1. Konsep Dasar & ERD

Kamus: Data = fakta mentah. Database = kumpulan data terorganisir. DBMS = software manajemen (MySQL, MariaDB, Oracle, PostgreSQL). RDBMS = DBMS berbasis relasi tabel.

Notasi ERD (Chen):	**Kotak**	Entitas (objek nyata)
	Elips	Atribut (properti)
	Elips Ganda	Atribut Multivalued
	Elips Putus	Atribut Turunan (derived)
	Belah Ketupat	Relasi
	Kotak Ganda	Entitas Lemah
	Garis Bawah	Primary Key
	Garis Ganda	Partisipasi Total

Kardinalitas:

- **1:1** — Dept punya 1 Manajer
- **1:N** — Dept punya banyak Pegawai
- **N:M** — Mhs ambil banyak Matkul

EERD (Lanjutan):

- **Generalization:** Segitiga IS-A. Pegawai → Dosen, Staff.
- **Aggregation:** Relasi jadi entitas baru.

2. Normalisasi

Tujuan: hilangkan redundansi, cegah anomali insert/update/delete.

- **1NF** — Setiap sel *atomik*, tidak boleh array/multi-nilai.
- **2NF** — 1NF + tidak ada *partial dependency* (non-PK tidak bergantung sebagian PK komposit).
- **3NF** — 2NF + tidak ada *transitive dependency* (non-PK tidak bergantung non-PK lain).
- **BCNF** — 3NF + setiap determinant adalah candidate key.
- **4NF** — BCNF + tidak ada *multi-valued dependency*.

Mnemonic Codd: *"The Key, The Whole Key, and Nothing But The Key."*

3. Konversi ERD → Tabel

1. **Entitas Kuat** → Tabel. PK entitas → PK tabel.
2. **Entitas Lemah** → Tabel. PK entitas kuat masuk sebagai FK + PK komposit. Beri `ON DELETE CASCADE`.
3. **Relasi 1:1** → Pilih salah satu tabel, taruh PK lainnya sebagai FK.
4. **Relasi 1:N** → PK sisi "1" masuk sebagai FK di sisi "N".
5. **Relasi N:M** → **WAJIB** tabel baru (junction). PK komposit dari kedua FK.
6. **Atribut Multivalued** → Tabel baru.
7. **Atribut Turunan** → JANGAN disimpan, hitung saat query.

4. DDL — Struktur

Database:

```
CREATE DATABASE [IF NOT EXISTS] db;
USE db;
SHOW DATABASES;
DROP DATABASE [IF EXISTS] db;
```

Tabel:

```
CREATE TABLE [IF NOT EXISTS] tbl (
  kolom1 TIPE [constraints],
  kolom2 TIPE [constraints],
  PRIMARY KEY (kolom),
  FOREIGN KEY (kolom)
    REFERENCES tbl_lain(kolom)
    ON DELETE CASCADE
);
```

Alter:

```
ALTER TABLE t ADD COLUMN c TIPE;
ALTER TABLE t DROP COLUMN c;
ALTER TABLE t MODIFY COLUMN c TIPE;
ALTER TABLE t CHANGE c1 c2 TIPE;
ALTER TABLE t RENAME TO t_baru;
ALTER TABLE t ADD PRIMARY KEY (c);
ALTER TABLE t DROP PRIMARY KEY;
ALTER TABLE t ADD INDEX idx (c);
```

Hapus:

```
TRUNCATE TABLE t;  -- hapus baris saja
DROP TABLE [IF EXISTS] t;  -- hapus semua
```

5. Tipe Data

Integer: `TINYINT` (1B), `SMALLINT` (2B), `MEDIUMINT` (3B), `INT` (4B), `BIGINT` (8B). Tambah `UNSIGNED` untuk non-negatif.

Desimal:

- `DECIMAL(p,s)` — EXACT, untuk UANG
- `FLOAT` — approx 4B, ilmiah
- `DOUBLE` — approx 8B, ilmiah

String:

- `CHAR(n)` — fixed, selalu n byte. Kode negara, kode pos.
- `VARCHAR(n)` — variabel. Nama, email, alamat.
- `TEXT` — panjang. Artikel, deskripsi.
- `ENUM('a','b')` — pilihan terbatas.

