Databases Design. Introduction to SQL

LECTURE 6

# SQL.
# Data Manipulation Language

# SQL

- **SQL (Structured Query Language)** is a special-purpose programming language designed for managing data held in a relational database management system (RDBMS).

- SQL keywords are NOT case sensitive: select is the same as SELECT
- All SQL queries must end with a semicolon “;”
- Semicolon is the standard way to separate each SQL statement in DBMS that allow more than one SQL statement to be executed in the same call.

# SQL Comments

- Comments are used to explain sections of SQL statements, or to prevent execution of SQL statements.
- Single line comments start with --.
- Any text between -- and the end of the line will be ignored (will not be executed).

```
--Select all:
SELECT * FROM Students;
```

# SQL Comments

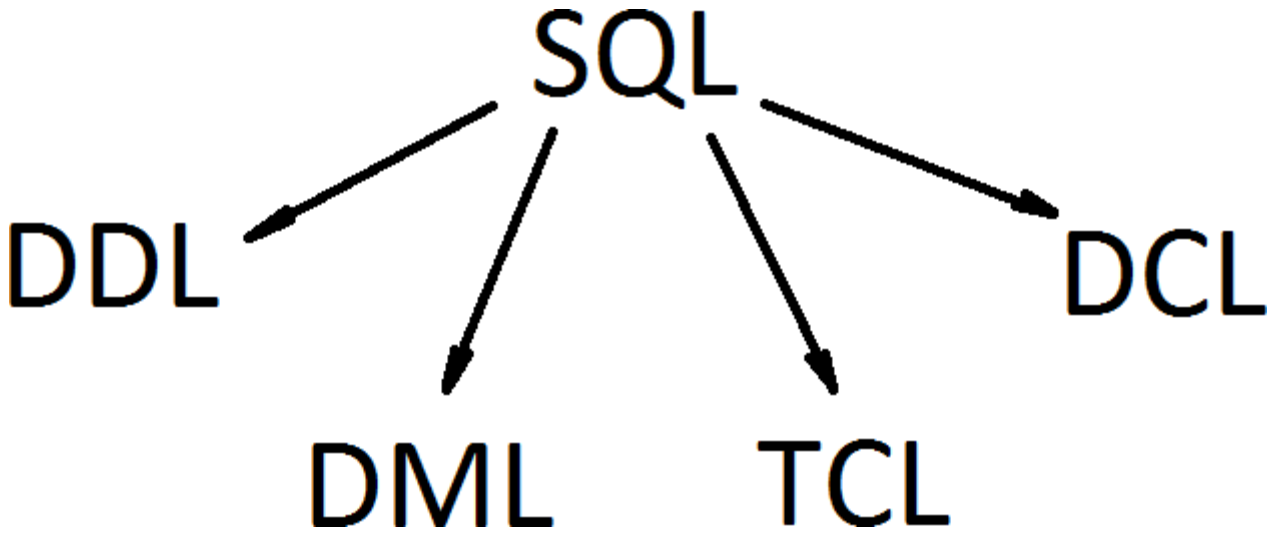
- Multi-line comments start with /* and end with */.
- Any text between /* and */ will be ignored.

```
/*Select all the columns of all the records
in the Students table:*/
SELECT * FROM  Students;
```

- To ignore just a part of a statement, also use the /* */ comment.

```
SELECT lname, /*fname,*/ gpa
FROM  Students;
```

# SQL Structure




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

# Last lecture

**Data Definition Language (DDL)** defines constructs that structure the data in the database.

DDL statements:

- CREATE DB
- CREATE TABLE
- ALTER TABLE
- DROP TABLE

# Data Manipulation Language

**Data Manipulation Language (DML)** is a sublanguage of SQL and it's used for selecting, inserting, deleting and updating data in a database.

DML statements:

- INSERT
- UPDATE
- DELETE
- SELECT

# DDL vs. DML

When a table is created (with DDL statements), it contains no data (data - DML).

| stud_id | f_name | l_name | bdate | group_id |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |

# INSERT

INSERT statement is used to insert new records in a table.

