Rakamin Academy

id/x partners

Project Based Internship

Principles of Schema & Fact Table and Dimension Table

Unlocking the Power of Data Warehousing: Exploring Fact and Dimension Tables and the Principles of Schema Design

Daftar Isi

A. *Introduction*

Saat ini dunia digerakkan oleh data, organisasi mengumpulkan dan menganalisis data dalam jumlah besar untuk mendapatkan wawasan tentang operasi, pelanggan, dan pasar mereka. Untuk melakukannya secara efektif, mereka perlu mengandalkan prinsip-prinsip *data engineering* untuk membangun solusi *data warehouse* yang efisien dan dapat diskalakan.

Salah satu konsep terpenting dalam *data warehouse* adalah fact table dan *dimension table*. *Fact table* digunakan untuk menyimpan pengukuran kuantitatif dari proses atau peristiwa tertentu, seperti data penjualan atau lalu lintas situs web. Di sisi lain, *dimension table,* memberikan konteks tambahan tentang data di *fact table*, seperti demografi pelanggan, informasi produk, atau data geografis. Dengan menggabungkan *fact table* dan *dimension table*, para *data engineer* dapat membangun model analitik yang kuat yang memberikan wawasan ke dalam data mereka.

Namun, merancang solusi *data warehouse* yang efektif membutuhkan lebih dari sekadar memahami *fact table* dan *dimension table*. Ini juga membutuhkan pemahaman mendalam tentang prinsip-prinsip desain skema, yang menentukan struktur dan hubungan antar tabel dalam database. Skema yang dirancang dengan baik dapat secara signifikan meningkatkan kinerja kueri dan mempermudah pembuatan dan pemeliharaan alur data.

Pada artikel ini, kita akan mengeksplorasi *fact table*, *dimension table*, dan prinsip desain skema dalam *data warehouse*. Kami akan membahas manfaat

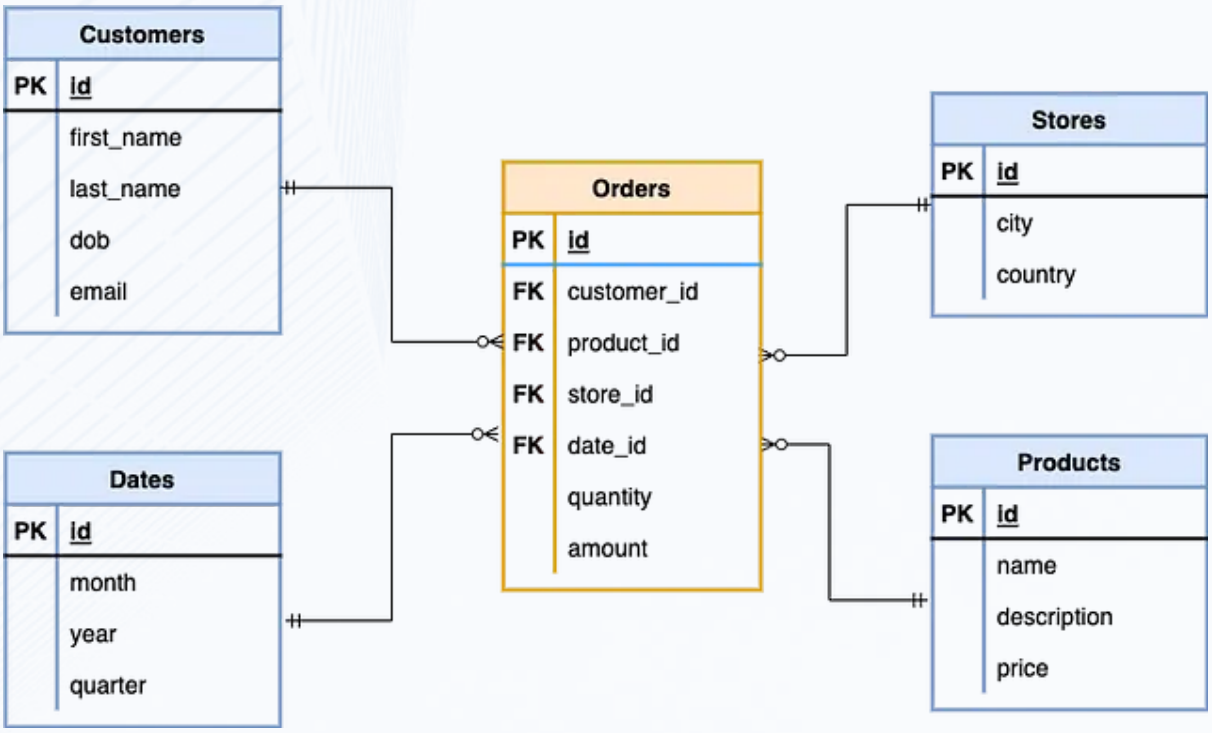
menggunakan konsep ini untuk mengatur dan menyusun data, dan memberikan beberapa praktik terbaik untuk merancang skema data yang efektif.

