LAPORAN *DATA WAREHOUSE*

"Perancangan Data Warehouse untuk Analisis Preferensi Konsumen Kendaraan di Industri Otomotif"

Disusun Oleh :

Arafi Ramadhan Maulana	121450137
Rayan Koemi Karuby	122450038
Hermawan Manurung	122450069
Chevando Daffa Pramanda	122450095
Mirzan Yusuf Rabbani	122450118
Daffa Ahmad Naufal	122450137

PROGRAM STUDI SAINS DATA

FAKULTAS SAINS

INSTITUT TEKNOLOGI SUMATERA

LAMPUNG SELATAN

2025

1. STRUKTUR DATA WAREHOUSE

1.1. Konsep Dimensi dan Fakta

Dalam Data Warehouse PT Mobilita Nusantara:

- Tabel Fakta menyimpan data kuantitatif (numerik) yang dapat diukur seperti jumlah unit terjual, harga total, dan margin laba.
- Tabel Dimensi menyediakan konteks bagi data dalam tabel fakta, seperti dimensi waktu (tanggal), pelanggan, dealer, dan kendaraan.

Data warehouse menggunakan **Star Schema**, di mana:

- Tabel fakta berada di tengah.
- Tabel dimensi mengelilinginya dan terhubung langsung.

1.2. Tabel DIM_DATE

Nama Atribut	**Tipe Data**	**Deskripsi**
Date_Key	INT	Primary key
Date	DATE	Tanggal
Month	INT	Bulan
Quarter	INT	Kuartal
Year	INT	Tahun

1.3. Tabel DIM_CUSTOMER

Nama Atribut	**Tipe Data**	**Deskripsi**
Customer_Key	INT	Primary key
Full_Name	VARCHAR	Nama Pelanggan
Age	INT	Usia

Gender	CHAR(1)	Jenis Kelamin (L/P)
Segmentation	VARCHAR	Segmentasi Pasar

1.4. Tabel DIM_DEALER

Nama Atribut	**Tipe Data**	**Deskripsi**
Dealer_Key	INT	Primary key
Dealer_Name	VARCHAR	Nama Dealer
Region	VARCHAR	Wilayah Operasional

1.5. Tabel DIM_VEHICLE

Nama Atribut	**Tipe Data**	**Deskripsi**
Vehicle_Key	INT	Primary key
Brand	VARCHAR	Merek kendaraan
Model	VARCHAR	Modelkendaraan
Price	DECIMAL	Harga kendaraan
Kategori	VARCHAR	Kategori (SUV, Sedan, dll.)

1.6. Tabel FACT_SALES

Nama Atribut	**Tipe Data**	**Deskripsi**
Date_Key	INT	FK ke DIM_DATE
Customer_Key	INT	FK ke DIM_CUSTOMER
Vehicle_Key	INT	FK ke DIM_VEHICLE
Dealer_Key	INT	FK ke DIM_DEALER
Units_Sold	INT	Jumlah unit yang terjual
Total_Price	DECIMAL	Total nilai penjualan
Profit_Margin	DECIMAL	Margin keuntungan dari penjualan

