NAMA: KEREN SANDRA SUDARTA
NIM : 2024071031

BASIS DATA PERTEMUAN 6

1. Pengenalan SQL

- Sejarah SQL: Dikembangkan oleh IBM tahun 1970-an berdasarkan model relasional Dr. E.F. Codd.
- Kegunaan SQL: Bahasa standar untuk mengakses dan memanipulasi database relasional.
- Pemakai SQL: Tidak hanya IT professional, tetapi juga businessman, manager, dan berbagai profesi lainnya.

2. Konsep Database

- Database Fundamentals: Kumpulan data terorganisir yang disimpan secara elektronik.
- Jenis Database: Personal, Workgroup/Departmental, Enterprise.
- Database Relasional: Terdiri dari tabel-tabel yang saling berelasi.
- DBMS: Software untuk mengelola database (contoh: SQLite, MySQL, Oracle).

3. Struktur SQL

- Fitur SQL: Non-procedural language, syntax mirip bahasa Inggris.

Jenis Perintah SQL:

- DDL (Data Definition Language): CREATE, ALTER, DROP
- DML (Data Manipulation Language): INSERT, UPDATE, DELETE
- DQL (Data Query Language): SELECT

- DCL (Data Control Language): GRANT, REVOKE
- Instalasi SQLite: Database engine gratis dan mudah digunakan.

4. Tipe Data

A. Numerik (Exact):

- **INTEGER** → Bilangan bulat (kisaran: -2.1 juta hingga +2.1 juta)
- **SMALLINT** → Integer kecil (kisaran: -32.768 hingga +32.767)
- **BIGINT** → Integer besar (kisaran: ±9.2 quintillion)
- **NUMERIC(p, s)** → Angka dengan presisi dan skala tertentu (*exact precision and scale*)
- **DECIMAL(p, s)** → Mirip dengan NUMERIC, tetapi dapat memiliki presisi lebih besar

B. Numerik (Approximate):

- **REAL** → Single-precision floating point
- **DOUBLE PRECISION** → Double-precision floating point
- **FLOAT(p, s)** → Dapat berupa single atau double precision tergantung nilai *p*

C. String:

- **CHAR(n)** → Fixed-length string (panjang tetap)
- **VARCHAR(n)** → Variable-length string (panjang berubah)
- **CLOB** → Character Large Object, untuk teks berukuran besar

D. Date / Time:

- **DATE** → Format: YYYY-MM-DD
- **TIME** → Format: HH:MM:SS
- **DATETIME** → Gabungan antara *date* dan *time*
- **TIMESTAMP** → Menyimpan waktu dan tanggal, biasanya dalam rentang tahun 1970–2038

E. Boolean:

- **TRUE**, **FALSE**, **NULL**

5. DDL Statements

- CREATE: Membuat tabel dan struktur database.
- ALTER: Memodifikasi struktur tabel.
- DROP: Menghapus tabel dan objek database.

```
CREATE TABLE nama_tabel (kolom1 tipe_data, kolom2 tipe_data);
ALTER TABLE nama_tabel ADD kolom_baru tipe_data;
DROP TABLE nama_tabel;
```

6. DML Statements

- INSERT: Menambah data ke tabel.
- UPDATE: Memperbarui data yang sudah ada.
- DELETE: Menghapus data dari tabel.

```
INSERT INTO tabel VALUES (value1, value2);
UPDATE tabel SET kolom = value WHERE kondisi;
DELETE FROM tabel WHERE kondisi;
```

7. DQL Statements

- SELECT: Mengambil data dari tabel.
- WHERE: Memfilter data berdasarkan kondisi.
- ORDER BY: Mengurutkan hasil query.
- GROUP BY: Mengelompokkan data dengan fungsi agregat.

```
SELECT kolom1, kolom2 FROM tabel WHERE kondisi;
SELECT * FROM tabel ORDER BY kolom;
SELECT kolom, COUNT(*) FROM tabel GROUP BY kolom;
```

8. Transaction Control

- COMMIT: Menyimpan perubahan secara permanen.

```
BEGIN TRANSACTION;
-- operasi SQL
COMMIT;
```

- ROLLBACK: Membatalkan transaksi.

```
BEGIN TRANSACTION;
-- operasi SQL
ROLLBACK;
```

- SAVEPOINT: Menandai titik penyimpanan dalam transaks

```
BEGIN TRANSACTION;
SAVEPOINT point1;
-- operasi SQL
ROLLBACK TO point1;
```

9. Database Views

- Definisi: Tabel virtual yang didasarkan pada query.
- Kegunaan: Simplifikasi query, meningkatkan keamanan, dan penyederhanaan data.
- Operasi: CREATE VIEW, DROP VIEW.