Waktu:

- `DATE` — YYYY-MM-DD
- `TIME` — HH:MM:SS
- `DATETIME` — gabungan
- `TIMESTAMP` — UTC, bisa auto-update
- `YEAR` — 4 digit

Lain: `BOOLEAN` (alias `TINYINT(1)`), `BINARY(n)`, `VARBINARY(n)`, `BLOB`, `JSON`.

6. Constraints

`NOT NULL`	Wajib diisi
`UNIQUE`	Tidak boleh duplikat, boleh NULL
`PRIMARY KEY`	Unik + tidak null, 1 per tabel
`FOREIGN KEY`	Referensi ke PK tabel lain
`CHECK`	Nilai harus penuhi kondisi
`DEFAULT`	Nilai bawaan
`AUTO_INCREMENT`	Isi otomatis naik

7. Index

```
CREATE INDEX idx_nama ON tbl (kolom);
CREATE UNIQUE INDEX idx ON tbl (kolom);
DROP INDEX idx ON tbl;
```

Buat index jika: kolom sering di `WHERE`, `JOIN`, `ORDER BY`, selektivitas tinggi.

JANGAN buat jika: tabel kecil, kolom sering di-`UPDATE`, kolom dengan sedikit nilai unik.

8. DML — SELECT & WHERE

```
SELECT kolom FROM tbl;
SELECT DISTINCT kolom FROM tbl;
SELECT * FROM tbl LIMIT 10 OFFSET 20;
SELECT kolom AS alias FROM tbl;
SELECT * FROM tbl ORDER BY c1 ASC, c2 DESC;
```

Operator WHERE:

- Logika: `AND`, `OR`, `NOT`
- Banding: `=`, `>`, `>=`, `<`, `<=`, `<>`
- `LIKE`: `%` (0+ huruf), `_` (1 huruf)
- `BETWEEN a AND b`: inklusif
- `IN (a,b,c)`: salah satu
- `IS NULL` / `IS NOT NULL`
- `EXISTS (subquery)`

Contoh LIKE: `'Fr%'` (awal Fr), `'%re'` (akhir re), `'_re'` (2 huruf + re).

9. Fungsi Built-in

Aggregate: `COUNT`, `SUM`, `AVG`, `MIN`, `MAX`, `GROUP_CONCAT`.

String:

```
CONCAT(a,b)  UPPER(s)  LOWER(s)
SUBSTRING(s,1,3)  LENGTH(s)
TRIM(s)  REPLACE(s,'x','y')
LOCATE('x', s)
```

Date:

```
NOW()  CURDATE()  CURTIME()
YEAR(d)  MONTH(d)  DAY(d)
DATEDIFF(d1,d2)  -- selisih hari
DATE_ADD(d, INTERVAL 7 DAY)
DATE_FORMAT(d, '%Y-%m-%d')
```

Numeric:

```
ROUND(x,n)  CEIL(x)  FLOOR(x)
ABS(x)  MOD(a,b)  POWER(x,2)
```

10. Aggregate & GROUP BY

```
SELECT branch, COUNT(*), AVG(balance)
FROM Account
GROUP BY branch
HAVING AVG(balance) > 1000
ORDER BY branch;
```

Aturan: semua kolom di `SELECT` yang bukan aggregate **WAJIB** ada di `GROUP BY`.

WHERE vs HAVING: `WHERE` filter sebelum grouping. `HAVING` filter setelah grouping.

11. JOIN

Jenis:

- `INNER JOIN` — irisan (cocok di keduanya)
- `LEFT JOIN` — semua kiri + cocok kanan
- `RIGHT JOIN` — semua kanan + cocok kiri
- `FULL OUTER` — semua kedua sisi
- `CROSS JOIN` — perkalian cartesian
- `SELF JOIN` — tabel dengan dirinya

```
SELECT A.Owner, D.Amount
FROM Account A
INNER JOIN Deposit D
  ON A.Number = D.Account
WHERE D.Amount > 1000;
```

Gaya lama (implicit):

```
FROM Account A, Deposit D
WHERE A.Number = D.Account
```

12. Subquery

Scalar (1 nilai):

```
SELECT nama FROM customer
WHERE balance > (SELECT AVG(balance)
                 FROM customer);
```

