Rakamin Academy

id/x partners

Project Based Internship

Data Warehouse Foundation

Normalization

Daftar Isi

Exploration Source

[Article]

DBMS Normalization: 1NF, 2NF, 3NF and BCNF with Examples - javatpoint

Types of Normal Forms in DBMS - GeeksforGeeks

[Video]

Basic Concept of Database Normalization - Simple Explanation for Beginners

A. Introduction

Dalam dunia *data warehouse*, istilah normalisasi sering digunakan untuk menjelaskan teknik pemrosesan data dalam jumlah besar. Sebagai seorang Data Engineer, penting untuk menguasai konsep normalisasi data, kapan mengimplementasikan nya, dan tau cara membuat normalisasi pada data.. Pada artikel ini, kita akan mengeksplorasi bagaimana normalisasi bekerja dalam konteks data warehouse menggunakan SQL. Artikel akan memberikan contoh bagaimana teknik ini dapat digunakan untuk mengekstrak *insight* dari kumpulan data besar dan meningkatkan efisiensi pemrosesan data.

B. Normalization

Normalisasi adalah metode yang digunakan dalam basis data untuk mengurangi redudansi (*duplicate*) data dan inkonsistensi data dari tabel. Dengan menggunakan normalisasi, jumlah tabel bertambah (tidak dikurangi). Meskipun normalisasi membawa sejumlah manfaat, perlu diingat bahwa **tingkat normalisasi yang tinggi juga dapat meningkatkan kompleksitas query dan pengelolaan data.** Oleh karena itu, pendekatan normalisasi harus seimbang dengan kebutuhan spesifik dari aplikasi atau sistem yang bersangkutan.

Sebagai seorang Data Engineer tentunya penting untuk pandai-pandai melihat dan menganalisis kondisi seperti apakah pada data kita yang tepat untuk mulai menerapkan normalisasi. Berikut adalah beberapa kondisi yang tepat untuk mengimplementasikan normalisasi pada data :

1. Data Terdistribusi pada Beberapa Tabel

   Ketika data tersebar di beberapa tabel dan ada potensi redundansi, normalisasi dapat membantu mengorganisir data sehingga tidak ada informasi yang disimpan secara berlebihan.

2. Basis Data yang Dirancang untuk Pemeliharaan Panjang

   Jika basis data dirancang untuk pemeliharaan (*maintenance*) jangka panjang dan akan sering mengalami pembaruan, normalisasi dapat membuat *maintenance* lebih mudah dan mengurangi risiko kesalahan.

3. Integritas Referensial Sangat Penting

   Jika integritas referensial (hubungan antara *primary key* dan *foreign key*) sangat penting, normalisasi dapat membantu menjaga konsistensi hubungan tersebut.

4. Aplikasi dengan Persyaratan Keamanan Tinggi

   Pada aplikasi yang memiliki persyaratan keamanan tinggi dan membutuhkan kontrol yang ketat terhadap data sensitif, normalisasi dapat membantu memastikan bahwa data sudah dikelompokkan secara logis.

Penting untuk diingat bahwa normalisasi bukanlah solusi satu ukuran untuk semua. Keputusan untuk menerapkan normalisasi harus dipertimbangkan dengan cermat berdasarkan kebutuhan khusus dari sistem atau aplikasi yang sedang dibangun. Terkadang, *trade-off* antara normalisasi dan denormalisasi perlu dipertimbangkan untuk memenuhi persyaratan khusus dari aplikasi tersebut.

Agar dapat memahami proses normalisasi seperti apa, silahkan pahami studi kasus normalisasi data berikut ini.

Studi Kasus : Normalisasi Data dalam Sistem Perbankan Digital

Client Anda adalah sebuah *bank* digital telah berkembang pesat dan menangani berbagai jenis transaksi keuangan. Awalnya, data transaksi disimpan dalam satu tabel tunggal. Dengan pertumbuhan yang signifikan, bank tersebut memutuskan untuk menerapkan normalisasi untuk meningkatkan efisiensi dan konsistensi data.

SOLUSI

1. **Tabel Transaksi Awal**

```
CREATE TABLE Transactions (
    TransactionID INT PRIMARY KEY,
    AccountNumber INT,
    TransactionDate DATE,
    Amount DECIMAL(15, 2),
    TransactionType VARCHAR(20),
    CustomerName VARCHAR(50),
    BranchCode VARCHAR(10),
    AdditionalInfo TEXT
);
```

Penjelasan

- Tabel **Transactions** awalnya menyimpan semua informasi transaksi dalam satu tempat.

- Informasi seperti nama pelanggan dan kode cabang dapat muncul berulang kali.

2. **Normalisasi dengan Pembentukan Tabel Terpisah**

```
CREATE TABLE Accounts (
    AccountNumber INT PRIMARY KEY,
    CustomerName VARCHAR(50),
    BranchCode VARCHAR(10)
);

CREATE TABLE TransactionDetails (
    TransactionID INT PRIMARY KEY,
    AccountNumber INT,
    TransactionDate DATE,
    Amount DECIMAL(15, 2),
    TransactionType VARCHAR(20),
    AdditionalInfo TEXT
);
```

Penjelasan

- Tabel **Accounts** dibuat untuk menyimpan informasi akun dengan *foreign key* yang terhubung ke tabel **TransactionDetails**.
- Tabel **TransactionDetails** hanya berisi detail transaksi yang berubah sering.

3. **Contoh Query Gabungan dengan Normalisasi**

```
SELECT TD.TransactionID, A.CustomerName, A.BranchCode,
TD.TransactionDate, TD.Amount
```

FROM TransactionDetails TD

JOIN Accounts A ON TD.AccountNumber = A.AccountNumber

WHERE TD.Amount > 1000;

Penjelasan

- Query ini menggabungkan informasi dari dua tabel ternormalisasi untuk memberikan hasil yang lengkap.
- Normalisasi memastikan bahwa informasi akun dan detail transaksi terpisah untuk efisiensi dan konsistensi data.

Melalui normalisasi, bank digital ini berhasil meningkatkan **struktur data**, **mengurangi redundansi**, dan **memudahkan *maintenance*** data transaksi mereka. Struktur yang ternormalisasi memungkinkan query untuk mengakses informasi yang dibutuhkan tanpa menghadapi kompleksitas struktur tabel yang besar.

REFERENCE

Normalisasi Database 1NF, 2NF & 3NF (Cara dan Contoh) | aantamim.ID

Normalization in DBMS. Overview | by Yash Goel | Medium

Database Normalization Explained

What is Normalization in SQL with Examples? 1NF, 2NF, 3NF and BCNF | Simplilearn