MEMBUAT VIEW (CREATE VIEW):

```
CREATE VIEW Customer_VW AS
SELECT CustomerName, CompanyName, ContactNo
FROM Customer_TBL;
```

MENGGUNAKAN VIEW:

```
SELECT * FROM Customer_VW;
```

MENGHAPUS VIEW (DROP VIEW):

```
DROP VIEW Customer_VW;
```

MEMBUAT DATA (CREATE) :

```
CREATE TABLE Customer_TBL (
   CustomerID INTEGER PRIMARY KEY,
   CustomerName VARCHAR NOT NULL,
   JobPosition VARCHAR,
   CompanyName VARCHAR NOT NULL,
   USState VARCHAR NOT NULL,
   ContactNo BIGINT NOT NULL
);
```

MEMASUKAN DATA (INSERT):

```
-- Single row
INSERT INTO Customer_TBL VALUES (1, 'John Doe', 'Manager', 'ABC Corp', 'NY',
123456789);

-- Multiple rows
INSERT INTO Customer_TBL VALUES
(2, 'Jane Smith', 'Director', 'XYZ Inc', 'CA', 987654321),
(3, 'Bob Wilson', 'VP', '123 Company', 'TX', 456123789);
```

QUERY DATA:

```
-- Select semua kolom
SELECT * FROM Customer_TBL;

-- Select dengan kondisi
SELECT CustomerName, CompanyName
FROM Customer_TBL
WHERE USState = 'CA';

-- Select dengan ordering
```

```
SELECT * FROM Customer_TBL ORDER BY CustomerName DESC;

-- Select dengan grouping
SELECT JobPosition, COUNT(*)
FROM Customer_TBL
GROUP BY JobPosition;
```

UPDATE DATA:

```
UPDATE Customer_TBL
SET JobPosition = 'Senior Manager'
WHERE CustomerName = 'John Doe';
```

DELETE DATA:

```
-- Delete specific records
DELETE FROM Customer_TBL WHERE CustomerID = 3;

-- Delete all records
DELETE FROM Customer_TBL;
```

10. Enhanced Database Design

- Primary Key: Identifier unik untuk setiap record.
- Foreign Key: Relasi antara tabel.
- Indexes: Mempercepat performa query.

```
CREATE INDEX idx_customer_name ON Customer_TBL (CustomerName);
DROP INDEX idx_customer_name;
```

Normalization:

- 1NF: Menghilangkan duplikasi dan memastikan atomic values.
- 2NF: Memenuhi 1NF dan menghapus partial dependency.
- 3NF: Memenuhi 2NF dan menghapus transitive dependency.

11. Advanced Topics

- **Cursors:** Memproses data baris demi baris.

```
DECLARE customer_cursor CURSOR FOR
SELECT CustomerName FROM Customer_TBL;

OPEN customer_cursor;
FETCH NEXT FROM customer_cursor;
CLOSE customer_cursor;
```

- **Triggers:** Mengotomatisasi aksi berdasarkan event tertentu.

```
CREATE TRIGGER audit_trigger
AFTER UPDATE ON Customer_TBL
FOR EACH ROW
BEGIN
   INSERT INTO audit_table VALUES (OLD.CustomerName, NEW.CustomerName,
CURRENT_TIMESTAMP);
END;
```

- **Error Handling:** Menggunakan SQLSTATE dan WHENEVER clause.

  ERROR: SYNTAX ERROR

```
-- SALAH
SELECT * Customers WHERE ID = 1;

-- BENAR
SELECT * FROM Customers WHERE ID = 1;
```

  ERROR: MISSING COMMAS

```
-- SALAH
```

```
SELECT FirstName LastName FROM Employees;

-- BENAR
SELECT FirstName, LastName FROM Employees;
```

ERROR: COLUMN NOT FOUND

```
-- SALAH
SELECT CustomerName, PhoneNumer FROM Customers;

-- BENAR
SELECT CustomerName, PhoneNumber FROM Customers;
```

ERROR: TABLE DOESN'T EXIST

```
-- SALAH
SELECT * FROM Customrs;

-- BENAR
SELECT * FROM Customers;
```