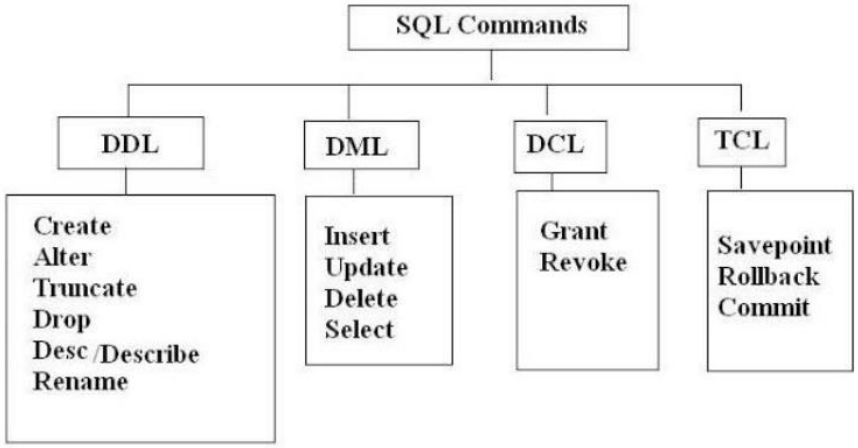
MATERI DATABASE

TIPE DATA SQL

1. **Tipe Numerik**
   a. Bilangan bulat positif dan negative
      - TINYINT (-128 s/d 127)
      - SMALLINT (-32.768 s/d 32.767)
      - MEDIUMINT (-8.388.608 s/d 8.388.607)
      - INT (-2.147.483.648 s/d 2.147.483.647)
      - BIGINT (± 9,22 x 1018)
   b. Bilangan pecahan positif dan negative
      - DOUBLE / REAL
      - DECIMAL / NUMERIC
2. **Tipe Date dan Time**
   a. Menyimpan data tanggal : DATE
   b. Menyimpan data waktu : TIME
   c. Menyimpan data tanggal dan waktu : DATETIME
   d. Menyimpan data tahun dari tanggal : YEAR
2. **Tipe String (Text)**
   a. Menyimpan Data String
      - Ukuran Tetap : CHAR (0 s/d 255 karakter)
      - Ukuran Dinamis VARCHAR (0 s/d 255 karakter (versi 4.1), 0 s/d 65.535 (versi 5.0.3))
   b. Menyimpan Data Text
      - TINYTEXT (0 s/d 255 karakter (versi 4.1), 0 s/d 65.535 (versi 5.0.3))
      - TEXT (0 s/d 65.535 (216 – 1) karakter)
      - MEDIUMTEXT (0 s/d 224 – 1 karakter)
      - LONGTEXT (0 s/d 232 – 1 karakter)
4. **Tipe BLOB (Biner) : Menyimpan Data Biner**
   a. BIT (64 digit biner)
   b. TINYBLOB (255 byte)
   c. BLOB (216 – 1 byte)
   d. MEDIUMBLOB (224 – 1 byte)
   e. LONGBLOB (232 – 1 byte)
5. **Tipe Data yang lain**
   a. ENUM : Enumerasi (kumpulan data) Sampai dengan 65535 string
   b. SET : Combination (himpunan data) Sampai dengan 255 string anggota

DATA DEFINITION LANGUAGE (DDL)

SQL COMMAND



a. CREATE :
   Digunakan untuk membuat database baru, tabel baru, view baru, dan kolom.
   `CREATE DATABASE nama_database;`
   CREATE TABLE nama_tabel (kolom1 tipe_data(panjang),kolom2 tipe_data(panjang),…kolom_n tipe_data(panjang), PRIMARY KEY (nama_kolom));
b. ALTER :
   Digunakan untuk mengubah struktur table yang telah dibuat
   Menambahkan Kolom/Field : **`ALTER TABLE table_name ADD column_name datatype;`**
   Menghapus Kolom/Field : **`ALTER TABLE table_name DROP column_name datatype;`**
c. RENAME
   Digunakan untuk merubah nama Objek : **`RENAME TABLE table_name TO new_table name;`**
d. DROP
   Menghapus database : **`DROP DATABASE nama_database;`**
   Menghapus Table **`DROP TABLE nama_tabel;`**