2. PERANCANGAN SKEMA DATA WAREHOUSE

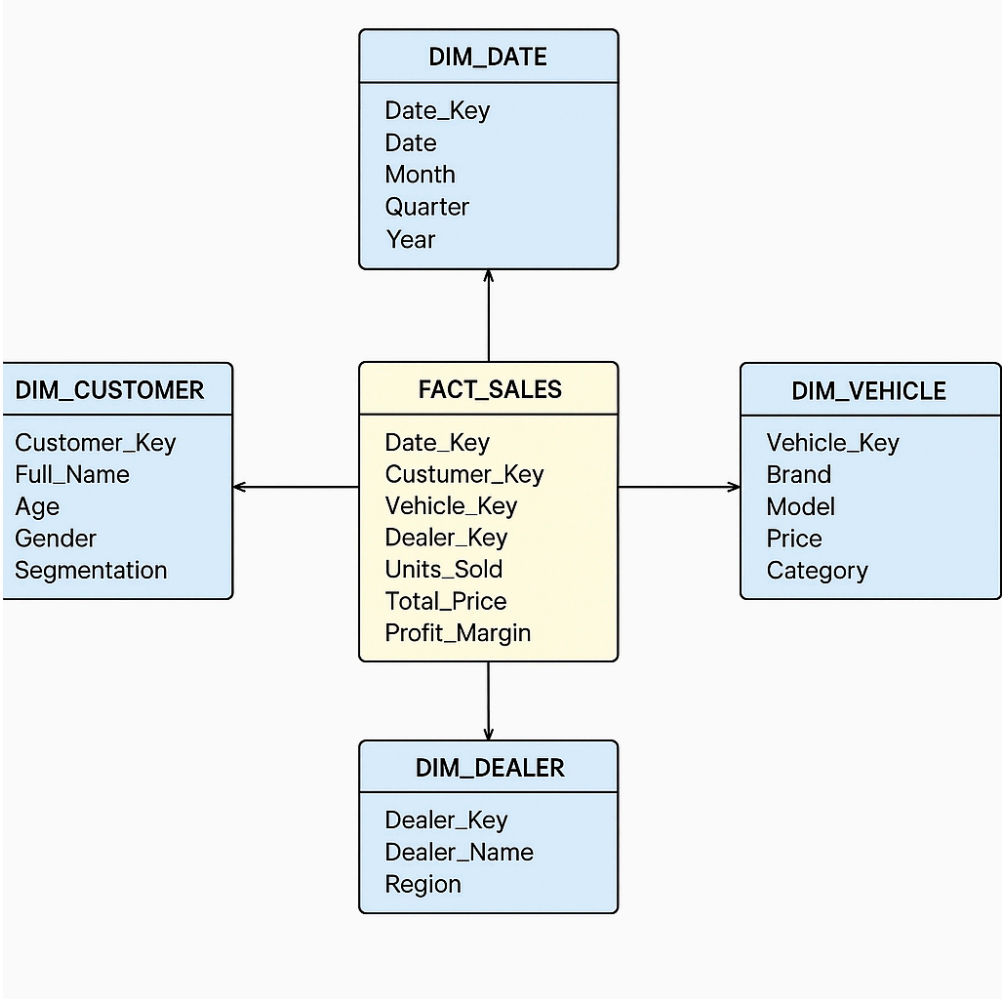
2.1. Model Skema: Star vs Snowflake

Dalam ranah data warehouse, terdapat dua metode utama dalam merancang skema multidimensional: Star Schema dan Snowflake Schema.

1. Star Schema memiliki desain yang langsung dengan tabel fakta yang berada di tengah dan tabel dimensi yang terhubung secara langsung ke tabel fakta. Tabel dimensi pada Star Schema biasanya tidak dinormalisasi, yang membuat akses data menjadi lebih cepat dan efisien bagi analisis.
2. Snowflake Schema merupakan pengembangan dari Star Schema dengan menerapkan normalisasi pada tabel-tabel dimensinya. Meskipun skema ini lebih rumit, ia menawarkan keuntungan dalam hal penghematan ruang.

Bagi analitik di PT Mobilita Nusantara, skema Star Schema dipilih karena:

1. Desainnya yang jelas dan mudah dimengerti.
2. Memungkinkan eksekusi kueri OLAP (Online Analytical Processing) dengan lebih cepat.
3. Cocok untuk kebutuhan pembuatan laporan dan tampilan visual data.



Gambar 2. Diagram Diagram Star Schema

2.2. Diagram Star Schema PT Mobilita Nusantara

Perancangan skema data warehouse untuk PT Mobilita Nusantara mengadopsi model Star Schema yang terdiri dari satu tabel fakta utama di tengah dan empat tabel dimensi yang berfungsi sebagai pendukung. Tabel utama tersebut adalah FACT_SALES yang menyimpan informasi mengenai transaksi penjualan, termasuk data seperti jumlah unit yang terjual, total nilai, dan margin laba. Tabel ini berinteraksi langsung dengan empat tabel dimensi melalui penggunaan kunci asing. Tabel DIM_DATE memberikan informasi temporal dengan atribut yang mencakup tanggal, bulan, kuartal, dan tahun. Tabel DIM_CUSTOMER menyimpan data mengenai pelanggan seperti nama, gender, usia, tingkat pendapatan, dan segmen pasar. Tabel DIM_VEHICLE berisi informasi tentang kendaraan yang terjual, mencakup merek, model, kategori, dan

harga. Di sisi lain, tabel DIM_DEALER memuat informasi terkait dealer termasuk nama dealer, area operasional, serta rating dealer. Struktur star schema ini diambil karena kemudahan, efisiensi dalam pemrosesan kueri, serta kesederhanaannya untuk diimplementasikan dalam visualisasi dan analisis bisnis yang berbasis OLAP. Desain ini berkontribusi untuk mempercepat proses analitik dan menjadikannya lebih teratur, membantu pengambilan keputusan yang didasarkan pada data historis yang tepat.

