13 - LRS

1. Apa yg dimaksud dengan LRS?

LRS (Logical Record Structure) adalah representasi logis dari database yang menggambarkan bagaimana data dalam sistem disusun berdasarkan hubungan antar entitas. LRS merupakan hasil dari konversi **ERD (Entity-Relationship Diagram)** ke dalam bentuk tabel-tabel dalam database.

2. Apa tujuan dari LRS? Mengapa ERD harus dikonversikan ke LRS?

Tujuan dari LRS:
- Mengubah desain konseptual ERD menjadi struktur yang dapat diimplementasikan dalam sistem manajemen basis data.
- Memastikan relasi antar entitas dapat direpresentasikan dalam bentuk tabel dengan relasi yang jelas.
- Mempermudah proses normalisasi untuk menghindari redudansi data.
- Memudahkan implementasi sistem berbasis database.

Mengapa ERD harus dikonversikan ke LRS?
- ERD hanya menggambarkan hubungan antar entitas secara konseptual dan belum bisa langsung digunakan dalam implementasi database.
- LRS memastikan bahwa hubungan antar entitas diterjemahkan dalam bentuk tabel yang sesuai dengan aturan basis data relasional.
- Dengan LRS, data lebih terstruktur dan bisa langsung diterapkan dalam sistem database seperti MySQL, PostgreSQL, atau SQL Server.

3. Jelaskan aturan dalam mengkonversikan ERD ke LRS! berikan contoh pada setiap poin aturan!

1. Setiap Entitas Menjadi Tabel
Aturan:

- Setiap entitas dalam ERD dikonversi menjadi tabel dengan atribut sebagai kolom.
- Kunci utama (Primary Key) ditentukan dari atribut yang unik.
  Contoh ERD:
  Entitas **Siswa** dengan atribut:

- `NIS` (Primary Key)
- `Nama`
- `Alamat`
- `Tanggal_Lahir`

Konversi ke LRS (Tabel Siswa):

```
CREATE TABLE Siswa (   NIS INT PRIMARY KEY,   Nama VARCHAR(50),   Alamat TEXT,
Tanggal_Lahir DATE );
```

Hasil Tabel Siswa:

NIS	Nama	Alamat	Tanggal_Lahir
1001	Andi	Jl. Merdeka 1	2005-03-15
1002	Siti	Jl. Sudirman 10	2004-11-22

2. Setiap Atribut Menjadi Kolom dalam Tabel
Aturan:

- Atribut dari entitas di ERD menjadi kolom dalam tabel yang sesuai.
- Tipe data kolom harus sesuai dengan data yang disimpan.
  Contoh ERD:
  Entitas **Mata_Pelajaran** dengan atribut:
- `Kode_Mapel` (Primary Key)
- `Nama_Mapel`
- `Jumlah_Jam`

Konversi ke LRS (Tabel Mata_Pelajaran):

```
CREATE TABLE Mata_Pelajaran (   Kode_Mapel CHAR(5) PRIMARY KEY,   Nama_Mapel
VARCHAR(100),   Jumlah_Jam INT );
```

Hasil Tabel Mata_Pelajaran:

Kode_Mapel	Nama_Mapel	Jumlah_Jam
M001	Matematika	4
M002	Fisika	3

3. Relasi One-to-Many Dikendalikan dengan Foreign Key
Aturan:

- Pada relasi One-to-Many, **satu** record pada tabel induk (parent) dapat berhubungan dengan **banyak** record pada tabel anak (child).
- Untuk menghubungkan kedua tabel, **primary key** dari tabel induk digunakan sebagai **foreign key** pada tabel anak.
  Contoh ERD:
- **Relasi Siswa (1) – (M) Nilai**
  - **Siswa:** Masing-masing siswa (identifikasi dengan NIS) dapat memiliki banyak nilai.
  - **Nilai:** Setiap entri nilai terkait dengan seorang siswa melalui `NIS`.

