Database Design by Alfan Nurrohim

💡
1. Given a scenario, design tables for storing data
2. Given a scenario, identify the appropriate primary key
3. Given a scenario, choose data types to meet requirements
4. Given a scenario, design relationships between tables
5. Normalize a database
6. Given a scenario, identify data protection measures

Database Design

Database design adalah proses merancang struktur dan organisasi basis data yang efisien dan efektif untuk menyimpan, mengelola, dan mengakses data. Database design melibatkan pemodelan data dan pemilihan struktur yang tepat untuk menyimpan informasi yang relevan untuk suatu aplikasi atau sistem.

---

1. Given a scenario, design tables for storing data

Identify entities, rows/records, columns/fields

```
CREATE TABLE Persons (
    PersonID int,
    LastName varchar(255),
    FirstName varchar(255),
    Address varchar(255),
    City varchar(255)
);
where is entities, column and rows?
```

In the SQL statement you provided, you are creating a table named "Persons." Here's how the elements of entities, columns, and rows are structured in this context:

Entity: The entity in this case is "Persons." Each row in the table will represent a specific person, and the table as a whole is used to store information about people.

Columns/Fields: The columns or fields are the individual pieces of information that you are storing about each person. In your table definition:

- `PersonID` is a column that will store a unique identifier for each person.

- `LastName` is a column for storing the last name of each person.
- `FirstName` is a column for storing the first name of each person.
- `Address` is a column for storing the address of each person.
- `City` is a column for storing the city where each person lives.

These columns collectively define the attributes or properties associated with each person.

Rows/Records: In this table, each row or record represents an individual person, and it contains the values for each of the defined columns. For example, if you were to insert a record into this table, it might look like this:

- Row 1 (Record 1):
  - `PersonID`: 1
  - `LastName`: Smith
  - `FirstName`: John
  - `Address`: 123 Main St
  - `City`: New York

---

2. Given a scenario, identify the appropriate primary key

Primary key, composite/compound key

A primary key is a column or a set of columns that uniquely identifies each row in the table. The primary key follows these rules:

- A primary key must contain unique values. If the primary key consists of multiple columns, the combination of values in these columns must be unique.
- A primary key column cannot have `NULL` values. Any attempt to insert or update `NULL` to primary key columns will result in an error. Note that MySQL implicitly adds a `NOT NULL` constraint to primary key columns.
- A table can have one an only one primary key.

Dalam konteks desain basis data, "Primary Key" (kunci utama), "Composite Key" (kunci gabungan), dan "Compound Key" (kunci campuran) adalah istilah yang merujuk pada cara unik untuk mengidentifikasi setiap catatan dalam tabel

Composite Key (kunci gabungan) :

```
CREATE TABLE Sales_Transactions (
    Transaction_ID INT PRIMARY KEY,
    Customer_ID INT,
    Product_ID INT,
    Transaction_Date DATE,
    Quantity INT,
    Amount DECIMAL(10, 2),
    PRIMARY KEY (Customer_ID, Product_ID)
);
```

Dalam contoh ini, `Customer_ID` dan `Product_ID` digunakan bersama-sama sebagai kunci gabungan, yang memastikan bahwa setiap kombinasi pelanggan dan produk dalam transaksi memiliki identifikasi unik.

Compound Key (kunci campuran)

```
CREATE TABLE Bank_Accounts (
    Customer_ID INT,
    Account_Name VARCHAR(255),
    Account_Balance DECIMAL(10, 2),
    PRIMARY KEY (Customer_ID, Account_Name)
);
```

Dalam contoh ini, `Customer_ID` dan `Account_Name` digunakan bersama-sama sebagai kunci campuran untuk mengidentifikasi setiap akun bank. Hal ini memastikan bahwa setiap kombinasi dari ID Nasabah dan Nama Akun adalah unik dalam tabel.

---

3. Given a scenario, choose data types to meet requirements

Definition and importance of data types

> 💡 Definition and importance of data types; how data types affect storage requirements; data types for storing text, numbers, dates and times, and Boolean values

1. **Numeric Data Types:** Different numeric data types (e.g., integers, floating-point numbers) have varying storage requirements. For example, an `INT` might require 4 bytes of storage, while a `DOUBLE` might require 8 bytes. Choosing the appropriate numeric data type can help conserve memory.