2.3. Justifikasi Pemilihan Star Schema

Pemilihan Star Schema untuk proyek ini didasarkan pada beberapa alasan kuat sebagai berikut:

1. Kecepatan Kueri: Star Schema mengurangi jumlah join yang diperlukan karena tidak ada normalisasi lebih lanjut, sehingga proses kueri menjadi lebih cepat.
2. Keterpahaman yang Tinggi: Desain yang sederhana membantu analis dan pengembang untuk dengan mudah memahami hubungan antar data.
3. Kemudahan Visualisasi: Sangat cocok untuk pelaporan dan dasbor BI karena struktur data yang jelas dan langsung.
4. Keselarasan dengan Data Mart: Star Schema sejalan dengan pendekatan dari bawah ke atas yang diterapkan dalam pengembangan data mart untuk penjualan dan preferensi pelanggan.

3. DESAIN FISIK: INDEKS DAN OPTIMASI AKSES

3.1. Konsep dan Jenis Indeks

Indeks dalam basis data adalah struktur data tambahan yang dibangun untuk mempercepat proses pencarian dan pengambilan

data pada tabel. Secara analogi, indeks berfungsi layaknya daftar isi pada sebuah buku, yang memungkinkan kita langsung menuju ke bagian tertentu tanpa harus membaca seluruh halaman satu per satu. Dengan adanya indeks, sistem manajemen basis data dapat mengurangi jumlah baris yang perlu dipindai saat menjalankan perintah seperti SELECT, JOIN, atau WHERE, sehingga kinerja query menjadi lebih efisien dan cepat.

Secara umum, terdapat beberapa jenis indeks yang sering digunakan dalam sistem manajemen basis data relasional, di antaranya:

a. Clustered Index

   Clustered index mengatur penyimpanan data secara fisik di dalam tabel sesuai dengan urutan nilai pada kolom yang diindeks. Artinya, susunan baris-baris data di tabel akan mengikuti urutan yang ditetapkan oleh clustered index tersebut. Karena data hanya dapat diurutkan secara fisik satu kali, maka dalam satu tabel hanya diperbolehkan memiliki satu clustered index saja.

b. Nonclustered Index

   Nonclustered index menyimpan indeks secara terpisah dari data fisik tabel. Setiap entri pada indeks ini memuat nilai dari kolom yang di indeks beserta penunjuk ke lokasi fisik baris data terkait. Berbeda dengan clustered index yang hanya dapat dibuat satu kali dalam sebuah tabel, nonclustered index dapat dibuat lebih dari satu pada satu tabel, sehingga memberikan fleksibilitas tinggi untuk mempercepat berbagai jenis query.

c. Columnstore Index

Columnstore index adalah tipe indeks yang menyimpan data secara fisik berdasarkan kolom, bukan baris seperti pada indeks tradisional. Pendekatan ini sangat cocok untuk kebutuhan analitik, terutama di lingkungan data warehouse, di mana query biasanya memproses data dalam jumlah besar, seperti operasi agregasi (SUM, AVG, COUNT), filter, dan pengelompokan data.

3.2. Penerapan Clustered dan Nonclustered Index

a. Clustered Index dalam Sistem Data Warehouse

Dalam konteks data warehouse PT Mobilita Nusantara, clustered index sangat bermanfaat untuk kolom yang memiliki urutan alami dan sering digunakan dalam pencarian berdasarkan waktu atau ID. Seperti pada tabel fakta FACT_SALES, clustered index dapat diterapkan pada kolom Date_Key atau Sales_ID untuk meningkatkan kecepatan analisis yang berfokus pada urutan waktu penjualan.

b. Nonclustered Index untuk Kolom Kunci Analisis

Pada tabel dimensi seperti DIM_CUSTOMER, DIM_DEALER, dan DIM_VEHICLE, nonclustered index dapat diterapkan untuk mempercepat pencarian berdasarkan kolom seperti Full_Name, Dealer_Region, atau Brand. Hal ini sangat berguna ketika pengguna ingin mencari data pelanggan atau menganalisis tren penjualan berdasarkan tipe kendaraan.

