Institut Informatika & Bisnis DARMAJAYA
Yayasan Alfian Husin

SAINS DATA
Institut Informatika & Bisnis Darmajaya

Kampus Merdeka INDONESIA JAYA

MERDEKA BELAJAR

DATA SCIENCE DARMAJAYA
"YOUR BEST FUTURE IN DATA"

MEETING: [17 & 18]

STRUCTURE QUERY LANGUAGE (SQL)
DATA DEFINITION LANGUAGE (DDL)
CREATE, ALTER, DROP

BY: HENDRA KURNIAWAN

[12-09-2023] | [KODE MK: SD23404 | MATAKULIAH: PENGANTAR BASIS DATA | SKS: 2/2 | VERSI: 01

STRUCTURE QUERY LANGUAGE (SQL)
DATA DEFINITION LANGUAGE (DDL)
CREATE, ALTER, DROP

1. Introduction to SQL
2. Data Definition Language (DDL) and
3. Data Manipulation Language (DML)

Learning Objectives

1. Students be able to comprehend and apply the principles of Data Definition Language (DDL),
2. including creating, modifying, and managing database objects such as tables, indexes, and constraints.
3. Students will gain the skills to design and define the structure of a database, ensuring data integrity and efficient data management.

Introduction to SQL

- A standard language used in most DBMS.
  - Well, not as standardized as one might hope
    - it keeps involving and growing
    - Vendors have the tendency to add “unique” features.
- Pronounced as “S-Q-L” or “Sequel.”
- Both as a DDL and DML language.
  - **DDL** (Data Definition Language): define the schema of the database.
  - **DML** (Data Manipulation Language): provides commands to manipulate the database (query, insert, update, delete).

- *Based* on relational algebra, but not entirely identical.
  - Relations ⇔ Tables
  - Tuples ⇔ Rows
  - Attributes ⇔ Columns
- Unlike a relation, a table is *not* a set. Duplicates are not automatically removed.
  - This is for practical reasons. Duplicate eliminations are inefficient in implementation.
- Like a relation, the order of rows in a table is irrelevant.