Konversi ke SQL (Tabel Nilai):

```
CREATE TABLE Nilai (    ID_Nilai INT PRIMARY KEY,    NIS INT,    Kode_Mapel
CHAR(5),    Nilai INT,    FOREIGN KEY (NIS) REFERENCES Siswa(NIS),    FOREIGN
KEY (Kode_Mapel) REFERENCES Mata_Pelajaran(Kode_Mapel) );
```

Hasil Tabel Nilai:

ID_Nilai	NIS	Kode_Mapel	Nilai	
1	1001	M001	85	
2	1002	M002	90	

4. Relasi Many-to-Many Dikendalikan dengan Tabel Relasi
Aturan:

- Pada relasi Many-to-Many, **banyak** record pada tabel A dapat berhubungan dengan **banyak** record pada tabel B.
- Untuk mengimplementasikan relasi ini, dibuatlah **tabel relasi** (junction table) yang berisi **foreign key** dari masing-masing tabel, serta seringkali dijadikan sebagai **primary key gabungan**.
  Contoh ERD:
- **Relasi Siswa (M) – (N) Ekskul (Ekstrakurikuler)**
  - **Siswa:** Seorang siswa bisa mengikuti lebih dari satu ekstrakurikuler.
  - **Ekskul:** Satu ekstrakurikuler dapat diikuti oleh banyak siswa.
  - **Siswa_Ekskul:** Tabel inilah yang mencatat relasi antara siswa dan ekstrakurikuler.

Konversi ke SQL (Tabel Relasi Siswa_Ekskul):

```
CREATE TABLE Siswa_Ekskul (    NIS INT,    ID_Ekskul INT,    PRIMARY KEY (NIS, ID_Ekskul),    FOREIGN KEY (NIS) REFERENCES Siswa(NIS),    FOREIGN KEY (ID_Ekskul) REFERENCES Ekskul(ID_Ekskul) );
```

Hasil Tabel Relasi Siswa_Ekskul:

NIS	ID_Ekskul
1001	1
1001	2
1002	1

5. Relasi One-to-One Bisa Digabung atau Dipisah
Aturan:

- Pada relasi One-to-One, setiap record pada tabel A berhubungan dengan tepat **satu** record pada tabel B, dan sebaliknya.
- Implementasinya dapat dilakukan dengan **menggabungkan** atribut kedua entitas ke dalam satu tabel, atau dengan **memisahkan** ke tabel yang berbeda dan menghubungkannya melalui foreign key.
  Contoh ERD:
- **Relasi Siswa (1) – (1) Kartu_Pelajar**
  - **Siswa:** Setiap siswa memiliki satu kartu pelajar.
  - **Kartu_Pelajar:** Tabel yang memuat data kartu pelajar, dengan setiap kartu terhubung ke satu siswa.
- **Opsi 1: Menggabungkan ke dalam Tabel Siswa**

```
CREATE TABLE Siswa (    NIS INT PRIMARY KEY,    Nama VARCHAR(50),    No_Kartu_Pelajar CHAR(10) UNIQUE );`
```

- **Opsi 2: Memisahkan ke Tabel Terpisah**

```
CREATE TABLE Kartu_Pelajar (    No_Kartu_Pelajar CHAR(10) PRIMARY KEY,    NIS INT UNIQUE,    FOREIGN KEY (NIS) REFERENCES Siswa(NIS) );
```

Hasil Tabel Kartu_Pelajar (Jika Dipisah):

No_Kartu_Pelajar	NIS
KP001	1001
KP002	1002

---

4. Apa hubungan antara LRS dan normalisasi? berikan penjelasan disertai contohnya!

LRS (Logical Relational Schema) adalah hasil dari konversi diagram ERD (Entity-Relationship Diagram) ke dalam bentuk tabel-tabel yang akan digunakan dalam database. Meskipun LRS telah memetakan entitas dan hubungan antar entitas melalui tabel dan foreign key, LRS yang dihasilkan langsung dari ERD sering kali masih mengandung **redundansi data** (pengulangan data yang sama) dan **anomali** (kesalahan dalam pengelolaan data, seperti masalah saat melakukan insert, update, atau delete).