3.3. Columnstore Index untuk Analitik

Columnstore index memiliki peranan penting dalam sistem data warehouse modern karena mendukung analisis data besar dengan performa yang tinggi. Di PT Mobilita Nusantara, columnstore

index sangat tepat diterapkan pada tabel FACT_SALES yang menjadi fokus utama dalam analisis penjualan berdasarkan waktu, kendaraan, pelanggan, dan dealer. Terdapat beberapa manfaat untuk Columnstore Index yaitu:

- Mengurangi ukuran penyimpanan data karena kompresi kolom
- Mempercepat query agregasi seperti AVG(Total_Price), SUM(Units_Sold).
- Efisien untuk dashboard pelaporan yang memuat data besar

3.4. Contoh Kode SQL Pembuatan Indeks

a. Membuat Clustered Index

```
CREATE CLUSTERED INDEX idx_sales_id
ON FACT_SALES(Sales_ID);
```

b. Membuat Nonclustered Index

```
CREATE NONCLUSTERED INDEX idx_customer_name
ON DIM_CUSTOMER(Full_Name);
-- Membuat nonclustered index pada kolom merek
kendaraan
CREATE NONCLUSTERED INDEX idx_vehicle_brand
ON DIM_VEHICLE(Brand);
```

c. Membuat Clustered Columnstore Index

```
CREATE CLUSTERED COLUMNSTORE INDEX
idx_fact_sales_columnstore
ON FACT_SALES;
```

d. Membuat Nonclustered Columnstore Index

```
CREATE NONCLUSTERED COLUMNSTORE INDEX
idx_fact_sales_nonclustered
ON FACT_SALES(Total_Price, Units_Sold,
Profit_Margin);
```

4. DESAIN FISIK: STRATEGI PENYIMPANAN

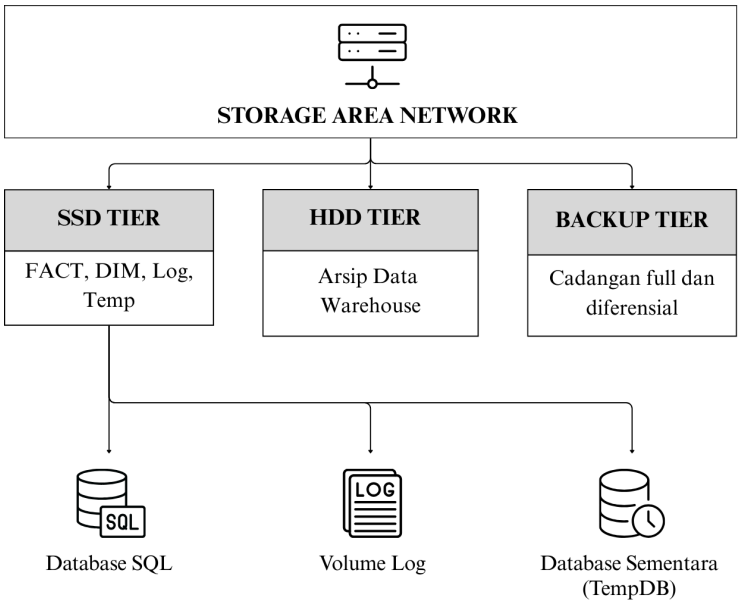
4.1. Arsitektur Penyimpanan DWH

Arsitektur penyimpanan yang disarankan adalah Hybrid Storage (SAN + SSD tiering) dengan RAID dan segregasi disk khusus berdasarkan jenis file. Untuk mendukung performa tinggi dan skalabilitas, arsitektur penyimpanan menggunakan SAN dengan tiered storage: data aktif seperti transaksi, dimensi, dan indeks disimpan di SSD, sementara arsip ditempatkan di HDD yang lebih ekonomis.

RAID 10 digunakan untuk partisi utama dan log transaksi karena performa dan redundansi tinggi, sedangkan RAID 5 digunakan untuk partisi backup yang lebih efisien dan cocok untuk akses baca sesekali. Arsitektur shared-disk SAN memungkinkan semua server SQL mengakses data dari satu pool disk dengan throughput tinggi dan jalur redundan melalui SAN switch.