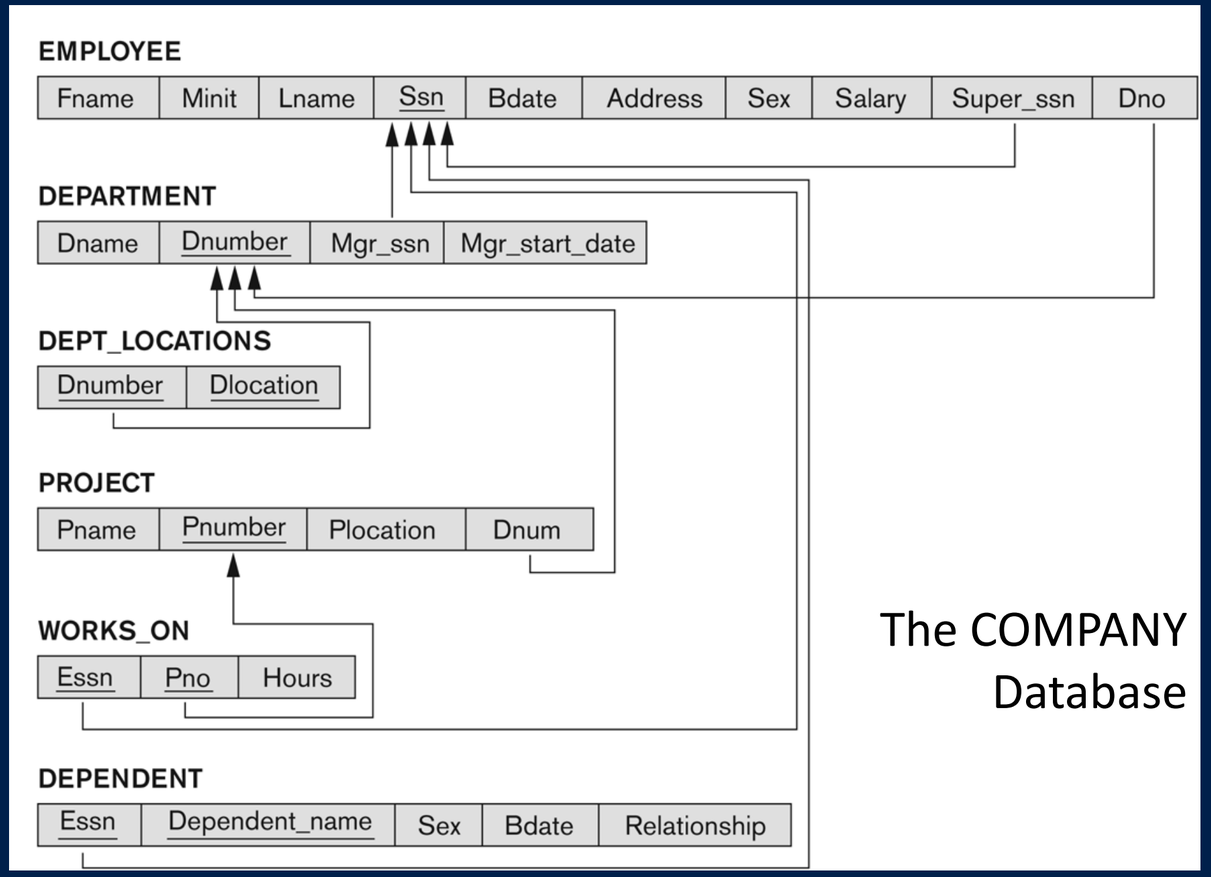
Basic DDL Commands in SQL

- **CREATE**: to define new tables (to define relation schemas)
- **DROP**: to delete table definitions (to delete relation schemas)
- **ALTER**: to change the definitions of existing tables (to change relation schema)
- Other features as DDL
  - Specify referential integrity constraints (FKs)
  - Specify user-defined attributes constraints

Basic DML Commands in SQL

- **INSERT**: to add new rows to table
- **UPDATE**: to change the “state” (the value) of rows.
- **DELETE**: to remove rows
- **SELECT**: a query command that uses relation algebra *like* expressions

- Various options available to handle the enforcement/violation of integrity constraints



The COMPANY Database

Create the COMPANY Database

- To create

  create datatbase COMPANY;
- To use (or switch to) the database

  use COMPANY;

- Subsequent commands will operate on the COMPANY database by default.

Create the COMPANY Database

- To create

  create datatbase COMPANY;
- To use (or switch to) the database

  use COMPANY;

- Subsequent commands will operate on the COMPANY database by default.

CREATE TABLE

```
CREATE TABLE DEPARTMENT (
        Dname                   VARCHAR(10) NOT NULL,
        Dnumber                 INTEGER         Default 0,
        Mgr_ssn         CHAR(9),
        Mgr_Sartdate    CHAR(9),
        PRIMARY KEY     (Dnumber),
        UNIQUE          (Dname),
        FOREIGN KEY     (Mgr_ssn)
                REFERENCES EMPLOYEE (Ssn));
```

- The “UNIQUE” clause specifies secondary keys.
- EMPLOYE)has to be created first for the FK Mgr_ssn to refer to it.
- How could we have defined the Dno FK in EMPLOYEE?

Additional Data Types

- **DATE:**
  - Made up of year-month-day in the format yyyy-mm-dd
- **TIME:**
  - Made up of hour:minute:second in the format hh:mm:ss
- **TIMESTAMP:**
  - Has both DATE and TIME components
- **Decimal (*i*,*j*):**
  - *i*: total number of digits
  - *j*: the number of digits after the decimal point
- Others: Boolean, Float, Double Precision
- See user's manual for more data types.