Numeric Types	Description	
`TINYINT`	A very small integer	
`SMALLINT`	A small integer	
`MEDIUMINT`	A medium-sized integer	
`INT`	A standard integer	
`BIGINT`	A large integer	
`DECIMAL`	A fixed-point number	
`FLOAT`	A single-precision floating point number	
`DOUBLE`	A double-precision floating point number	
`BIT`	A bit field	

2. **Text Data Types:** Text data types (e.g., `VARCHAR`, `CHAR`) have storage requirements based on the length of the text they can store. Longer text requires more storage. For instance, a `VARCHAR(255)` column can store up to 255 characters. Type:

- `TINYTEXT` **– 255 Bytes (255 characters)**
- `TEXT` **– 64KB (65,535 characters)**
- `MEDIUMTEXT` **– 16MB (16,777,215 characters)**
- `LONGTEXT` **– 4GB (4,294,967,295 characters)**

3. **Date and Time Data Types:** Date and time data types (e.g., `DATE`, `DATETIME`) have fixed storage requirements. For example, a `DATE` might require 3 bytes to store the date.

Date and Time Types	Description	
`DATE`	A date value in `CCYY-MM-DD` format	
`TIME`	A time value in `hh:mm:ss` format	
`DATETIME`	A date and time value in `CCYY-MM-DD hh:mm:ss` format	
`TIMESTAMP`	A timestamp value in `CCYY-MM-DD hh:mm:ss` format	
`YEAR`	A year value in `CCYY` or `YY` format	

4. **Boolean Data Types:** Boolean values often require just a single bit of storage, representing either `true` or `false`.

Data tipe boolean, juga dikenal sebagai data tipe logika, adalah tipe data yang digunakan untuk menyimpan nilai biner yang mewakili dua kondisi yang mungkin: benar (true) atau salah (false).

```
is_student = True
is_registered = False

if is_student:
    print("This person is a student.")

if not is_registered:
    print("This person is not registered.")
```

Dalam contoh di atas, kita menggunakan variabel boolean `is_student` dan `is_registered` untuk mengontrol alur program berdasarkan kondisi yang benar atau salah. Jika `is_student` adalah `True`, maka pesan "This person is a student." akan dicetak. Jika `is_registered` adalah `False`, maka pesan "This person is not registered." akan dicetak.

---

4. Given a scenario, design relationships between tables

How to establish relationships using primary and foreign keys

> 💡 How to establish relationships using primary and foreign keys, entityrelationship diagrams (ERDs), referential integrity

A primary key is a column or a set of columns that uniquely identifies each row in the table.

A foreign key is a column or group of columns in a table that links to a column or group of columns in another table. The foreign key places constraints on data in the related tables, which allows MySQL to maintain referential integrity.

Membangun hubungan antara tabel dalam database relasional menggunakan kunci primer dan kunci asing, bersama dengan diagram entitas-relasi (ERD) dan memastikan integritas referensial, adalah konsep dasar dalam desain database. Berikut panduan langkah demi langkah tentang cara melakukannya:

1. **Identifikasi Entitas dan Atribut:**
    - Mulailah dengan mengidentifikasi entitas yang ingin Anda wakili dalam database Anda. Ini adalah objek atau konsep dunia nyata yang ingin Anda simpan informasinya. Misalnya, dalam database perpustakaan, Anda mungkin memiliki entitas seperti "Buku," "Penulis," dan "Pelanggan."
    - Untuk setiap entitas, tentukan atribut-atributnya. Ini adalah potongan informasi khusus yang ingin Anda simpan tentang setiap entitas. Untuk entitas "Buku," atributnya bisa mencakup "Judul," "ISBN," dan "Tahun Terbit."
2. **Buat Tabel:**
    - Setelah Anda mengidentifikasi entitas dan atributnya, buat tabel untuk setiap entitas. Setiap tabel akan mewakili suatu entitas, dan kolom-kolom dalam tabel akan sesuai dengan atribut-atribut dari entitas tersebut.
    - Tentukan tipe data untuk setiap kolom untuk menentukan jenis data yang akan diisi (misalnya, integer, varchar, tanggal).
3. **Desain Kunci Primer:**
    - Pilih kunci primer untuk setiap tabel. Kunci primer adalah identifikasi unik untuk setiap catatan (baris) dalam tabel. Ini memastikan bahwa setiap baris dapat diidentifikasi secara unik.
    - Pilihan umum untuk kunci primer termasuk integer yang otomatis meningkat, pengidentifikasi unik, atau kunci komposit yang terbuat dari beberapa kolom.
4. **Desain Kunci Asing:**
    - Identifikasi hubungan antara entitas. Tentukan tabel mana yang memiliki hubungan satu sama lain. Misalnya, dalam database perpustakaan, mungkin ada hubungan antara tabel "Buku" dan tabel "Penulis."
    - Dalam tabel yang mewakili sisi "banyak" dari hubungan, buat kolom kunci asing yang merujuk ke kunci primer dari tabel pada sisi "satu" dari hubungan tersebut. Kunci asing ini membangun hubungan antara dua tabel.
    - Pastikan tipe data kunci asing cocok dengan tipe data kunci primer yang dirujuk.
5. **Diagram Entitas-Relasi (ERD):**
    - Buat Diagram Entitas-Relasi (ERD) untuk memvisualisasikan hubungan antara tabel Anda. Gunakan simbol seperti persegi panjang untuk entitas, berlian untuk hubungan, dan garis yang menghubungkannya untuk menggambarkan koneksi.
    - Beri label pada hubungan dengan notasi kardinalitas (misalnya, satu-ke-satu, satu-ke-banyak, banyak-ke-banyak) untuk menunjukkan bagaimana entitas terkait.

Integritas referensial memastikan bahwa data dalam tabel yang saling terhubung tetap akurat dan tidak melanggar aturan yang telah ditentukan. Konsep ini biasanya terkait dengan penggunaan kunci primer (primary keys) dan kunci asing (foreign keys) dalam desain basis data.

Integritas referensial ini membantu menjaga data yang konsisten dan memastikan bahwa tidak ada data yang "tertunda" atau tidak valid dalam database Anda. Ini juga memungkinkan Anda untuk menjaga hubungan yang konsisten antara tabel dalam basis data relasional Anda.

---

5. Normalize a database

Reasons for normalization, how to normalize a database to third normal form (3NF)

Database normalization is a process used to organize a database into tables and columns. There are three main forms: first normal form , second normal form, and third normal form. The main idea is each table should be about a *specific* topic and only supporting topics included. Take a spreadsheet containing the information as an example, where the data contains salespeople and customers serving several purposes:

- Identify salespeople in your organization
- List all customers your company calls upon to sell a product
- Identify which salespeople call on specific customers.

Normalisasi basis data adalah sebuah proses yang digunakan untuk mengorganisir basis data menjadi tabel-tabel dan kolom-kolom. Terdapat tiga bentuk utama: bentuk normal pertama (first normal form), bentuk normal kedua (second normal form), dan bentuk normal ketiga (third normal form). Ide pokoknya adalah setiap tabel harus berkaitan dengan suatu topik yang *spesifik* dan hanya mencakup topik-topik yang mendukung.

Normalisasi adalah proses pengorganisasian data dalam sebuah basis data relasional untuk menghilangkan redudansi data dan meningkatkan integritas data. Alasan utama untuk normalisasi adalah untuk mengurangi duplikasi data, menjaga konsistensi data, dan menyederhanakan pembaruan data. Pada tiap prosesnya akan selalu dilakukan pengujian pada beberapa kondisi, apakah ada kesulitan pada saat proses insert, delete, update suatu data. Bila ada kesulitan, maka relasi tersebut dipecahkan menjadi beberapa tabel lagi.

Inti dari normalisasi adalah:

1. Membuat data yang ada memiliki integritas yang kuat
2. Tidak redundant
3. Tidak memiliki anomali.

Normalisasi dilakukan dengan mengikuti langkah-langkah sederhana, yaitu mengubahnya agar memenuhi apa yang disebut sebagai bentuk tidak normal (unnormalized), bentuk normal pertama (1NF), kedua (2NF), lalu ketiga (3NF) secara berurutan.

- **1st NORMAL FORM (1NF)**
    1. Semantik tabel menjadi lebih eksplisit.
    2. Semua operator relasional dapat diaplikasikan pada tabel.
- **2nd NORMAL FORM (2NF)**

  Untuk normalisasi ke bentuk 2NF, maka tabel 1NF didekomposisi menjadi beberapa tabel yang masing-masing memenuhi 2NF, bila terdapat ketergantungan parsial maka harus di eliminate.