Tabel 1. Rancangan Arsitektur

Komponen	Rekomendasi
Platform	SQL Server
Storage Medium	Storage Area Network (SAN) dengan tiered storage (SSD + HDD)
RAID Configuration	RAID 10 untuk data, RAID 5 untuk backup
Disk Type	SSD untuk data OLTP & DW, HDD untuk arsip & backup
IO Pattern	Random (data), Sequential (log), Mixed (tempdb)



Gambar 2. Diagram Arsitektur Penyimpanan

Diagram ini menunjukkan arsitektur penyimpanan fisik Data Warehouse berbasis SAN dengan tiered storage untuk mengoptimalkan performa sesuai jenis data dan pola akses. SAN Storage sebagai pusat terbagi menjadi tiga tier: SSD untuk data aktif seperti tabel fakta, dimensi, log transaksi, dan tempdb; HDD untuk data arsip yang jarang diakses; serta Backup Tier untuk file backup dengan media cepat agar proses restore efisien. Data dari tiap tier dialirkan ke komponen SQL Server yang sesuai SSD ke database utama, log transaksi, dan tempdb menjamin data sering pakai di storage cepat, sementara data historis dan backup ditempatkan di tier hemat biaya untuk performa dan efisiensi optimal.

4.2. Skema Kompresi Data

Untuk efisiensi penyimpanan dan performa kueri, digunakan kompresi berbasis page pada tabel fakta (FACT_SALES), karena ukurannya besar dan sering diakses dalam analisis OLAP. Page

compression mengurangi ukuran blok disk secara signifikan tanpa banyak overhead proses.

Tabel dimensi seperti DIM_CUSTOMER, DIM_VEHICLE, dan lainnya menggunakan row-level compression, cukup untuk data ukuran sedang yang diakses secara sering namun tidak berat.

Jika SQL Server mendukung, tabel arsip (misalnya data lebih dari 5 tahun) dapat disimpan menggunakan clustered columnstore indexes, yang memberikan keuntungan performa sangat tinggi untuk query besar dan scan intensif.

4.3. Pengaturan Filegroup dan Layout Storage

Struktur penyimpanan dibagi dalam beberapa filegroup terpisah:

- Filegroup DATA menyimpan semua tabel utama seperti FACT_SALES, DIM_CUSTOMER, dll. Ini ditempatkan di SSD pada RAID 10 untuk memastikan akses cepat.
- Filegroup INDEX menampung indeks-indeks non-clustered penting agar bisa diatur terpisah dan dioptimasi mandiri.
- Filegroup LOG untuk log transaksi (.ldf) berada pada volume SSD tersendiri dengan konfigurasi RAID 10 untuk menampung beban tulis berurutan yang tinggi.
- Filegroup TEMPDB menggunakan volume khusus yang didedikasikan, sangat penting karena tempdb digunakan intensif oleh SQL Server untuk proses sementara, sorting, dan pengolahan OLAP.
- Filegroup BACKUP menyimpan file backup terbaru di SSD/HDD cepat agar proses restore dapat dilakukan tanpa delay. Backup lama kemudian dipindahkan ke media penyimpanan lambat seperti tape atau cold-storage NAS sesuai kebijakan retensi data.

Setelah pengaturan filegroup ditetapkan untuk memisahkan beban kerja berdasarkan jenis data dan aktivitasnya, langkah selanjutnya adalah menentukan layout penyimpanan fisik yang sesuai. Layout storage ini bertujuan untuk mengimplementasikan strategi filegroup ke dalam struktur disk aktual.