Adding the Dno FK to EMPLOYEE

- If “create table EMPLOYEE” is issued first, we cannot specify Dno as a FK in that create command.
- An ALTER command must be used to change the schema of EMPLOYEE, after the “create table DEPARTMENT,” to add a FK.

```
alter table EMPLOYEE
        add constraint
                foreign key (Dno)
                        references DEPARTMENT (Dnumber);
```

ALTER TABLE

ALTER TABLE employees ADD COLUMN salary INT; (Add new column)

ALTER TABLE employees ALTER COLUMN salary DECIMAL(10, 2); (Change Data Type)

ALTER TABLE employees DROP COLUMN Salary; (drop column)

ALTER TABLE employees RENAME TO staff; (change table name

Referential Integrity Options

- **Causes** of referential integrity violation for a foreign key FK (consider the `Mgr_ssn` of `DEPARTMENT`).
  - **On Delete**: when deleting the foreign tuple
    - What to do when deleting the manager tuple in `EMPLOYEE` ?
  - **On Update:** when updating the foreign tuple
    - What to do when updating/changing the SSN of the manager tuple in `EMPLOYEE` is changed ?
- **Actions** when the above two causes occur.
  - **Set Null**: the `Mgr_ssn` is set to null.
  - **Set Default**: the `Mgr_ssn` is set to the default value.
  - **Cascade**: the `Mgr_ssn` is updated accordingly
    - If the manager is deleted, the department is also deleted.

The Mgr_ssn Example

```
CREATE TABLE DEPARTMENT (
    …
    Mgr_ssn     CHAR(9),
    …
    FOREIGN KEY (Mgr_ssn)
        REFERENCES EMPLOYEE (Ssn)
          ON DELETE   ???
          ON UPDATE   ???
  );
```

Another Example

```
Create table EMP(
   …
   ESSN              CHAR(9),
   DNO               INTEGER  DEFAULT 1,
   SUPERSSN          CHAR(9),
   PRIMARY KEY (ESSN),
   FOREIGN KEY (DNO) REFERENCES DEPT
                     ON DELETE SET DEFAULT
                     ON UPDATE  CASCADE,
  FOREIGN KEY (SUPERSSN) REFERENCES EMP              ON DELETE SET
   NULL
                     ON UPDATE CASCADE);
```

Miscellaneous Commands

- show databases;
  - Show all the databases on the server
- show tables;
  - Show all the tables of the present database
- show columns from table EMPLOYEE;
- drop table *t_name*;
  - Delete the entire table *t_name*
- drop database *db_name*;
  - Delete the entire database *db_name*
- load data infile *f_name* into table *t_name*;
  - To be discussed with the next homework.

```
Microsoft Windows [Version 10.0.19045.3570]
(c) Microsoft Corporation. All rights reserved.

C:\Users\pc>cd..

C:\Users>cd..

C:\>cd xampp/mysql/bin

C:\xampp\mysql\bin>mysql -u root
```

```
MariaDB [(none)]> create database RumahSakit;
Query OK, 1 row affected (0.002 sec)

MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| phpmyadmin         |
| rumahsakit         |
| test               |
+--------------------+
6 rows in set (0.001 sec)

MariaDB [(none)]> use rumahsakit;
Database changed
MariaDB [rumahsakit]> create table pasien (id_pasien varchar(5) primary key,
    -> nama_pasien varchar(50), alamat_pasien varchar(75),
    -> jenis_kelamin varchar(9))
    -> ;
Query OK, 0 rows affected (0.024 sec)
```

```
MariaDB [rumahsakit]> desc pasien
    -> ;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| id_pasien    | varchar(5)  | NO   | PRI | NULL    |       |
| nama_pasien  | varchar(50) | YES  |     | NULL    |       |
| alamat_pasien| varchar(75) | YES  |     | NULL    |       |
| jenis_kelamin| varchar(9)  | YES  |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
4 rows in set (0.020 sec)
```

```
MariaDB [rumahsakit]> create table dokter(id_dokter char(5) not null,
    -> nama_dokter varchar(50),
    -> alamat_dokter varchar(50),
    -> tanggal_lahir datetime,
    -> no_hp varchar(13),
    -> waktukerja datetime,
    -> spesialis varchar(35),
    -> primary key (id_dokter));
Query OK, 0 rows affected (0.030 sec)
```

```
MariaDB [rumahsakit]> desc dokter
    ->
    -> ;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| id_dokter    | char(5)     | NO   | PRI | NULL    |       |
| nama_dokter  | varchar(50) | YES  |     | NULL    |       |
| alamat_dokter| varchar(50) | YES  |     | NULL    |       |
| tanggal_lahir| datetime    | YES  |     | NULL    |       |
| no_hp        | varchar(13) | YES  |     | NULL    |       |
| waktukerja   | datetime    | YES  |     | NULL    |       |
| spesialis    | varchar(35) | YES  |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
7 rows in set (0.020 sec)
```

```
MariaDB [rumahsakit]> create table administrator(id_admin char(5),
    -> nama_admin varchar(50),
    -> waktu_jaga datetime,
    -> primary key (id_admin));
Query OK, 0 rows affected (0.026 sec)
```

```
MariaDB [rumahsakit]> desc administrator;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| id_admin   | char(5)     | NO   | PRI | NULL    |       |
| nama_admin | varchar(50) | YES  |     | NULL    |       |
| waktu_jaga | datetime    | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
3 rows in set (0.018 sec)
```

```
MariaDB [rumahsakit]> create table pasien_dokter(id int(5) auto_increment,
    -> id_dokter char(5),
    -> id_pasien varchar(5),
    -> waktu_periksa datetime,
    -> resep varchar(50),
    -> primary key (id),
    -> foreign key (id_pasien) references pasien(id_pasien));
Query OK, 0 rows affected (0.064 sec)
```

```
MariaDB [rumahsakit]> desc pasien_dokter;
+--------------+-------------+------+-----+---------+----------------+
| Field        | Type        | Null | Key | Default | Extra          |
+--------------+-------------+------+-----+---------+----------------+
| id           | int(5)      | NO   | PRI | NULL    | auto_increment |
| id_dokter    | char(5)     | YES  | MUL | NULL    |                |
| id_pasien    | varchar(5)  | YES  | MUL | NULL    |                |
| waktu_periksa | datetime   | YES  |     | NULL    |                |
| resep        | varchar(50) | YES  |     | NULL    |                |
+--------------+-------------+------+-----+---------+----------------+
5 rows in set (0.018 sec)
```