DATA MANIPULATION LANGUAGE (DML)

a. INSERT
   Memasukkan data baru ke dalam sebuah table : **`INSERT INTO nama_tabel VALUES (data1, data2, dst…);`**
b. SELECT
   Mengambil dan menampilkan data dari table : **`SELECT nama_kolom1, nama_kolom2 FROM nama_tabel;`**
c. UPDATE
   Memperbaharui data pada sebuah tabel. : **`UPDATE nama_tabel SET kolom1=data1, kolom2=data2,… WHERE kolom=data;`**
d. DELETE

Menghapus data dari sebuah table : **`DELETE FROM nama_tabel WHERE kolom=data;`**

DATA CONTROL LANGUAGE (DCL)

a. GRAND
   Memberikan hak akses oleh admin ke salah satu user atau pengguna
b. REVOKE
   Mencabut hak akses yang telah diberikan kepada user

OPERATOR

a. OPERATOR AS
   Digunakan untuk menampilkan kolom dengan nama lain
   `SELECT [nama_kolom] AS [nama_kolom_pengganti] FROM [nama_tabel];`
b. OPERATOR AND
   Digunakan untuk melakukan pencarian dan menampilkan data yang lebih akurat
   `SELECT * FROM [nama_tabel] WHERE [nama_kolom1] = '[pencarian1]' AND [nama_kolom2] = '[pencarian2]';`
c. OPERATOR OR
   Digunakan menampilkan data yang hanya memenuhi salah satu dari kedua syarat yang ditentukan
   `SELECT * FROM [nama_tabel] WHERE [nama_kolom] = '[pencarian1]' OR [nama_kolom] = '[pencarian2]';`
d. OPERATOR BETWEEN
   < kurang dari
   > lebih dari
   <= kurang dari sama dengan
   >= lebih dari sama dengan
   = sama dengan
   <> atau != tidak sama dengan
e. OPERATOR LIKE
   Digunakan untuk mencri data yang menyerupai atau hampir sama dengan kriteria tertentu.
   SELECT * FROM [nama_tabel] WHERE [nama_kolom] LIKE '[operator]';
   Contoh :
   - Diawali dengan huruf A : LIKE 'A%'
   - Diakhiri dengan huruf A : LIKE '%A'
   - Huruf A pada posisi kedua : LIKE '_A%
   - Diawali dengan huruf A dan mengandung huruf I : LIKE 'A%I%'
   - Tidak diawali dengan huruf A : NOT LIKE 'A%'
f. OPERATOR DISTINCT

Digunakan untuk menampilkan data tanpa duplikasi data pada suatu kolom, jika terdapat duplikasi data maka yang ditampilkan hanya satu data saja

```
SELECT DISTINCT [nama_kolom] FROM [nama_tabel];
```

g. OPERATOR LIMIT

Digunakan untuk membatasi jumlah data yang boleh ditampilkan

```
SELECT * FROM [nama_tabel] ORDER BY [nama_kolom] LIMIT [jumlah_datayang_ingin_ditampilkan];
```

h. OPERATOR OFFSET

Digunakan untuk menghilangkan jumlah data yang akan ditampilkan dan menampilkan sisa dari data yang dihilangkan.

```
SELECT * FROM [nama_tabel] ORDER BY [nama_kolom] OFFSET [jumlah_data_yang_ingin_dihilangkan];
```

i. OPERATOR CASE

Digunakan untuk membentuk output tersendiri berupa sebuah kolom baru dengan data yang berasal dari operasi yang terdapat di dalam querynya.

```
SELECT * CASE WHEN [nama_kolom] = '[isi_kolom]' THEN '[isi_kolom_baru_kondisi1]' ELSE '[isi_kolom_baru_kondisi2]' END AS [nama_kolm_baru] FROM [nama_tabel];
```

j. IF EXISTS

Operator yang menyatakan apakah suatu komponen basisdata ada atau tidak

DROP TABLE IF EXISTS hobi;

//Jika terdapat tabel hobi maka tabel tersebut akan terhapus dari basisdata

```
CREATE TABLE IF NOT EXISTS hobi (kd_hobi char(5), nm_hobi char(55), PRIMARY KEY (kd_hobi));
```

//Jika tabel hobi sudah ada, maka perintah query untuk membuat tabel dengan nama yang sama tersebut tidak bisa dilaksanakan.

k. IS NULL

Operator yang menyatakan apakah suatu nilai bernilai NULL (kosong)

```
SELECT [kolom1, kolom2, kolom3] FROM [nama_tabel] WHERE [kolom_syarat] IS NULL;
```

l. Operator UNION, EXCEPT dan INTERSECT

m. OPERATOR IN dan NOT IN

Operator IN berguna melakukan pencocokan dengan salah satu yang ada pada suatu daftar nilai

```
SELECT * FROM [nama_tabel] WHERE [nama_kolom] IN (kata_kunci1, kata_kunci2,kata_kunci3);
```

n. REGEXP

NORMALISASI DATABASE

1. **Proses Normalisasi Model Data**
   - Temukan entitas-entitas utama dalam model data
   - Temukan hubungan antara setiap entitas