B. *Fact Table* dan *Dimension Table*

Fact table adalah salah satu yang memegang kunci utama (*primary key*) dari *dimension table* yang direferensikan bersama dengan beberapa metrik kuantitatif (pengukuran) di mana beberapa jenis perhitungan dapat dilakukan. Beberapa contoh umum *fact table* mencakup *orders*, log, dan data keuangan (*time series*).

Di sisi lain, *dimension table* menyimpan informasi deskriptif untuk semua bidang terkait yang termasuk dalam catatan *fact table*. Beberapa contoh umum *dimension table* adalah entitas fisik seperti tabel Pelanggan dan Produk atau bahkan Tabel Waktu. Secara umum, *Dimension table* diharapkan berukuran jauh lebih kecil dibandingkan dengan *fact table*.

Sebagai contoh, mari pertimbangkan kasus penggunaan di mana pelanggan membeli produk di toko fisik. *Star schema* diilustrasikan di bawah ini.



Gambar 1 Star Schema

Dimension table — dalam warna biru — sesuai dengan tabel yang berisi informasi tentang Pelanggan, Toko, Produk, dan Tanggal. Ini adalah kata benda dari kasus bisnis.

Fact table yang ditampilkan dalam warna orange, berisi semua Kunci Utama (*Primary Key*/ PK) dari tabel dimensi — yang merupakan Kunci Asing (*Foreign Key*/ FK) dalam tabel fakta — bersama dengan dua bidang kuantitatif, yaitu kuantitas dan jumlah.

C. *Principles of Schema*

Skema pada *data warehouse* adalah struktur yang mendefinisikan organisasi data dalam *data warehouse*. Ini adalah komponen penting dari setiap solusi *data*

warehouse karena menyediakan *blue print* untuk desain, implementasi, dan pemeliharaan *data warehouse*. Skema *data warehouse* yang dirancang dengan baik sangat penting untuk memastikan bahwa data dapat diakses, akurat, dan relevan bagi pengguna. Ada beberapa prinsip yang mendasar untuk membuat skema *data warehouse* yang efektif. Pada artikel ini, kita akan mengeksplorasi prinsip-prinsip ini dan kepentingannya dalam konteks *data warehouse*.

1. Identifikasi Entitas

   Langkah pertama dalam desain skema adalah mengidentifikasi entitas yang akan disimpan dalam *database*. Entitas adalah orang, tempat, benda, atau konsep yang relevan dengan aplikasi. Misalnya, dalam sistem registrasi mahasiswa, entitas mungkin termasuk mahasiswa, mata kuliah, dosen, dan departemen. Penting untuk mengidentifikasi semua entitas yang relevan untuk memastikan bahwa database tersebut komprehensif dan mencakup semua data yang diperlukan.