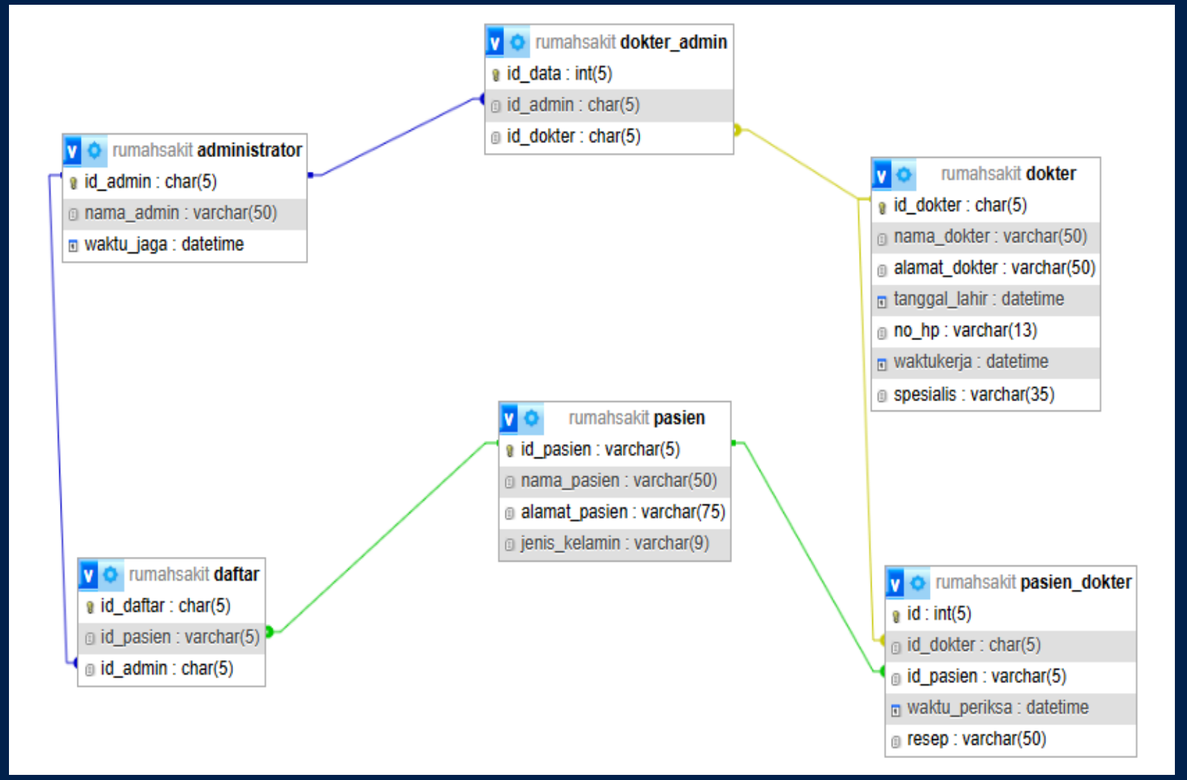
```
MariaDB [rumahsakit]> create table daftar(id_daftar char(5),
    -> id_pasien varchar(5),
    -> id_admin char(5),
    -> primary key (id_daftar),
    -> foreign key (id_pasien) references pasien(id_pasien),
    -> foreign key (id_admin) references administrator(id_admin));
Query OK, 0 rows affected (0.079 sec)

MariaDB [rumahsakit]> desc daftar
    -> ;
+-----------+------------+------+-----+---------+-------+
| Field     | Type       | Null | Key | Default | Extra |
+-----------+------------+------+-----+---------+-------+
| id_daftar | char(5)    | NO   | PRI | NULL    |       |
| id_pasien | varchar(5) | YES  | MUL | NULL    |       |
| id_admin  | char(5)    | YES  | MUL | NULL    |       |
+-----------+------------+------+-----+---------+-------+
3 rows in set (0.016 sec)
```

```
MariaDB [rumahsakit]> create table dokter_admin(id_data int(5) auto_increment,
    -> id_admin char(5),
    -> id_dokter char(5),
    -> primary key (id_data),
    -> foreign key (id_admin) references administrator(id_admin),
    -> foreign key (id_dokter) references dokter (id_dokter));
Query OK, 0 rows affected (0.059 sec)
```

```
MariaDB [rumahsakit]> desc dokter_admin;
+-----------+---------+------+-----+---------+----------------+
| Field     | Type    | Null | Key | Default | Extra          |
+-----------+---------+------+-----+---------+----------------+
| id_data   | int(5)  | NO   | PRI | NULL    | auto_increment |
| id_admin  | char(5) | YES  | MUL | NULL    |                |
| id_dokter | char(5) | YES  | MUL | NULL    |                |
+-----------+---------+------+-----+---------+----------------+
3 rows in set (0.022 sec)
```

```
MariaDB [rumahsakit]> show tables;
+----------------------+
| Tables_in_rumahsakit |
+----------------------+
| administrator        |
| daftar               |
| dokter               |
| dokter_admin         |
| pasien               |
| pasien_dokter        |
+----------------------+
6 rows in set (0.001 sec)
```



