Database I

(60012301-1)

Lecture 7: Structured Query Language (SQL) & Data Definition Language

Dr. Obead Alhadreti

Outline

- SQL
- Data Definition Language
  - CREATE
  - ALTER
  - DROP
  - Constraints

Database & DBMS

- Database: is a collection of related data.
- DBMS (**Database Management Systems**): A software package/system is used to facilitate the creation and maintenance of a computerized database.
- **Examples**:

- We will use MySQL as a DBMS in this course.

SQL

- SQL (**Structured Query Language):** is a standard language for storing, manipulating, and retrieving data in relational databases.
- It is designed over relational algebra and tuple relational calculus.
- Originally, SQL was called SEQUEL (Structured English QUEry Language) and was designed and implemented at IBM Research in 1974 and first released in 1986.
- SQL comes as a package with all major distributions of *relational* databases management systems (RDBMS).

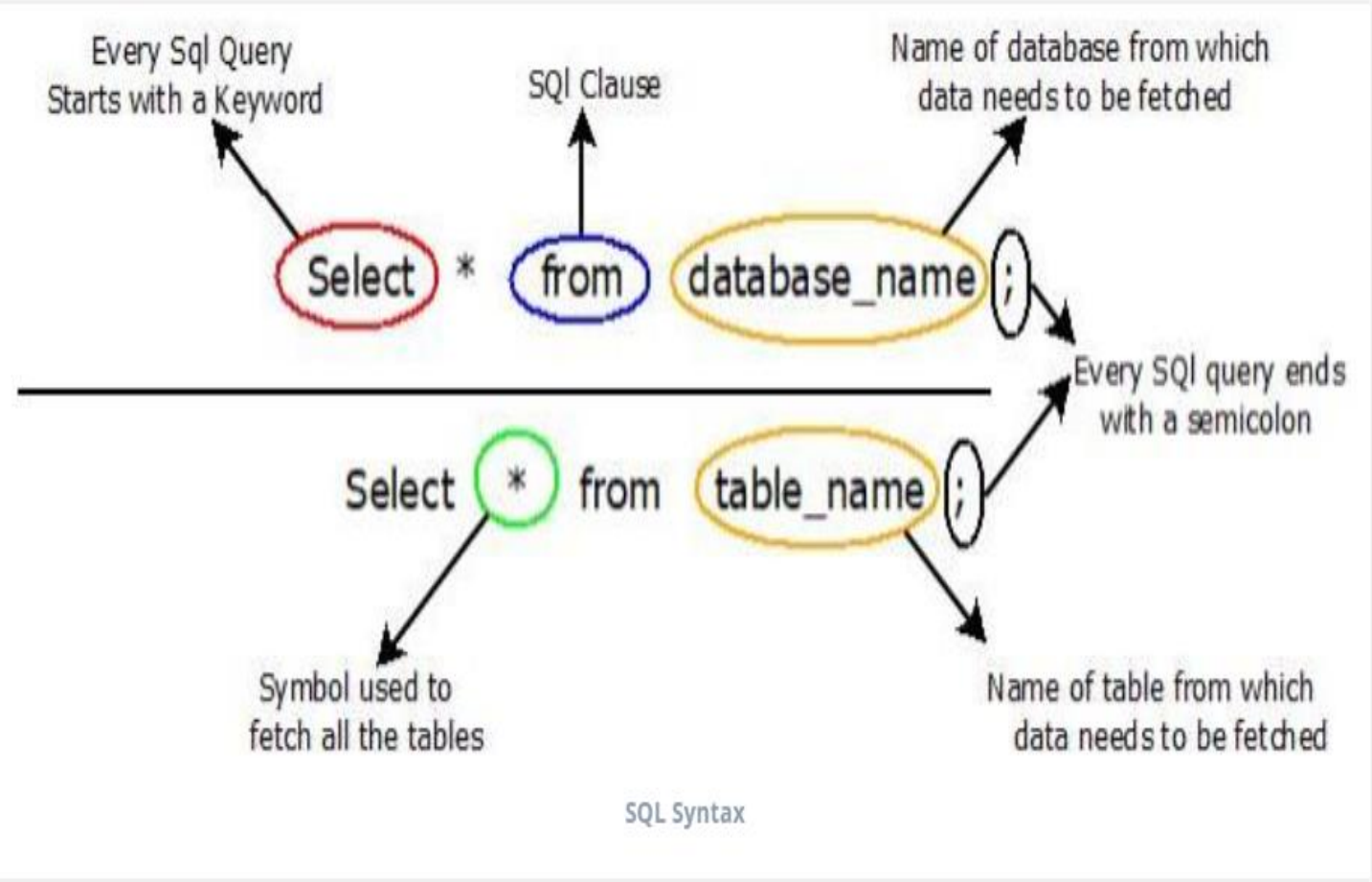
SQL Syntax

- SQL has its own unique syntax.