Syntax:

```
INSERT INTO table_name VALUES (values);
```

- In SQL all strings and dates must be single-quoted.
- The data values are listed in the order in which the columns appear in the table, separated by commas.

# INSERT: example 1

To insert a record in the Students table, we write:

```
INSERT INTO Students
VALUES (1, 'Firstname1', 'LastName1',
                                    '31.12.1994',1);
```

So, we would be incorrect in writing:

```
INSERT INTO Students
VALUES ('Firstname1', 'LastName1', 1,
                                    '31.12.1994', 1);
```

# INSERT: example 2

It’s possible to insert a subset of the data in a table. We can tell which columns we would like to insert:

```
INSERT INTO table_name (column(s))
                                VALUES (values);
```

Insert a new record into the Students table for a student with id=2 and no birthdate:

```
INSERT INTO Students (stud_id, f_name,
                                l_name, group_id)
VALUES (2, ‘Firstname2’, ‘Lastname2’, 1);
```

# INSERT: example 3

Data is conceptually inserted one row at a time. However you can insert multiple rows in a single command:

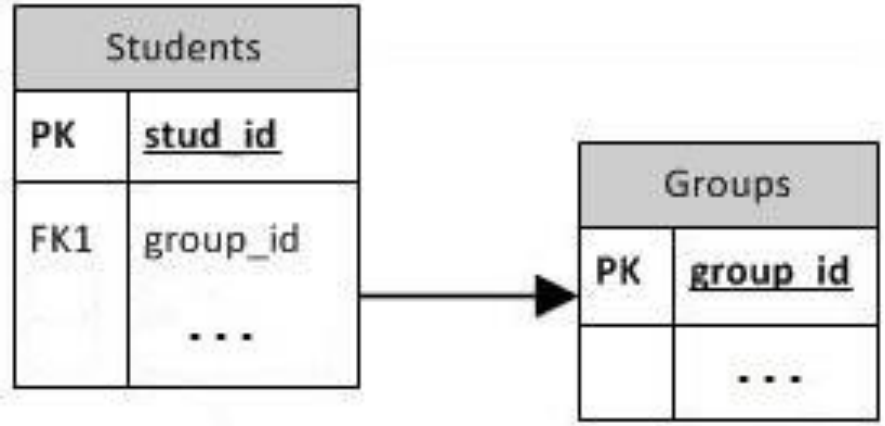
```
INSERT INTO Students (stud_id, f_name,
                                    l_name, group_id)
VALUES (1, ‘FirstName1’, ‘LastName1’, 1),
          (2, ‘FirstName2’, ‘LastName2’, 1);
```

# Referential Integrity (CREATE TABLE)

- Condition for using referential integrity:

  The table that the FK references must have been created before the foreign keys are defined

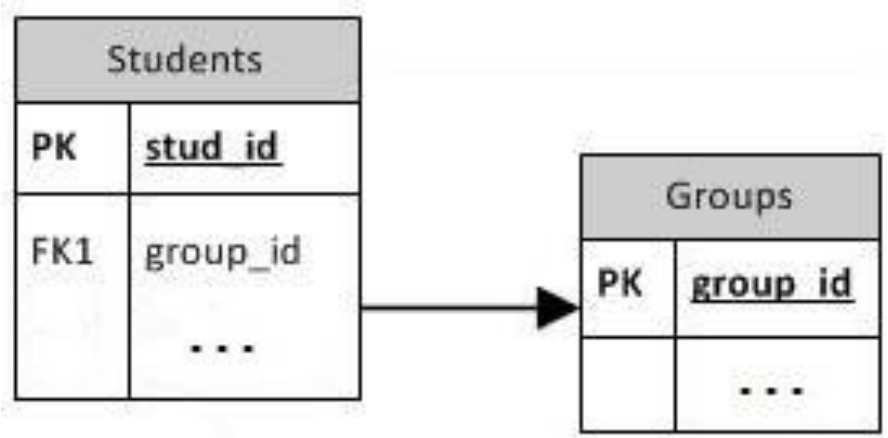
  Ex., the Groups table must be created first, and then the Students table (with its FK) may be created.