Tabel 2. File Layout

Disk	Isi	Keterangan
Disk 1	OS & SQL Server binaries	Tidak campur dengan database
Disk 2	System DB (master, msdb, model)	Hindari konflik dengan user DB
Disk 3	.mdf dan .ndf untuk user DB (fakta dan dimensi)	Prioritaskan SSD
Disk 4	.ldf log files	Sequential write support
Disk 5	tempdb file	Gunakan SSD, scaling by core count
Disk 6	Backup	SSD/HDD cepat, arsip lama ke storage lambat/arsip NAS

5. DESAIN FISIK: PARTISI DAN VIEW

5.1. Partisi Tabel: Konsep dan Manfaat

Partisi tabel adalah teknik memecah satu tabel fakta besar menjadi beberapa segmen fisik (partisi) berdasarkan kriteria tertentu yang umumnya rentang tanggal atau rentang nilai kunci. Meskipun

partisi-partisi ini secara fisik terpisah, secara logis tetap tampil sebagai satu kesatuan tabel. Manfaat utamanya adalah:

- Peningkatan performa kueri: Dengan memisahkan data historis dan data terkini ke dalam partisi tersendiri, engine database hanya memindai partisi yang relevan sesuai filter kueri (partition elimination), sehingga I/O dan waktu respons berkurang drastis.
- Kemudahan pemeliharaan: Operasi bulk load, pengarsipan, maupun pembersihan data lama dapat dilakukan pada tingkat partisi, tanpa harus mengubah baris per baris di seluruh tabel menghemat waktu dan sumber daya.
- Manajemen siklus hidup data: Penghapusan data yang sudah kadaluarsa cukup dengan menjatuhkan (DROP) partisi, sehingga menghindari fragmentasi dan long-running transaction.

5.2. Implementasi Partisi pada FACT_SALES

Dalam data warehouse PT Mobilita Nusantara, tabel FACT_SALES mengelola jutaan baris transaksi penjualan. Dengan mempartisi tabel ini berdasarkan kolom Date_Key (misalnya per tahun atau per kuartal), proses ETL dan kueri analitik menjadi lebih efisien. Penerapan partisi mencakup:

- Penentuan kriteria partisi: Biasanya menggunakan atribut waktu (tahun, kuartal, atau bulan).
- Pemetaan ke filegroup atau tablespace terpisah: Setiap partisi disimpan pada storage area yang sesuai, untuk mempermudah backup/restore granular.
- Pengaturan default dan future partitions: Menyediakan satu partisi “catch-all” untuk data masa depan agar ETL tidak gagal ketika nilai Date_Key belum punya partisi khusus.

Melakukan pendefinisian Partition Function

```
CREATE PARTITION FUNCTION PF_DateKey_ByYear (INT)
AS RANGE RIGHT FOR VALUES
  (20221231,  -- partisi 1: Date_Key ≤ 20221231 (tahun
2022)
   20231231); -- partisi 2: Date_Key ≤ 20231231 (tahun
2023)
```

Membuat Partition Scheme

```
CREATE PARTITION SCHEME PS_DateKey_Year
AS PARTITION PF_DateKey_ByYear
TO (FG_2022, FG_2023, [PRIMARY]);
-- [PRIMARY] sebagai catch-all untuk data > 2023-12-31
```

Membuat Tabel FACT_SALES pada Partition Scheme

```
-- Buat staging table identik
CREATE TABLE dbo.FACT_SALES_Staging
(
  Date_Key       INT          NOT NULL,
  Customer_Key   INT          NOT NULL,
  Vehicle_Key    INT          NOT NULL,
  Dealer_Key     INT          NOT NULL,
  Units_Sold     INT          NULL,
  Total_Price    DECIMAL(18,2) NULL,
  Profit_Margin  DECIMAL(18,2) NULL
)
ON PS_DateKey_Year(Date_Key);

-- Switch per partisi
ALTER TABLE dbo.FACT_SALES_Staging
  SWITCH PARTITION 1 TO dbo.FACT_SALES PARTITION 1;
ALTER TABLE dbo.FACT_SALES_Staging
  SWITCH PARTITION 2 TO dbo.FACT_SALES PARTITION 2;
```

5.3. Partition Elimination dan Switching

Partition Elimination: Saat user menjalankan kueri dengan filter waktu (misal “WHERE Date_Key BETWEEN ’20230101’ AND ’20230331’”), optimizer database hanya akan membuka partisi kuartal I 2023. Ini memangkas beban baca dan mempercepat agregasi atau join.

```
SELECT
  D.Region,
  SUM(F.Units_Sold)   AS Total_Units,
  SUM(F.Total_Price)  AS Total_Revenue
FROM dbo.FACT_SALES F
JOIN dbo.DIM_DEALER D ON F.Dealer_Key = D.Dealer_Key
WHERE F.Date_Key BETWEEN 20230101 AND 20231231;
-- Optimizer hanya memindai partisi 2023
```

Partition Switching: Untuk memuat data periodik (bulan atau kuartal), data dimuat terlebih dahulu ke dalam tabel staging yang strukturnya identik dengan partisi target. Setelah lengkap, partisi tersebut di-switch secara atomik ke dalam FACT_SALES, menghindari overhead insert/delete individual baris dan menjaga konsistensi indeks.