  Tujuan membentuk 2NF :
    1. Semantik tabel 2NF menjadi lebih eksplisit.
    2. Mengurangi update anomaly yang masih mungkin terjadi pada 1NF.
- **3rd NORMAL FORM (3NF)**

  Jika suatu relasi sudah memenuhi 2NF tapi tidak memenuhi 3 NF, maka untuk normalisasi ke bentuk 3NF, tabel 2NF didekomposisi menjadi beberapa tabel hingga masing-masing memenuhi 3NF.

  Tujuan membentuk 3NF :
    1. Semantik tabel 3NF menjadi lebih eksplisit (fully FD hanya pada primary key).
    2. Menghindari update anomali yang masih mungkin terjadi pada 2NF.

---

6. Given a scenario, identify data protection measures

Backups, restore, principle of least privilege , GRANT, WITH GRANT OPTION, REVOKE, purpose of roles

💡 Data protection measures, atau tindakan perlindungan data, adalah strategi dan praktik yang dirancang untuk menjaga kerahasiaan, integritas, dan ketersediaan data yang disimpan dan diproses dalam sistem informasi. Tujuan utama dari tindakan perlindungan data adalah melindungi data dari ancaman dan risiko seperti akses tidak sah, perubahan data yang tidak sah, penghapusan data, atau kerusakan data akibat bencana atau gangguan sistem.

1. **MySQL Backup (Pencadangan MySQL):**
   - Pencadangan MySQL adalah proses membuat salinan data dari basis data MySQL Anda untuk melindungi data tersebut dari kehilangan, kerusakan, atau kejadian tidak terduga lainnya.
   - Perintah umum yang digunakan untuk mencadangkan database adalah `mysqldump`. Contoh: `mysqldump -u username -p database_name > backup.sql`.
   - Anda juga dapat menggunakan alat pencadangan pihak ketiga atau layanan cloud untuk pencadangan otomatis.

```
-- Mencadangkan seluruh database ke file SQL:
mysqldump -u username -p --all-databases > backup.sql

-- Mencadangkan database tertentu ke file SQL:
mysqldump -u username -p database_name > backup.sql
```

2. **MySQL Restore (Pemulihan MySQL):**
   - Pemulihan MySQL adalah proses mengembalikan data dari cadangan ke database MySQL setelah terjadinya kegagalan atau kehilangan data.
   - Perintah yang umum digunakan untuk mengembalikan database adalah `mysql`. Contoh: `mysql -u username -p database_name < backup.sql`.
   - Pastikan untuk memiliki cadangan yang up-to-date sebelum melakukan pemulihan.
3. **MySQL REVOKE (Mencabut Izin MySQL):**
   - Perintah `REVOKE` digunakan untuk mencabut izin (privilege) yang telah diberikan sebelumnya kepada pengguna MySQL.
   - Misalnya, Anda dapat mencabut izin SELECT dari pengguna dengan perintah `REVOKE SELECT ON database_name.* FROM 'username'@'localhost';`.
4. **MySQL GRANT (Memberikan Izin MySQL):**
   - Perintah `GRANT` digunakan untuk memberikan izin kepada pengguna MySQL untuk melakukan berbagai tindakan pada database atau tabel tertentu.
   - Misalnya, Anda dapat memberikan izin SELECT kepada pengguna dengan perintah `GRANT SELECT ON database_name.* TO 'username'@'localhost';`.
5. **MySQL Roles (Peran MySQL):**
   - Peran MySQL adalah kumpulan izin yang dapat diberikan kepada pengguna dalam satu operasi, membuat manajemen izin menjadi lebih efisien.

```
-- Membuat peran MySQL:
CREATE ROLE my_role;

-- Memberikan izin kepada peran:
GRANT SELECT ON database_name.* TO my_role;

-- Memberikan peran kepada pengguna:
GRANT my_role TO 'username'@'localhost';

-- Penggunaan peran dalam query:
SET ROLE my_role;
-- Sekarang pengguna memiliki izin yang telah diberikan kepada peran.
```

```
-- Mencabut peran dari pengguna:
REVOKE my_role FROM 'username'@'localhost';
```