- Tentukan atribut yang dimiliki masingmasing entitas

2. **Langkah-langkah Normalisasi**
   - Bentuk Normal Pertama (1NF) :
     Sebuah model data dikatakan memenuhi bentuk normal pertama apabila setiap atribut yang dimilikinya memiliki satu dan hanya satu nilai
   - Bentuk Normal Kedua (2NF) :
     Sebuah model data dikatakan memenuhi bentuk normal kedua apabila ia memenuhi bentuk normal pertama dan setiap atribut non-identifier sebuah entitas bergantung sepenuhnya hanya pada semua identifier entitas tersebut
   - Bentuk Normal Ketiga (3NF)
     Sebuah model data dikatakan memenuhi bentuk normal ketiga apabila ia memenuhi bentuk normal kedua dan tidak ada satupun atribut non-identifying (bukan mengidentifikasi unik) yang bergantung pada atribut non-identifying lain.
3. **Relasi Antar-Entitas (ERD)**
   - Relasi 1-1 (One-to-one)
   - Relasi 1-N (One-to-many) atau N-1 (Many-toone)
   - Relasi M-N (Many-to-many)
4. **Menerjemahkan Model Data**
   - Setiap entitas menjadi tabel tersendiri
   - Setiap atribut menjadi kolom-kolom tabel tersebut, dengan tipe data yang sesuai
   - Identifier entitas tersebut menjadi kolom ID yang tidak boleh kosong (NOT NULL) dan berisi indeks yang unik. ID unik ini dalam database dinamakan primary key
   - Relasi diterjemahkan menjadi foreign key

FR DATABASE

1. **Query masukkin data baru ke table**
   ```
   insert into nama_tabel value ('isi satu', 'isi dua', 'isi tiga');
   ```
2. **Query kalo mau hapus data dari table**
   ```
   delete from nama_tabel where id = 1;
   ```
3. **Query inner join untuk 3 table**
   ```
   select * from tabel 1 inner join tabel2 on tabel1.key = tabel2.key
   inner join tabel3 on tabel1.key = tabel3.key;
   ```
4. **ERD :** Entity Relationship Diagram adalah model atau rancangan untuk membuat database, supaya lebih mudah dalam menggambarkan data yang memiliki hubungan atau relasi dalam bentuk sebuah desain.
5. **Simbol ERD** : Kotak => entitas, belah ketupat => relasi, bulet lonjong => atribut
6. **Indexing :** Pengindeksan adalah cara untuk mengoptimalkan kinerja database dengan meminimalkan jumlah akses disk yang diperlukan saat query diproses.
7. **Normalisasi**

a. 1NF : Intinya pada tahap 1NF ini tidak diperbolehkan ada grouping data ataupun duplikasi data
b. 2NF : Intinya adalah pada 2NF ini tabel tersebut harus dipecah berdasarkan primary key
c. 3NF : Intinya adalah pada 3NF ini, jika terdapat suatu atribut yang tidak bergantung pada primary key tapi bergantung pada field yang lain maka atribut-atribut tersebut perlu dipisah ke tabel baru.

8. **Normalisasi adalah** mengurangi redudansi data
9. **Query yang benar dalam membuat tabel (nanti PG nya pilihan query untuk buat db)**

Membuat table :

```
CREATE TABLE nama_tabel (nama_kolom TIPE_DATA, nama_kolom TIPE DATA);
```

```
MariaDB [dbtoko]> create table tblkelompok
    -> (
    -> idkelompok INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    -> kelompok VARCHAR(100)
    -> );
Query OK, 0 rows affected (0.04 sec)
```

Membuat database :

```
CREATE DATABASE nama_database;
```

```
MariaDB [(none)]> create database dbtoko;
Query OK, 1 row affected (0.00 sec)
```

10. **GRANT -> Memberikan hak akses / hak istemewa pengguna Contoh :**

Perintah untuk buat user baru :

```
CREATE USER 'nama_user'@'localhost' IDENTIFIED BY 'password';
```

Perintah untuk memebuat hak akses full pada user :

```
GRANT ALL PRIVILEGES ON * . * TO 'nama_user'@'localhost';
```

