RDBMS LANGUAGE

Eng. Lina Hammad

August 8th , 2022

RDBMS language

DBMS languages

- **RDBMS** stands for Relational Database Management System.
- **RDBMS** is the basis for SQL, and for all modern database systems such as MS SQL Server, IBM DB2, Oracle, MySQL, and Microsoft Access.
- Database languages are used for **read**, **update** and **store** data in a database. There are several such languages that can be used for this purpose; one of them is SQL (Structured Query Language).
- **Types of DBMS languages:**
  - **Data Definition Language (DDL)**: DDL is used for specifying the database schema. Let's take SQL for instance to categorize the statements that comes under DDL.
  - **Data Manipulation Language (DML)**: DML is used for accessing and manipulating data in a database.
  - **Data Control language (DCL)**: DCL is used for granting and revoking user access on a database.
- **Note**: In practical data definition language, data manipulation language and data control languages are not separate language; rather they are the parts of a single database language such as SQL.

DBMS languages

- **Data Definition Language (DDL)**
  - To create the database instance – **CREATE**
  - To alter the structure of database – **ALTER**
  - To drop database instances – **DROP**
  - To delete tables in a database instance – **TRUNCATE**
- **Data Manipulation Language (DML)**
  - To read records from table(s) – **SELECT**
  - To insert record(s) into the table(s) – **INSERT**
  - Update the data in table(s) – **UPDATE**
  - Delete all the records from the table – **DELETE**
- **Data Control language (DCL)**
  - To grant access to user – **GRANT**
  - To revoke access from user – **REVOKE**

DDL language

Data Definition Language – Create Database statement

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

- Syntax:

```
CREATE DATABASE databasename;
```

- **Tip**: Make sure you have admin privilege before creating any database. Once a database is created, you can check it in the list of databases with the following SQL command: SHOW DATABASES;

Data Definition Language – Drop Database statement

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

- Syntax:

```
DROP DATABASE databasename;
```

- **Note:** Be careful before dropping a database. Deleting a database will result in loss of complete information stored in the database!

Data Definition Language – Backup Database statement (Export)

- The BACKUP DATABASE statement is used in SQL Server to create a full back up of an existing SQL database.

- To get a backup of your database, you should follow the next steps:
  1. Run the **WampServer** and then open the **phpMyadmin** in your browser.
  2. On the left, **select** the database that you will be working.
  3. Click **Export** in the top menu.
  4. Under File **format**, make sure you have selected the **SQL** option.
  5. Click **Go** at the bottom right to **export** the database SQL file.
  6. When the database has been exported **successfully**, you should see the **downloaded file** in the download folder.

Data Definition Language – Backup Database statement (Import)

- To import a backup of your database, you should follow the next steps:
  1. Run the **WampServer** and then open the **phpMyadmin** in your browser.
  2. Click **SQL** in the top menu.
  3. Run a statement that **creates** a new database.
  4. On the left, **select** the database that you created.
  5. Click **Import** in the top menu.
  6. Under File to Import, click **Browse** and select the db.sql file.
  7. Click **Go** at the bottom right to import the database file.
  8. When the database has been imported successfully, you should see a message at the top of the page similar to: **Import has been successfully finished, ## queries executed**.

Data Definition Language – Create table statement

- The CREATE TABLE statement is used to create a new table in a database.
- Syntax:

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

Data Definition Language – Create table with Constraint's statement

- Constraints can be specified when the table is created with the CREATE TABLE statement, or after the table is created with the ALTER TABLE statement.

- Syntax:

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

Data Definition Language – SQL Constraints

- SQL constraints are used to specify rules for the data in a table.

- Constraints are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the table. If there is any violation between the constraint and the data action, the action is aborted.

- Constraints can be column level or table level. Column level constraints apply to a column, and table level constraints apply to the whole table.

- The following constraints are commonly used in SQL:
  - **NOT NULL** - Ensures that a column cannot have a NULL value
  - **UNIQUE** - Ensures that all values in a column are different
  - **PRIMARY KEY** - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
  - **FOREIGN KEY** - Uniquely identifies a row/record in another table
  - **CHECK** - Ensures that all values in a column satisfies a specific condition
  - **DEFAULT** - Sets a default value for a column when no value is specified

Data Definition Language – Primary key Constraint

- The PRIMARY KEY constraint uniquely identifies each record in a table.
- Primary keys must contain UNIQUE values and cannot contain NULL values.
- A table can have only ONE primary key; and in the table, this primary key can consist of single or multiple columns (fields).