- SQL syntax is basically the "Structure of Statements" used while generating the queries:
  - All SQL commands ends with a semicolon ;
  - SQL commands are case insensitive.
  - While writing queries, table name and database name must match with original database and tables created.
  - All SQL commands starts with a keyword.

- SQL Keywords: Keywords such as Create, Select, Insert, Drop etc. have specific functionalities in SQL.

SQL Syntax


Every Sql Query
Starts with a Keyword
SQl Clause
Name of database from which
data needs to be fetched
Select
*
from
database_name
;
Every SQl query ends
with a semicolon
Select
*
from
table_name
;
Symbol used to
fetch all the tables
Name of table from which
data needs to be fetched
SQL Syntax

Types of SQL Commands

- **SQL Commands**: Basic SQL statements for storing, retrieving, and manipulating data in a relational database.

- There are five types of SQL commands:

1. **DDL (Data Definition Language).**
2. **DML (Data Manipulation Language).**
3. **DQL (Data Query Language).**
4. **DCL (Data Control Language).**
5. **TCL (Transaction Control Language).**

1. Data Definition Language

- A **data definition language** (**DDL**) consists of the SQL commands that can be used to create and modify the structure of **database objects** in a **database**.

- These **database objects** include views, schemas, tables, indexes, etc.

The CREATE DATABASE Command

- CREATE DATABASE command is used to create a new SQL database.

- Syntax:

```
CREATE DATABASE databasename;
```

- Example:
  The following SQL statement creates a database called "testDB":

```
CREATE DATABASE testDB;
```

The CREATE TABLE Command

- CREATE TABLE Command is used to specify a new table by giving it a name and specifying its attributes and initial constraints.
- The attributes are specified first.
- Each attribute is given a name, a data type to specify its domain of values.

The CREATE TABLE Command

- Syntax:

```
CREATE TABLE table_name (
 column1  datatype,
 column2 datatype,
 column3 datatype,
....
 );
```

- Example:

```
CREATE TABLE Persons (
PersonID int,
LastName varchar(255),
FirstName varchar(255),
Address varchar(255),
);
```

SQL Data Types

- The data type of an attribute column defines what value the column can hold: integer, character, date and time, and so on.

- **Data types** might have *different* names in *different* **DBMS**. And even if the name is the same, the size and other details may be different.

MySQL Data Types

1. String Data Types: most common types include:

Data type	Description
CHAR (size)	A FIXED length string (can contain letters, numbers, and special characters). The *size* parameter specifies the column length in characters - can be from 0 to 255. Default is 1
VARCHAR (size)	A VARIABLE length string (can contain letters, numbers, and special characters). The *size* parameter specifies the maximum column length in characters - can be from 0 to 65,535
MEDIUMTEXT	Holds a string with a maximum length of 16,777,215 characters
LONGTEXT	Holds a string with a maximum length of 4,294,967,295 characters

- Be Consistent in the length. For example use:
  VARCHAR (50) for short strings
  VARCHAR (225) for medium-length strings

MySQL Data Types

2. Integer Data Types: we use integer to store whole numbers that do not have decimal points. Most common types include:

Data type	Description
TINYINT (*size*)	A very small integer. Signed range is from -128 to 127. Unsigned range is from 0 to 255. The *size* parameter specifies the maximum display width (which is 255)
SMALLINT (*size*)	A small integer. Signed range is from -32768 to 32767. Unsigned range is from 0 to 65535. The *size* parameter specifies the maximum display width (which is 255)
MEDIUMINT (*size*)	A medium integer. Signed range is from -8388608 to 8388607. Unsigned range is from 0 to 16777215. The *size* parameter specifies the maximum display width (which is 255)
INT(*size*)	A medium integer. Signed range is from -2147483648 to 2147483647. Unsigned range is from 0 to 4294967295. The *size* parameter specifies the maximum display width (which is 255)
BIGINT (*size*)	A large integer. Signed range is from -9223372036854775808 to 9223372036854775807. Unsigned range is from 0 to 18446744073709551615.