CONCLUSION

1. Table Creation and Modification: You can create new tables, modify existing ones, and define relationships among them. This is essential for designing a database that aligns with the requirements of your application.
2. Data Integrity: DDL enables you to enforce data integrity through constraints, ensuring that the data in your database meets specified rules and standards.
3. Indexing for Performance: By using DDL to create indexes, you can significantly enhance query performance, as indexes provide faster access to data.
4. Flexibility in Column Management: DDL allows you to add, modify, or remove columns, providing the flexibility to adapt the database structure as the business requirements evolve.
5. Understanding Database Objects: Through DDL, you gain a deep understanding of various database objects such as tables, indexes, constraints, and views, and how they contribute to efficient data management.
6. Schema Evolution: DDL is crucial for managing the evolution of database schemas over time, accommodating changes without compromising data consistency.

REFERENCES

1. Connoly. T., Begg. Carolyn. 2015. Database System: A Pratical Approach to Design, Implementation, and Management. Sixth Edition. Global Edition. Pearson.

Institut Informatika & Bisnis DARMAJAYA
Yayasan Alfian Husin

SAINS DATA
Institut Informatika & Bisnis Darmajaya

Kampus Merdeka INDONESIA JAYA

MERDEKA BELAJAR

THANK YOU!!

DATA SCIENCE DARMAJAYA “YOUR BEST FUTURE IN DATA”