Perintah untuk memebuat hak akses SELECT saja :

```
GRANT SELECT ON *.* TO 'nana_user'@'localhost';
```

Perintah untuk memebuat hak akses DML saja :

```
GRANT SELECT,INSERT, UPDATE, DELETE ON *.* TO 'nama_user'@'localhost';
```

11. **REVOKE -> Menarik hak akses pengguna yang diberikan lewat perintah GRANT Contoh :**

Perintah untuk mencabut hak akses INSERT pada user :

```
REVOKE INSERT ON *.* FROM 'nama_user'@'localhost';
```

Perintah untuk mencabut seluruh hak akses full:

```
REVOKE ALL ON *.* FROM ‘nama_user’@’localhost’;
```

Perintah untuk mencabut seluruh hak akses pada databas tertentu dan tabel tertentu, dapat menggunakan :

```
REVOKE ALL ON nama_database.nama_table FROM 'username'@'localhost';
```

12. Query sql cuma 1, mencari nama pemain film yg memiliki idfilm 002.

```
select pemain from film where idfilm=002;
```

13. Cara mengubah/berpindah dari 3F ke 4F?

Bentuk 3NF

↓ (Menghilangkan anomali hasil ketergantungan fungsional)

Bentuk Normal Boyce-Codd(BCNF)

↓ (Menghilangkan Ketergantungan Multivalue)

Bentuk 4NF

↓ (Menghilangkan anomali yang tersisa)

14. Query memasukkan data baru ke tabel

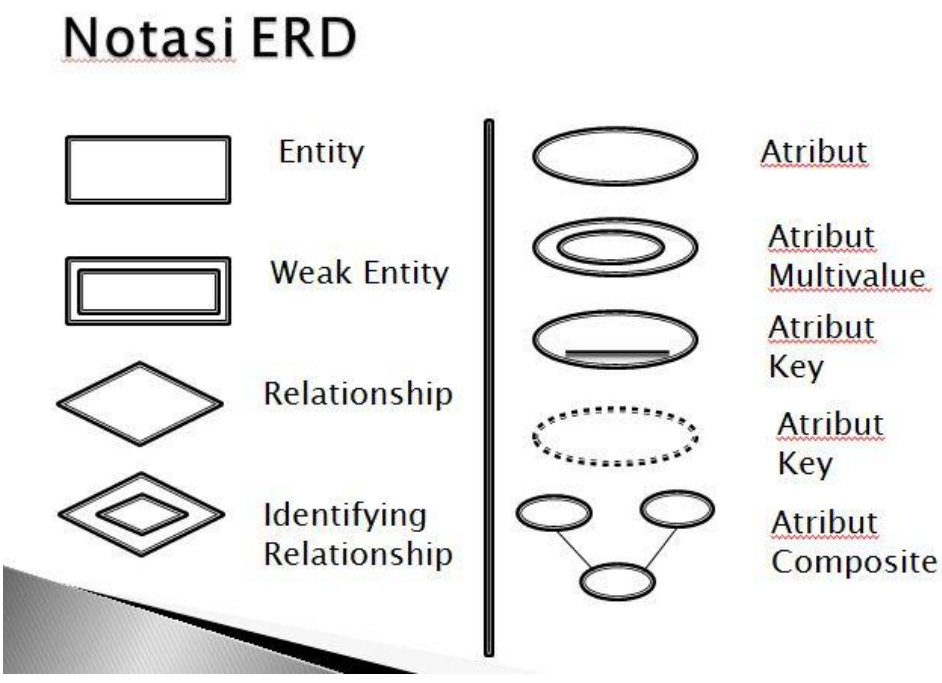
```
INSERT INTO STAFF (id, name, job, dept, salary) VALUES (111, 'Deny', 'Mgr', 20, 10000);
```

15. Query hapus tabel

```
DROP TABLE nama_tabel;
```

16. Entity Relationship Diagram

ERD atau Entity Relationship Diagram adalah suatu bentuk diagram yang menjelaskan hubungan antar objek-objek data yang mempunyai hubungan antar relasi. ERD digunakan untuk menyusun struktur data dan hubungan antar data, dan untuk menggambarkannya digunakan notasi, simbol, bagan, dan lain sebagainya.



17. Keamanan database? Keamanan server, Trusted IP Access, Koneksi database, Kontrol Akses
18. sekumpulan field yg sama dlm satu database disebut? : record
19. Grant create table? : Memberikan hak akses untuk membuat tabel
20. Implementasi ERP? Supply chain management, human resource management