2. Definisikan Hubungan

   Setelah entitas diidentifikasi, langkah selanjutnya adalah mendefinisikan hubungan antara entitas tersebut. Hubungan adalah koneksi antara dua atau lebih entitas yang menjelaskan bagaimana mereka saling berhubungan. Ada tiga jenis hubungan dalam database relasional: *one-to-one, one-to-many, dan many-to-many*. Misalnya, dalam sistem registrasi mahasiswa, seorang mahasiswa dapat mendaftar ke banyak mata kuliah, dan setiap mata kuliah dapat memiliki banyak mahasiswa. Ini adalah contoh dari hubungan many-to-many.

3. Normalisasi Data

Normalisasi adalah proses pengorganisasian data dalam *database* untuk menghilangkan redudansi dan meningkatkan integritas data. Ada beberapa tingkat normalisasi, dan setiap tingkat membangun di atas tingkat sebelumnya. Tujuan normalisasi adalah untuk memastikan bahwa data disimpan dengan cara yang meminimalkan duplikasi dan inkonsistensi.

4. Denormalisasi untuk Performa

   Sementara normalisasi penting untuk kualitas dan konsistensi data, denormalisasi dapat digunakan untuk meningkatkan kinerja. Denormalisasi melibatkan penggabungan tabel yang dinormalisasi untuk membuat tabel baru yang dioptimalkan yang memenuhi kebutuhan khusus pengguna. Denormalisasi penting dalam lingkungan pergudangan data karena dapat meningkatkan kinerja kueri dan mengurangi jumlah gabungan yang diperlukan untuk mengakses data.

5. Pilih Jenis Data

   Setiap bidang dalam database harus diberi jenis data. Jenis data mendefinisikan jenis data yang dapat disimpan dalam bidang, seperti teks, angka, atau tanggal. Penting untuk memilih jenis data yang tepat untuk memastikan bahwa data disimpan dengan akurasi dan efisiensi.

6. Buat Indeks

   Indeks digunakan untuk meningkatkan kinerja kueri dengan memungkinkan database untuk dengan cepat menemukan data yang relevan. Indeks adalah struktur data yang dibuat untuk satu atau lebih kolom dalam tabel. Ketika kueri dieksekusi, database menggunakan indeks untuk menemukan data, daripada mencari seluruh tabel.