MySQL Data Types

3. Fixed-point and floating-points types: used for storing numbers with decimal points. common types include:

Data type	Description
DECIMAL(*size*, *d*)	An exact fixed-point number. The total number of digits is specified in *size*. The number of digits after the decimal point is specified in the *d* parameter. The maximum number for *size* is 65. The maximum number for *d* is 30. The default value for *size* is 10. The default value for *d* is 0.
FLOAT(*size*, *d*)	A floating point number. The total number of digits is specified in *size*. The number of digits after the decimal point is specified in the *d* parameter.
DOUBLE(*size*, *d*)	A normal-size floating point number. The total number of digits is specified in *size*. The number of digits after the decimal point is specified in the *d* parameter

- Example: DECIMAL (9,2) => 1234567.89

MySQL Data Types

4. Boolean Types: most common types include:

Data type	Description
BOOLEAN	Zero is considered as false, nonzero values are considered as true.

MySQL Data Types

5. Date and Time Data Types: most common types include:

Data type	Description
DATE	A date. Format: YYYY-MM-DD. The supported range is from '1000-01-01' to '9999-12-31'
TIME	A time. Format: hh:mm:ss. The supported range is from '-838:59:59' to '838:59:59'
DATETIME	A date and time combination. Format: YYYY-MM-DD hh:mm:ss. The supported range is from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'. Adding DEFAULT and ON UPDATE in the column definition to get automatic initialization and updating to the current date and time
TIMESTAMP	A timestamp. TIMESTAMP values are stored as the number of seconds
YEAR	A year in four-digit format. Values allowed in four-digit format: 1901 to 2155, and 0000. MySQL 8.0 does not support year in two-digit format.

Good Practice

Use the smallest data type that suits your needs. This would make your database's size smaller and your query executed faster.

SQL Constraints

- SQL constraints are used to specify rules for the data in a table.
- **In SQL, we have the following constraints:**
- **NOT NULL:** Indicates that a column cannot store NULL value
- **UNIQUE:** Ensures that each row for a column must have a unique value
- **PRIMARY KEY:** A combination of a NOT NULL and UNIQUE.

SQL Constraints

- Ensures that a column (or combination of two or more columns) have a unique identity which helps to find a particular record in a table more easily and quickly

- **FOREIGN KEY:** Ensure the referential integrity of the data in one table to match values in another table. Syntax: FOREIGN KEY (*column 1*, *column* 2,…) REFERENCES table_ name (*column 1*, *column* 2,…) ;

- **CHECK:** Ensures that the value in a column meets a specific condition

- **DEFAULT:** Specifies a default value for a column

SQL Constraints

- The key, entity integrity, and referential integrity constraints can be specified within the CREATE TABLE statement after the attributes are declared, or they can be added later using the ALTER TABLE

- Syntax:

```
CREATE TABLE table_name (
 column1  datatype constraint,
 column2 datatype constraint,
 column3 datatype constraint,
 ....
);
```

Example 1

```
CREATE TABLE DEPARTMENT
        ( Dname                 VARCHAR(15)             NOT NULL,
          Dnumber               INT                     NOT NULL,
          Mgr_ssn               CHAR(9)                 NOT NULL,
          Mgr_start_date        DATE,
        PRIMARY KEY (Dnumber),
        UNIQUE (Dname),
        FOREIGN KEY (Mgr_ssn) REFERENCES EMPLOYEE(Ssn) );
```

Example 2

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    CHECK (Age>=18)
);
```

Example 3

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    City varchar(255) DEFAULT 'Sandnes'
);
```

Referential Integrity Constraints

- Referential integrity constraint can be violated when tuples are inserted or deleted, or when a foreign key or primary key attribute value is updated. ON DELETE and ON UPDATE.

- The default action that SQL takes for an integrity violation is to **reject** the update operation that will cause a violation, which is known as the RESTRICT option.

- However, the schema designer can specify an alternative action to be taken by attaching a **referential triggered action** clause to any foreign key constraint. The options include SET NULL, CASCADE, and SET DEFAULT.

Referential Integrity Constraints

- The action for CASCADE ON DELETE is to delete all the referencing tuples, whereas the action for CASCADE ON UPDATE is to change the value of the referencing foreign key attribute(s) to the updated (new) primary key value for all the referencing tuples.

- The action taken by the DBMS for SET NULL or SET DEFAULT is the same for both ON DELETE and ON UPDATE: The value of the affected referencing attributes is changed to NULL for SET NULL and to the specified default value of the referencing attribute for SET DEFAULT.

