as creating, modifying, and dropping objects. DDL commands are usually executed in a SQL browser or stored procedure.

Types of DDL commands:

- CREATE
- DROP
- ALTER
- TRUNCATE

The CREATE Statement

To create a new table in PostgreSQL, you use the **CREATE TABLE** statement. The following illustrates the syntax of the **CREATE TABLE** 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** – the value of the column must be unique across the whole table. However, the column can have many NULL values because PostgreSQL treats each NULL value to be unique. Notice that SQL standard only allows one NULL value in the column that has the UNIQUE constraint.
- **PRIMARY KEY** – this constraint is the combination of NOT NULL and UNIQUE constraints. You can define one column as PRIMARY KEY by using column-level constraint. In case the primary key contains multiple columns, you must use the table-level constraint.
- **REFERENCES** – constrains the value of the column that exists in a column in another table. You use REFERENCES to define the foreign key constraint.

PostgreSQL table constraints

- **UNIQUE** (column_list) – to force the value stored in the columns listed inside the parentheses to be unique.
- **PRIMARY KEY** (column_list) – to define the primary key that consists of multiple columns.
- **REFERENCES** - to constrain the value stored in the column that must exist in a column in another table.

PostgreSQL CREATE TABLE example

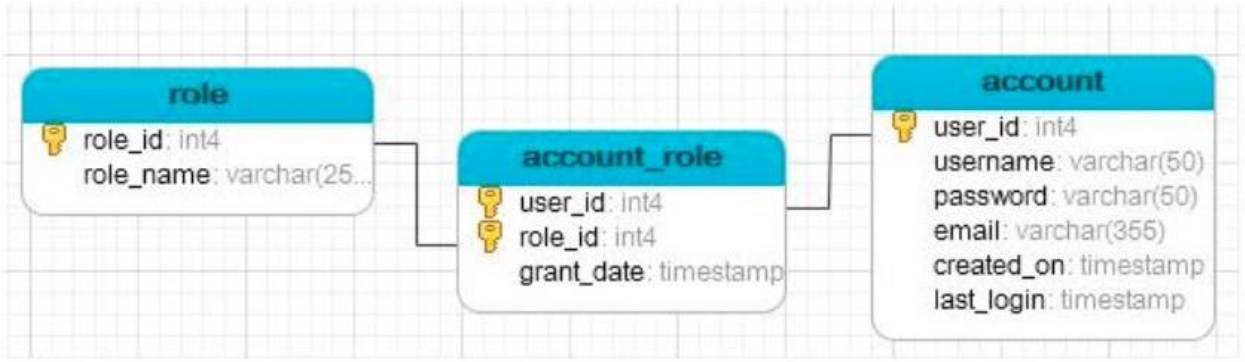
```
CREATE TABLE account(
user_id serial PRIMARY KEY,
username VARCHAR (50) UNIQUE NOT NULL,
password VARCHAR (50) NOT NULL,
email VARCHAR (355) UNIQUE NOT NULL,
created_on TIMESTAMP NOT NULL,
last_login TIMESTAMP
);
```

PostgreSQL CREATE TABLE example

```
CREATE TABLE role(
role_id serial PRIMARY KEY,
role_name VARCHAR (255) UNIQUE NOT
NULL
);
```

```
CREATE TABLE account_role (
user_id integer NOT NULL,
role_id integer NOT NULL,
grant_date timestamp,
FOREIGN KEY (role_id) REFERENCES role (role_id)
);
```

PostgreSQL CREATE TABLE example



If you have forgotten to add Primary and Foreign keys, you can modify your table using ALTER TABLE command:

```
ALTER TABLE account_role
ADD CONSTRAINT constraint_fkey
FOREIGN KEY (user_id) REFERENCES account (user_id);
```

The ALTER Statement

To change the existing table structure, you use PostgreSQL statement. The syntax of the **ALTER TABLE** is as follows:

```
ALTER TABLE table_name action;
```

Example

```
ALTER TABLE account DROP COLUMN email;
```

PostgreSQL provides many actions that allow you to:

1)Add a column, drop a column, rename a column, or change a column’s datatype;

2) Set a default value for the column;

3) Rename a table.

The following illustrates the ALTER TABLE statement variants:

1) To add a new column:

```
ALTER TABLE table_name ADD COLUMN new_column_name TYPE;
```

2)To remove an existing column:

```
ALTER TABLE table_name DROP COLUMN column_name;
```

3)To rename an existing column:

```
ALTER TABLE table_name RENAME COLUMN column_name TO new_column_name;
```

4)To change the NOT NULL constraint, you use ALTER TABLE ALTER COLUMN statement:

```
ALTER TABLE table_name ALTER COLUMN column_name [SET NOT NULL| DROP
NOT NULL];
```

The following illustrates the ALTER TABLE statement variants:

5)To add a constraint:

```
ALTER TABLE table_name ADD CONSTRAINT constraint_name
constraint_definition;
```

6)To rename a table:

```
ALTER TABLE table_name RENAME TO new_table_name;
```

7)To change the data type of the column to VARCHAR (for example), you use the following statement:

```
ALTER TABLE table_name ALTER COLUMN column_name TYPE VARCHAR;
```

The DROP Statement

The DROP statement:

drops, removes, deletes, cancels, blows away, and/or destroys the object it is dropping.

The syntax:

```
DROP TABLE teams;
```

PostgreSQL DROP TABLE syntax:

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

The TRUNCATE Statement

To remove all data from a table, you use the DELETE statement. However, when you use the DELETE statement to delete all data from a table that has a lot of data, it is not efficient. In this case, you need to use the TRUNCATE TABLE statement:

```
TRUNCATE TABLE table_name;
```

Remove all data from multiple tables

To remove all data from multiple tables at once, you separate each table by a comma (,) as follows:

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