7. Gunakan *Constraint*

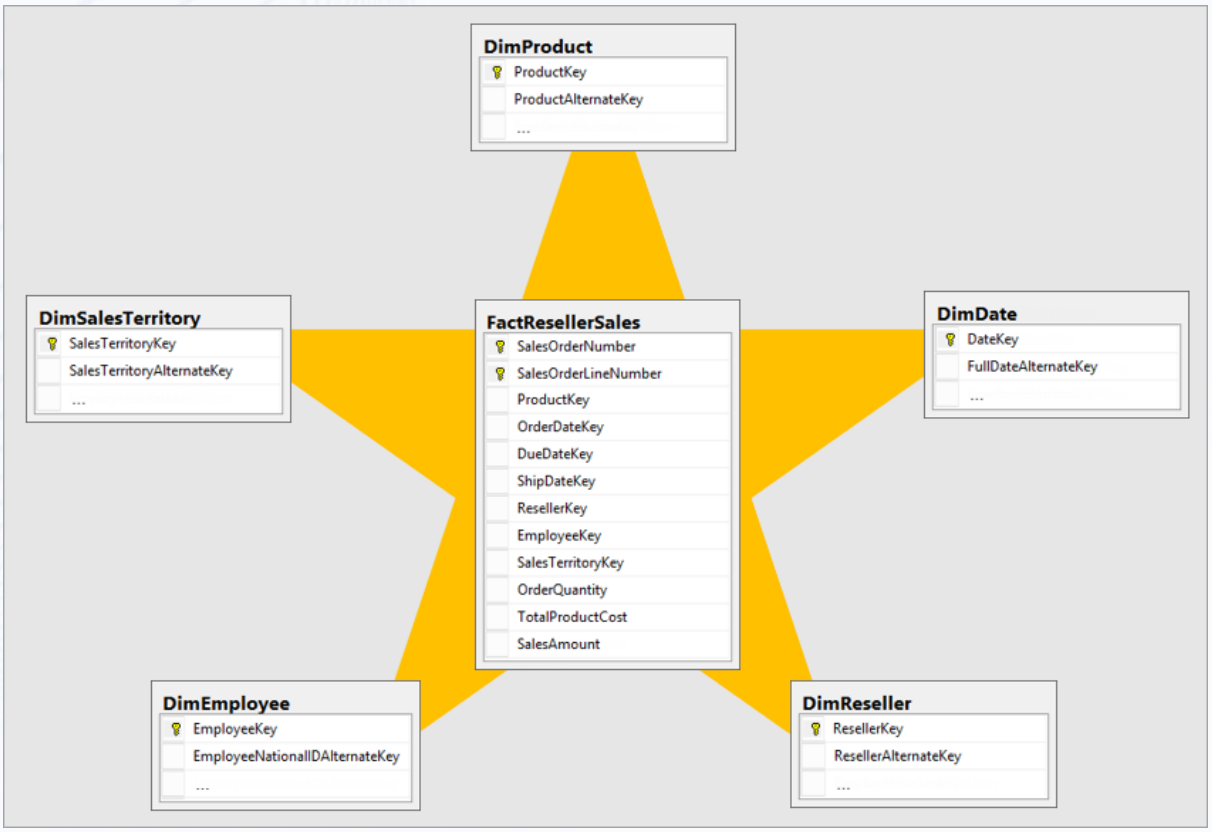
Constraint digunakan untuk menerapkan aturan pada data yang disimpan dalam database. Ada beberapa jenis kendala, termasuk kendala kunci utama, kendala kunci asing, dan kendala cek. Kendala kunci utama digunakan untuk memastikan bahwa setiap catatan dalam tabel adalah unik. Kendala kunci asing digunakan untuk menerapkan hubungan antara tabel, dan kendala cek digunakan untuk memastikan bahwa data memenuhi kondisi tertentu.

8. Dokumentasikan Skema

   Dokumentasi sangat penting dalam pergudangan data karena membantu memastikan bahwa skema dipahami dan dipelihara dari waktu ke waktu. Dokumentasi skema harus menyertakan deskripsi skema, persyaratan bisnis, model data, sumber data, dan aturan pemetaan data. Dokumentasi juga harus menyertakan informasi tentang asumsi atau kendala yang digunakan dalam desain skema.

1) Star Schema : A Simple and Intuitive Design

Star Schema adalah skema populer dalam *data warehouse* karena desainnya yang sederhana dan intuitif. Dalam skema ini, data disusun menjadi tabel fakta (*fact table*) pusat yang terhubung ke beberapa tabel dimensi (*dimension table*) melalui *foreign key*. Tabel fakta berisi data paling penting di gudang, seperti angka penjualan, dan tabel dimensi berisi informasi pendukung seperti detail produk dan data pelanggan. Skema Bintang relatif mudah dipahami dan digunakan, menjadikannya pilihan yang sangat baik untuk bisnis yang baru memulai pergudangan data.



Gambar 2. Star Schema

Kelebihan *star schema*:

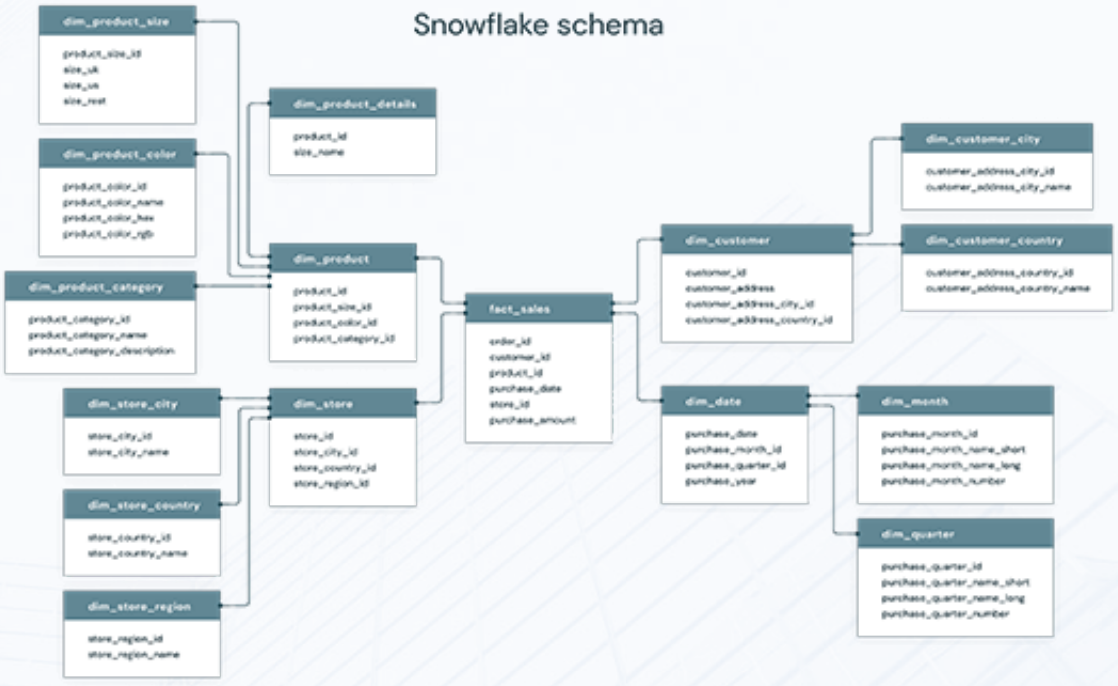
- Desain sederhana dan intuitif,
- Mudah dipahami dan digunakan,
- Efisien untuk kueri sederhana,
- Membutuhkan lebih sedikit ruang penyimpanan daripada *snowflake schema*,
- Baik untuk bisnis yang baru memulai dengan *data warehouse*.

Kelemahan *star schema*:

- Dapat mengakibatkan redundansi data,
- Tidak sefleksibel skema lainnya,
- Mungkin tidak menangani kueri kompleks seefisien skema lainnya.

2) *Snowflake Schema: A More Complex Design*

Snowflake schema adalah perpanjangan dari *star schema* yang menambahkan lebih banyak kerumitan pada desain. Dalam skema ini, tabel dimensi dinormalisasi lebih lanjut, menciptakan struktur yang lebih kompleks. Sementara *Snowflake Schema* memberikan fleksibilitas yang lebih besar dan kualitas data yang lebih baik, itu juga dilengkapi dengan peningkatan kompleksitas dan membutuhkan lebih banyak ruang penyimpanan. Akibatnya, umumnya digunakan untuk data warehouse yang lebih besar di mana kualitas data adalah yang terpenting.



Gambar 3 *Snowflake Schema*

Kelebihan *snowflake schema*:

- Memberikan fleksibilitas yang lebih besar dan kualitas data yang lebih baik daripada *star schema*,