# Referential Integrity (CREATE TABLE)

```
CREATE TABLE Groups(
group_id  int,
group_name varchar(15),
PRIMARY KEY (group_id));

CREATE TABLE Students(
stud_id int,
first_name varchar(20),
last_name varchar(20),
bdate date,
group_id int,
PRIMARY KEY (stud_id),
FOREIGN KEY (group_id) REFERENCES Groups(group_id));
```



# Referential Integrity (INSERT)

Rows can’t be inserted in the Students table without a matching rows in the Groups table. This record is unavailable while the Groups table is empty:

```
INSERT INTO Students
VALUES (1, ‘FirstName1’, ‘LastName1’,’31.12.1994’,1);
```

*ERROR: insert or update on table “students" violates foreign key constraint “students_group_id_fkey"*
*DETAIL: Key (group_id)=(1) is not present in table “groups".*

# UPDATE

SQL provides the UPDATE statement to change values in tables.

Syntax:

```
UPDATE table_name
SET column = value [, column = value, …]
[WHERE condition] ;
```

- The condition in the WHERE portion of the UPDATE statement is known as a **selection condition**.
- A selection condition is similar to if-statements in a traditional programming language.

# UPDATE: example 1

Suppose we have a student with stud_id=2 and group_id=1 that needs to be updated to group_id=2.

We would write this as

```
UPDATE Students
SET group_id=2
WHERE stud_id=2;
```

# UPDATE: examples 2, 3

The SQL command to change the name of the student with stud_id=1 to Aigul:

```
UPDATE Students
SET f_name='Aigul'
WHERE stud_id=1;
```

The SQL command to change the birthdate of the student with stud_id=2 to 15.10.1994:

```
UPDATE Students
SET bdate='15.10.1994'
WHERE stud_id=2;
```

# UPDATE: example 4

- Each column can be updated separately; the other columns are not affected.
- You can update individual rows, all the rows in a table, or a subset of all rows.
- For example, this command updates all students that have group_id=1 to have group_id=2:

```
UPDATE Students
SET group_id=2
WHERE group_id=1;
```

# UPDATE: example 5

- The expression for the new value can refer to the existing value(s) in the row.
- If you want to raise the price of all products by 10% you could use:

```
UPDATE Products SET price = price*1.10;
```

- We also left out the WHERE clause. If it is omitted, it means that all rows in the table are updated. If it is present, only those rows that match the WHERE condition are updated.

# DELETE: example 1

SQL provides the DELETE statement to delete data from tables.

Syntax:

```
DELETE FROM table_name
[WHERE condition];
```

# DELETE: example 2

- To delete all rows in the Students table:

```
DELETE FROM Students;
```

- To delete the student with stud_id=2:

```
DELETE FROM Students
 WHERE stud_id=2;
```

# DELETE: example 3

- To remove all rows from the Products table that have a price of 10:

```
DELETE FROM Products
WHERE price=10;
```

# Next lecture: SELECT

SQL allows to query data with SELECT statement.

Syntax:

```
SELECT attribute(s)
FROM table(s);
```

# Books


- **Connolly, Thomas M. Database Systems**: A Practical Approach to Design, Implementation, and Management / Thomas M. Connolly, Carolyn E. Begg.- United States of America: Pearson Education

- **Garcia-Molina, H. Database system**: The Complete Book / Hector Garcia-Molina.- United States of America: Pearson Prentice Hall

- **Sharma, N. Database Fundamentals**: A book for the community by the community / Neeraj Sharma, Liviu Perniu.- Canada

- www.postgresql.org/docs/manuals/

# Question

**Third normal form is based on the concept of …**

- Transitive dependency
- Partial dependency
- Foreign dependency
- None of the given

# Question

**A table in 1NF in which the Primary key consists of two of its three attributes:**

- Always violates 2NF
- Never violates 2NF
- May violate 2NF
- None

# Question

**A table has fields F1, F2, F3, F4 and F5 with the following functional dependencies:**

**F1 -> F3,**

**F2 -> F4,**

**(F1, F2) -> F5.**

**In terms of normalization, this table is in …**

- 1NF
- 2NF
- 3NF
- UNF