Example

```
CREATE TABLE EMPLOYEE
    ( ... ,
      Dno           INT             NOT NULL        DEFAULT 1,
    CONSTRAINT EMPPK
      PRIMARY KEY (Ssn),
    CONSTRAINT EMPSUPERFK
      FOREIGN KEY (Super_ssn) REFERENCES EMPLOYEE(Ssn)
                      ON DELETE SET NULL          ON UPDATE CASCADE,
    CONSTRAINT EMPDEPTFK
      FOREIGN KEY(Dno) REFERENCES DEPARTMENT(Dnumber)
                      ON DELETE SET DEFAULT       ON UPDATE CASCADE);
CREATE TABLE DEPARTMENT
    ( ... ,
      Mgr_ssn CHAR(9)           NOT NULL        DEFAULT '888665555',
      ... ,
    CONSTRAINT DEPTPK
      PRIMARY KEY(Dnumber),
    CONSTRAINT DEPTSK
      UNIQUE (Dname),
    CONSTRAINT DEPTMGRFK
      FOREIGN KEY (Mgr_ssn) REFERENCES EMPLOYEE(Ssn)
                      ON DELETE SET DEFAULT       ON UPDATE CASCADE);
CREATE TABLE DEPT_LOCATIONS
    ( ... ,
    PRIMARY KEY (Dnumber, Dlocation),
    FOREIGN KEY (Dnumber) REFERENCES DEPARTMENT(Dnumber)
                      ON DELETE CASCADE           ON UPDATE CASCADE);
```

The ALTER TABLE Command

- The definition of a table can be changed by using the ALTER command.

- ALTER TABLE actions include:
  1. Adding or dropping a column (attribute)
  2. Changing a column definition;
  3. Adding or dropping table constraints.

General Syntax:

ALTER TABLE *table_name*
Action *column_name datatype*;

The ALTER TABLE Command

1. ALTER TABLE - ADD Column

- Syntax: ALTER TABLE *table_name*
  ADD *column_name datatype*;

- Example: ALTER TABLE Customers
  ADD Email varchar(255);

2. ALTER TABLE - DROP COLUMN

- Syntax: ALTER TABLE *table_name*
  DROP COLUMN *column_name*;

- Example: ALTER TABLE Customers
  DROP COLUMN Email;

The ALTER TABLE Command

3. ALTER TABLE - ALTER/MODIFY COLUMN

- Syntax: ALTER TABLE *table_name*
  MODIFY COLUMN *column_name datatype*;

- Example: ALTER TABLE Persons
  MODIFY COLUMN DateOfBirth year;

The ALTER TABLE Command

4. ALTER TABLE - ADD PRIMARY KEY

- Syntax: ALTER TABLE *table_name*
  ADD PRIMARY KEY (*column_name*);

- Example: ALTER TABLE student
  ADD PRIMARY KEY (student_id);

5. ALTER TABLE - DROP PRIMARY KEY

- Syntax: ALTER TABLE *table_name*
  DROP PRIMARY KEY ;
- Example: ALTER TABLE student
  DROP PRIMARY KEY;

The ALTER TABLE Command

6. ALTER TABLE - ADD FOREIGN KEY

- Syntax: ALTER TABLE *table_name*
  ADD FOREIGN KEY (*column_name*) REFERENCES *table_name* (*column_name*);

- Example: ALTER TABLE Orders
  ADD FOREIGN KEY (PersonID) REFERENCES Persons(PersonID);

7. ALTER TABLE - DROP FOREIGN KEY

- Syntax: ALTER TABLE *table_name*
  DROP FOREIGN KEY (*column_name*);
- Example: ALTER TABLE Orders
  DROP FOREIGN KEY (PersonID);

The ALTER TABLE Command

8. ALTER TABLE – ADD a CHECK CONSTRAINT

- Syntax: ALTER TABLE *table_name*
  ADD CONSTRAINT *check_name*
  CHECK (Age>=18);

- Example: ALTER TABLE Persons
  ADD CONSTRAINT che123 CHECK (Age>=18);

9. ALTER TABLE - DROP a CHECK CONSTRAINT

- Syntax: ALTER TABLE *table_name*
  DROP CONSTRAINT *check_name*

- Example: ALTER TABLE Persons
  DROP CONSTRAINT che123;

The ALTER TABLE Command

10. ALTER TABLE – SET DEFAULT

- Syntax: ALTER TABLE *table_name*
  ALTER *column_name* SET DEFAULT 'default value';

- Example: ALTER TABLE Persons
  ALTER City SET DEFAULT 'Sandnes';

The DROP DATABASE Command

- DROP DATABASE command is used to drop an existing SQL database.

- Syntax:

```
DROP DATABASE databasename;
```

- Example:
- The following SQL statement drops the existing database "testDB":

```
DROP DATABASE testDB;
```

The DROP TABLE Command

- DROP TABLE command is used to drop an existing SQL table.

- Syntax:

```
DROP TABLE table_name;
```

- Example:
- The following SQL statement drops the existing table "Shippers":

```
DROP TABLE Shippers;
```