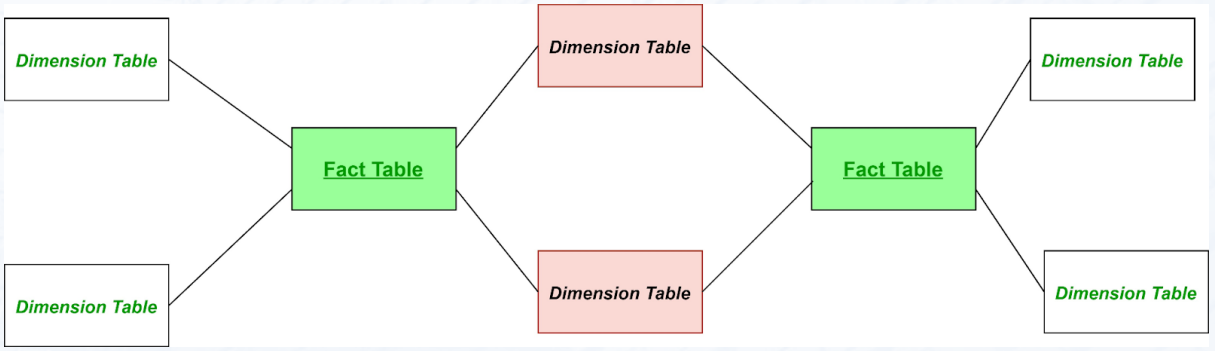
- Dapat menangani struktur dan kueri data yang lebih kompleks daripada *star schema*,
- Bagus untuk *data warehouse* yang lebih besar di mana kualitas data adalah yang terpenting

Kelemahan *snowflake schema*:

- Membutuhkan lebih banyak ruang penyimpanan daripada *star schema*,
- Lebih kompleks untuk dipahami dan digunakan daripada *star schema*,
- Mungkin memerlukan sumber daya dan keahlian tambahan untuk mengelola

3) Fast Constellation Schema: The Game-Changer Among Star and Snowflake Schemas

Fast Constellation Schema adalah skema revolusioner yang dengan cepat menjadi pilihan utama bagi bisnis yang ingin mengoptimalkan operasi data warehouse mereka. Dalam skema ini, data diatur ke dalam konstelasi *star schema*, dengan setiap *star schema* mewakili proses bisnis tertentu. *Star schema* ini saling terhubung melalui tabel dimensi bersama, memungkinkan bisnis untuk melakukan analisis data yang kompleks sambil tetap mempertahankan kinerja tinggi.



Gambar 4 *Fact Costellation Schema*

Salah satu keunggulan utama *Fast Constellation Schema* adalah kemampuannya untuk menangani volume data yang besar secara efisien. Dengan memecah data menjadi *star schema* yang lebih kecil, bisnis dapat memproses data lebih cepat, menghasilkan peningkatan kinerja dan peningkatan efisiensi. Selain itu, *Fast Constellation Schema* sangat dapat di skalakan, memungkinkan bisnis menambahkan *star schema* baru saat kebutuhan data mereka bertambah.

Keunggulan lain dari Fast Constellation Schema adalah kemampuannya menangani kueri kompleks dengan mudah. Karena skema dirancang untuk saling terhubung, bisnis dapat melakukan kueri kompleks di beberapa skema bintang tanpa mengalami penurunan kinerja apa pun.

Terakhir, Skema Konstelasi Cepat sangat fleksibel, memungkinkan bisnis menyesuaikan operasi pergudangan data mereka untuk memenuhi kebutuhan unik mereka. Bisnis dapat membuat star schema yang spesifik untuk proses bisnis mereka, memastikan bahwa mereka memiliki data yang mereka perlukan untuk mengambil keputusan. Namun skema ini juga memiliki beberapa kelemahan, diantaranya:

- Lebih kompleks untuk dipahami dan digunakan daripada *star schema*,
- Mungkin memerlukan sumber daya dan keahlian tambahan untuk mengelola
- Membutuhkan lebih banyak ruang penyimpanan daripada *star schema* dalam skenario tertentu

References

https://towardsdatascience.com/star-schema-924b995a9bdf

https://www.toptal.com/data-science/data-warehouse-concepts-principles

https://dataschool.com/data-governance/single-source-of-truth/

https://learn.microsoft.com/id-id/power-bi/guidance/star-schema

https://www.databricks.com/glossary/snowflake-schema

https://www.geeksforgeeks.org/fact-constellation-in-data-warehouse-modelling/