Database design principles and entity relationship modeling

1. What is Database Design?

Database design is the structured process of organizing data into tables and defining relationships between them so that the data is:

- **Accurate** (no redundancy or inconsistency),
- **Efficiently stored and retrieved**, and
- **Easily maintainable** as the system grows.

Good design = strong foundation for your application.

🧱 2. Database Design Principles

1. Understand the Requirements

Before creating tables, understand:

- What entities (things) you need to store (e.g., User, Wallet, Transaction)
- What operations users will perform (e.g., transfer money, view balance)
- What data constraints exist (e.g., balance cannot be negative)

📘 *Example:*
In your **Digital Wallet System**, main entities might be:

- `User`
- `Wallet`
- `Transaction`

---

2. Identify Entities and Attributes

- **Entity:** A real-world object or concept represented in a table.
- **Attributes:** Properties of that entity (columns).

📘 *Example:*

Entity	Attributes
User	user_id, name, email, password
Wallet	wallet_id, balance, user_id
Transaction	txn_id, wallet_id, amount, type, description, timestamp

---

3. Define Relationships Between Entities

Relationships describe how entities are connected.

Type	Description	Example
One-to-One (1:1)	One record in A relates to one in B.	Each user has one profile.
One-to-Many (1:N)	One record in A relates to many in B.	One user can have many wallets.
Many-to-Many (M:N)	Many records in A relate to many in B.	Students enrolled in many courses.

📘 *Example in your project:*

- **User** ↔ **Wallet** → 1-to-Many
- **Wallet** ↔ **Transaction** → 1-to-Many

---

4. Use Primary and Foreign Keys

- **Primary Key (PK):** Uniquely identifies each record.
  Example: `wallet_id` in `Wallet` table.

- **Foreign Key (FK):** Links one table to another.
  Example: `user_id` in `Wallet` references `user_id` in `User`.

📘 *Wallet Table Example:*

```
CREATE TABLE Wallet (
    wallet_id BIGINT PRIMARY KEY AUTO_INCREMENT,
    balance DECIMAL(15,2) NOT NULL,
    user_id BIGINT,
    FOREIGN KEY (user_id) REFERENCES User(user_id)
);
```

---

5. Apply Normalization

Normalization ensures that data is logically stored and redundancy is minimized.

Normal Form	Rule	Example
1NF	No repeating groups; each cell contains atomic data.	Separate phone numbers into another table if multiple per user.
2NF	1NF + all non-key columns depend on the whole primary key.	Remove partial dependencies.
3NF	2NF + no transitive dependencies.	Non-key attributes depend only on the key.

✅ In short:
Each piece of data should live in only one place.

---

6. Ensure Data Integrity

Use **constraints** to enforce correctness:

- `NOT NULL` → column must have a value

- `UNIQUE` → values must be unique

- `CHECK` → restricts allowed values
- `FOREIGN KEY` → enforces valid references

📘 *Example:*

```
CREATE TABLE Transaction (
    txn_id BIGINT PRIMARY KEY AUTO_INCREMENT,
    wallet_id BIGINT NOT NULL,
    amount DECIMAL(15,2) CHECK (amount > 0),
    type ENUM('CREDIT', 'DEBIT'),
    description VARCHAR(255),
    timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (wallet_id) REFERENCES Wallet(wallet_id)
);
```

---

7. Plan for Scalability

- Use indexes for faster queries on frequently searched columns (e.g., `wallet_id`, `user_id`).
- Avoid over-normalization for performance-critical systems.
- Design with future changes in mind.

🎨 3. Entity Relationship (ER) Modeling

An **ER Model** (Entity-Relationship Model) is a **diagram** that visually represents entities, their attributes, and relationships between them.

Key Elements

Symbol	Meaning
🟩 Rectangle	Entity (e.g., User, Wallet)
🟠 Oval	Attribute (e.g., name, email)

🔷 Diamond	Relationship (e.g., owns, performs)
🔑	Primary Key

Example: Digital Wallet ER Diagram

```
[User] 1 ──────────< owns >────────── N [Wallet] 1 ──────────< has >────────── N [Transaction]

User
 ├── user_id (PK)
 ├── name
 ├── email
 └── password

Wallet
 ├── wallet_id (PK)
 ├── balance
 ├── user_id (FK)

Transaction
 ├── txn_id (PK)
 ├── wallet_id (FK)
 ├── amount
 ├── type
 ├── description
 └── timestamp
```

---

4. Steps to Create an ER Model

1. Identify entities from requirements.
2. List attributes for each entity.
3. Define relationships and their cardinality (1:1, 1:N, M:N).
4. Choose primary and foreign keys.
5. Normalize the model.

6. Convert ER model into actual tables (SQL schema).