Normalisasi adalah proses penyempurnaan struktur tabel dalam LRS untuk menghilangkan masalah tersebut. Dengan normalisasi, kita mengatur ulang data ke dalam tabel-tabel yang lebih kecil dan terpisah sehingga:

- **Redundansi data** dapat dikurangi.
- **Anomali** (update, insert, delete) dapat dihindari.
- **Integritas data** tetap terjaga.
  Normalisasi dilakukan dengan mengikuti aturan **Normal Form (NF)** seperti **1NF, 2NF, 3NF, BCNF, dll.**.

Contoh Hubungan antara LRS dan Normalisasi

Misalkan kita memiliki tabel hasil LRS langsung dari ERD sebagai berikut

NIS	Nama	Alamat	Mata_Pelajaran	Guru
1001	Andi	Jl. Merdeka 1	Matematika	Budi
1002	Siti	Jl. Sudirman 10	Fisika	Rina
1003	Rudi	Jl. Ahmad Yani 2	Kimia	Tono
1004	Lina	Jl. Kartini 5	Matematika	Budi

Masalah yang ada pada tabel tersebut:

1. **Redundansi Data:**
   - Data pelajaran "Matematika" dan guru "Budi" muncul dua kali.
2. **Anomali Update:**
   - Jika terjadi perubahan pada guru pengampu untuk mata pelajaran "Matematika", kita harus mengupdate setiap baris yang terkait. Jika ada salah satu baris yang terlewat, maka akan terjadi inkonsistensi data.

Proses Normalisasi

1. **Membagi Data Menjadi Tabel yang Lebih Spesifik**
   Tabel Siswa: Menyimpan data siswa yang unik.
   `CREATE TABLE Siswa ( NIS INT PRIMARY KEY, Nama VARCHAR(50), Alamat VARCHAR(100) );`

```
    **Contoh Data Tabel Siswa:**
```

NIS	Nama	Alamat
1001	Andi	Jl. Merdeka 1
1002	Siti	Jl. Sudirman 10
1003	Rudi	Jl. Ahmad Yani 2
1004	Lina	Jl. Kartini 5

- **Tabel Mata_Pelajaran:** Menyimpan data mata pelajaran dan gurunya.

```
  `CREATE TABLE Mata_Pelajaran (    Kode_Mapel CHAR(5) PRIMARY KEY,
Mata_Pelajaran VARCHAR(50),    Guru VARCHAR(50) );`
```

Contoh Data Tabel Mata_Pelajaran:

Kode_Mapel	Mata_Pelajaran	Guru
M001	Matematika	Budi
M002	Fisika	Rina

Kode_Mapel	Mata_Pelajaran	Guru
M003	Kimia	Tono

Tabel Relasi Siswa_Mapel: Menyimpan hubungan antara siswa dan mata pelajaran yang mereka ambil.

```
`CREATE TABLE Siswa_Mapel (    NIS INT,    Kode_Mapel CHAR(5),    PRIMARY KEY (NIS, Kode_Mapel),    FOREIGN KEY (NIS) REFERENCES Siswa(NIS),    FOREIGN KEY (Kode_Mapel) REFERENCES Mata_Pelajaran(Kode_Mapel) );`
```

Contoh Data Tabel Siswa_Mapel:

NIS	Kode_Mapel
1001	M001
1002	M002
1003	M003
1004	M001

1. **Keuntungan dari Normalisasi**

- **Mengurangi Redundansi:**
  Data pelajaran dan guru disimpan hanya satu kali di tabel **Mata_Pelajaran**.
- **Menghindari Anomali:**
  Perubahan data guru untuk mata pelajaran "Matematika" hanya perlu dilakukan di satu tempat, yakni di tabel **Mata_Pelajaran**.
- **Menjaga Integritas Data:**
  Relasi antar tabel dijaga melalui foreign key, sehingga hubungan antar data menjadi lebih konsisten.