IN / NOT IN:

```
SELECT * FROM customer
WHERE cust_name IN
  (SELECT cust_name FROM borrower);
```

EXISTS (korelasi):

```
SELECT * FROM customer c
WHERE EXISTS
  (SELECT 1 FROM borrower b
   WHERE b.cust_name = c.cust_name);
```

13. CRUD

```
INSERT INTO t (c1,c2) VALUES (v1,v2);
INSERT INTO t VALUES (v1,v2,v3);
INSERT INTO t SELECT ...;
```

```
UPDATE t SET c1=v1 WHERE kondisi;
UPDATE t SET balance = CASE
  WHEN balance > 1000
    THEN balance * 1.06
  ELSE balance * 1.05 END;
```

```
DELETE FROM t WHERE kondisi;
DELETE FROM t;  -- semua baris
```

14. Transaction & ACID

```
START TRANSACTION;
  ...perintah...
COMMIT;     -- simpan permanen
ROLLBACK;   -- batalkan
SAVEPOINT sp1;
ROLLBACK TO sp1;
```

ACID:

- **Atomicity** — semua atau tidak
- **Consistency** — tetap valid
- **Isolation** — tidak saling ganggu
- **Durability** — permanen setelah commit

Isolation Level (longgar → ketat):

1. `READ UNCOMMITTED` (dirty read)
2. `READ COMMITTED` (non-repeatable)
3. `REPEATABLE READ` (phantom) — default MySQL
4. `SERIALIZABLE` (paling ketat)

15. Views

```
CREATE VIEW v AS SELECT ...;
CREATE OR REPLACE VIEW v AS ...;
DROP VIEW v;
```

Contoh:

```
CREATE VIEW saldo_tinggi AS
SELECT cust_name, balance
FROM customer
WHERE balance > 5000;
```

Kegunaan: sederhanakan query, batasi akses (security), virtual table (tidak simpan data).

16. Trik Ujian (20 Poin)

1. NULL. Jangan tulis `= NULL`. Selalu `IS NULL`.

2. N:M. WAJIB tabel baru. Bukan FK di tabel existing.

3. Cartesian. `FROM A, B` tanpa `WHERE` = bencana. Selalu hubungkan PK-FK.

4. DELETE vs TRUNCATE vs DROP.

- `DELETE` — hapus baris, bisa `WHERE`, bisa rollback
- `TRUNCATE` — semua baris, reset auto-increment
- `DROP` — hapus struktur + data

5. CHAR vs VARCHAR. `CHAR` fixed (kode). `VARCHAR` variabel (nama, email).

6. FK CASCADE. `ON DELETE CASCADE` untuk auto-hapus anak.

7. COUNT(*). `COUNT(*)` hitung semua baris. `COUNT(kolom)` hitung non-null saja.

8. GROUP BY. Semua kolom `SELECT` non-aggregate wajib ada di `GROUP BY`.

9. WHERE vs HAVING. `WHERE` sebelum grouping. `HAVING` setelah.

10. LEFT JOIN. Tetap tampilkan yang tidak cocok (NULL di kanan).

11. LIKE. Case-insensitive default. Pakai `BINARY` untuk case-sensitive.

12. TIMESTAMP. Auto-update dengan `ON UPDATE CURRENT_TIMESTAMP`.

13. Compound PK. `PRIMARY KEY (a, b)` di akhir `CREATE TABLE`.

14. Subquery WHERE. Kalau return banyak nilai, pakai `IN` bukan `=`.

15. EXISTS vs IN. `EXISTS` lebih cepat untuk korelasi.

16. DECIMAL untuk uang. `FLOAT/DOUBLE` tidak presisi. Selisih kecil bisa menumpuk.

17. Aggregate di WHERE. Tidak boleh. Pakai `HAVING`. Contoh: `WHERE COUNT(*) > 5` SALAH.

18. ORDER BY nomor. `ORDER BY 1` urut kolom pertama. Portabel tapi tidak jelas.

19. LIMIT. `LIMIT 10 OFFSET 20` = ambil 10 baris lewati 20. Setara `LIMIT 20,10` di MySQL.

20. Alias. `SELECT kolom AS nama` wajib `AS` untuk nama dengan spasi: `AS 'Nama Lengkap'`.