5.4. Pembuatan Partitioned View

Partitioned view adalah metode untuk mensimulasikan partisi fisik pada database yang tidak mendukung partition scheme secara bawaan. Daripada satu tabel besar, data dibagi ke dalam beberapa tabel fisik masing-masing memuat subset berdasarkan rentang Date_Key dan kemudian digabungkan secara logis lewat sebuah view tunggal.

Membuat tabel per tahun (2022 dan 2023)

```
CREATE TABLE dbo.FACT_SALES_2022
(
  Date_Key       INT         NOT NULL,
  Customer_Key   INT         NOT NULL,
  Vehicle_Key    INT         NOT NULL,
  Dealer_Key     INT         NOT NULL,
  Units_Sold     INT         NULL,
  Total_Price    DECIMAL(18,2) NULL,
  Profit_Margin  DECIMAL(18,2) NULL,
  CONSTRAINT PK_FACT_SALES_2022 PRIMARY KEY CLUSTERED
(Date_Key, Customer_Key, Vehicle_Key, Dealer_Key),
  CONSTRAINT CK_FS_2022_Date CHECK (Date_Key BETWEEN
20220101 AND 20221231)
);

CREATE TABLE dbo.FACT_SALES_2023
(
  Date_Key       INT         NOT NULL,
```

```
  Customer_Key   INT          NOT NULL,
  Vehicle_Key    INT          NOT NULL,
  Dealer_Key     INT          NOT NULL,
  Units_Sold     INT          NULL,
  Total_Price    DECIMAL(18,2) NULL,
  Profit_Margin  DECIMAL(18,2) NULL,
  CONSTRAINT PK_FACT_SALES_2023 PRIMARY KEY CLUSTERED
(Date_Key, Customer_Key, Vehicle_Key, Dealer_Key),
  CONSTRAINT CK_FS_2023_Date CHECK (Date_Key BETWEEN
20230101 AND 20231231)
);
```

Membuat view gabungan

```
CREATE VIEW dbo.V_FACT_SALES
AS
  SELECT Date_Key, Customer_Key, Vehicle_Key,
Dealer_Key, Units_Sold, Total_Price, Profit_Margin
    FROM dbo.FACT_SALES_2022
  UNION ALL
  SELECT Date_Key, Customer_Key, Vehicle_Key,
Dealer_Key, Units_Sold, Total_Price, Profit_Margin
    FROM dbo.FACT_SALES_2023;
```

6. Kesimpulan

Laporan ini menyajikan perancangan data warehouse untuk PT Mobilita Nusantara yang bertujuan memenuhi kebutuhan strategis bisnis akan pengambilan keputusan berbasis data historis, pemantauan tren, dan personalisasi strategi. Dengan pendekatan bottom-up yang berfokus pada data mart penjualan dan preferensi pelanggan, struktur data warehouse dirancang menggunakan model multidimensi dengan Skema Bintang. Skema ini menempatkan tabel fakta FACT_SALES sebagai pusat, yang merekam metrik penjualan dan terhubung dengan tabel dimensi DIM_DATE, DIM_CUSTOMER, DIM_DEALER, serta DIM_VEHICLE untuk menyediakan konteks analisis. Pemilihan Skema Bintang didasarkan pada kesederhanaan, efisiensi kueri, dukungan visualisasi, dan kesesuaiannya untuk pengembangan data mart. Aspek desain fisik, meliputi strategi pengindeksan (termasuk Columnstore Index), strategi penyimpanan, dan partisi tabel FACT_SALES, juga telah dipertimbangkan

secara cermat untuk mengoptimalkan kinerja dan pengelolaan data. Secara keseluruhan, implementasi desain data warehouse ini diharapkan akan PT Mobilita Nusantara akan menghasilkan kemampuan analisis yang kuat, memungkinkan pengambilan keputusan yang lebih cepat dan tepat sasaran berdasarkan data historis yang terstruktur dan terintegrasi.