- Syntax 1:

```
CREATE TABLE table_name (
    column1 datatype constraint,
    column2 datatype constraint,
    column3 datatype constraint,
    PRIMARY KEY (column1)
)ENGINE=InnoDB;
```

- Syntax 2:

```
CREATE TABLE table_name (
    column1 datatype PRIMARY KEY,
    column2 datatype constraint,
    column3 datatype constraint,
) ENGINE=InnoDB;
```

Data Definition Language – Not null Constraint

- By default, a column can hold NULL values.
- The NOT NULL constraint enforces a column to NOT accept NULL values.
- This enforces a field to always contain a value, which means that you cannot insert a new record, or update a record without adding a value to this field.Syntax:

- Syntax:

```
CREATE TABLE table_name (
    column1 datatype PRIMARY KEY,
    column2 datatype NOT NULL,
    column3 datatype constraint
   ) ENGINE=InnoDB;
```

Data Definition Language – Unique Constraint

- The UNIQUE constraint ensures that all values in a column are different.
- Both the UNIQUE and PRIMARY KEY constraints provide a guarantee for uniqueness for a column or set of columns.
- A PRIMARY KEY constraint automatically has a UNIQUE constraint.
- However, you can have many UNIQUE constraints per table, but only one PRIMARY KEY constraint per table.Syntax:

- Syntax:

```
CREATE TABLE table_name (
    column1 datatype PRIMARY KEY,
    column2 datatype NOT NULL,
    column3 datatype UNIQUE
  ) ENGINE=InnoDB;
```

Data Definition Language – Check Constraint

- The CHECK constraint is used to limit the value range that can be placed in a column.
- If you define a CHECK constraint on a single column it allows only certain values for this column.
- If you define a CHECK constraint on a table, it can limit the values in certain columns based on values in other columns in the row.

- Syntax:

```
CREATE TABLE table_name (
    column1 datatype PRIMARY KEY,
    column2 datatype NOT NULL,
    column3 datatype UNIQUE,
    column4 datatype,
    CHECK (column4 > 20)
   ) ENGINE=InnoDB;
```

Data Definition Language – DEFAULT Constraint

- The DEFAULT constraint is used to provide a default value for a column.
- The default value will be added to all new records IF no other value is specified.

- Syntax:

```
CREATE TABLE table_name (
    column1 datatype PRIMARY KEY,
    column2 datatype NOT NULL,
    column3 datatype UNIQUE,
    column4 datatype,
    column5 datatype DEFAULT 'Sandnes',
    CHECK (column4 > 20)
   ) ENGINE=InnoDB;
```

Data Definition Language – Auto increment field

- Auto-increment allows a unique number to be generated automatically when a new record is inserted into a table.
- Often this is the primary key field that we would like to be created automatically every time a new record is inserted.

- Syntax:

```
CREATE TABLE table_name (
    column1 datatype PRIMARY KEY AUTO_INCREMENT,
    column2 datatype NOT NULL,
    column3 datatype UNIQUE,
    column4 datatype,
    column5 datatype DEFAULT 'Sandnes’,
    CHECK (column4 > 20)
    column6 datatype,




) ENGINE=InnoDB;
```

Data Definition Language – Foreign key Constraint

- A FOREIGN KEY is a key used to link two tables together.
- A FOREIGN KEY is a field (or collection of fields) in one table that refers to the PRIMARY KEY in another table.
- The table containing the foreign key is called the child table, and the table containing the candidate key is called the referenced or parent table.
- Syntax:

```
CREATE TABLE table_name (
    column1 datatype PRIMARY KEY AUTO_INCREMENT,
    column2 datatype NOT NULL,
    column3 datatype UNIQUE,
    column4 datatype,
    column5 datatype DEFAULT 'Sandnes',
    column6 datatype,
    CHECK (column4 > 20),
    CONSTRAINT 'FK_name' FOREIGN KEY(column6) REFERENCES table_name(name of the referenced column)
) ENGINE=InnoDB;
```

Data Definition Language – Delete Cascade and update Cascade

- **Delete Cascade**: When we create a foreign key using this option, it deletes the referencing rows in the child table when the referenced row is deleted in the parent table which has a primary key.
- **Update Cascade:** When we create a foreign key using UPDATE CASCADE the referencing rows are updated in the child table when the referenced row is updated in the parent table which has a primary key.

- Syntax:

```
CREATE TABLE table_name (
    column1 datatype PRIMARY KEY AUTO_INCREMENT,
    column2 datatype NOT NULL,
    column3 datatype UNIQUE,
    column4 datatype,
    column5 datatype DEFAULT 'Sandnes’,
    column6 datatype,
    CHECK (column4 > 20),
    CONSTRAINT ‘FK_name’ FOREIGN KEY(column6) REFERENCES table_name(name of the referenced column)
        ON UPDATE CASCADE ON DELETE CASCADE – or On UPDATE CASCADE ON DELETE SET NULL
) ENGINE=InnoDB;
```

Data Definition Language – Alter table statement

- The ALTER TABLE statement is used to add, delete, or modify columns in an existing table.
- The ALTER TABLE statement is also used to add and drop various constraints on an existing table.

- ALTER TABLE - ADD Column Syntax:

```
ALTER TABLE table_name ADD COLUMN column_name datatype;
```

- ALTER TABLE - DROP Column Syntax:

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

- ALTER TABLE – Modify Column Syntax:

```
ALTER TABLE table_name MODIFY COLUMN column_name datatype;
```

Data Definition Language – Alter table statement

- ALTER TABLE – ADD FK Column Syntax:

```
ALTER TABLE table_name ADD FOREIGN KEY(column_name)REFERENCES teble_name(name of the referenced column) ON UPDATE CASCADE ON DELETE SET NULL;
```

- ALTER TABLE – Drop Constraints Syntax:

```
ALTER TABLE table_name DROP CONSTRAINT(Constraint_name);
```

Data Definition Language – Rename Table statement

- The RENAME table statement is used to rename an existing table.

- Syntax:

```
RENAME TABLE old_table_name TO new_table_name;
```