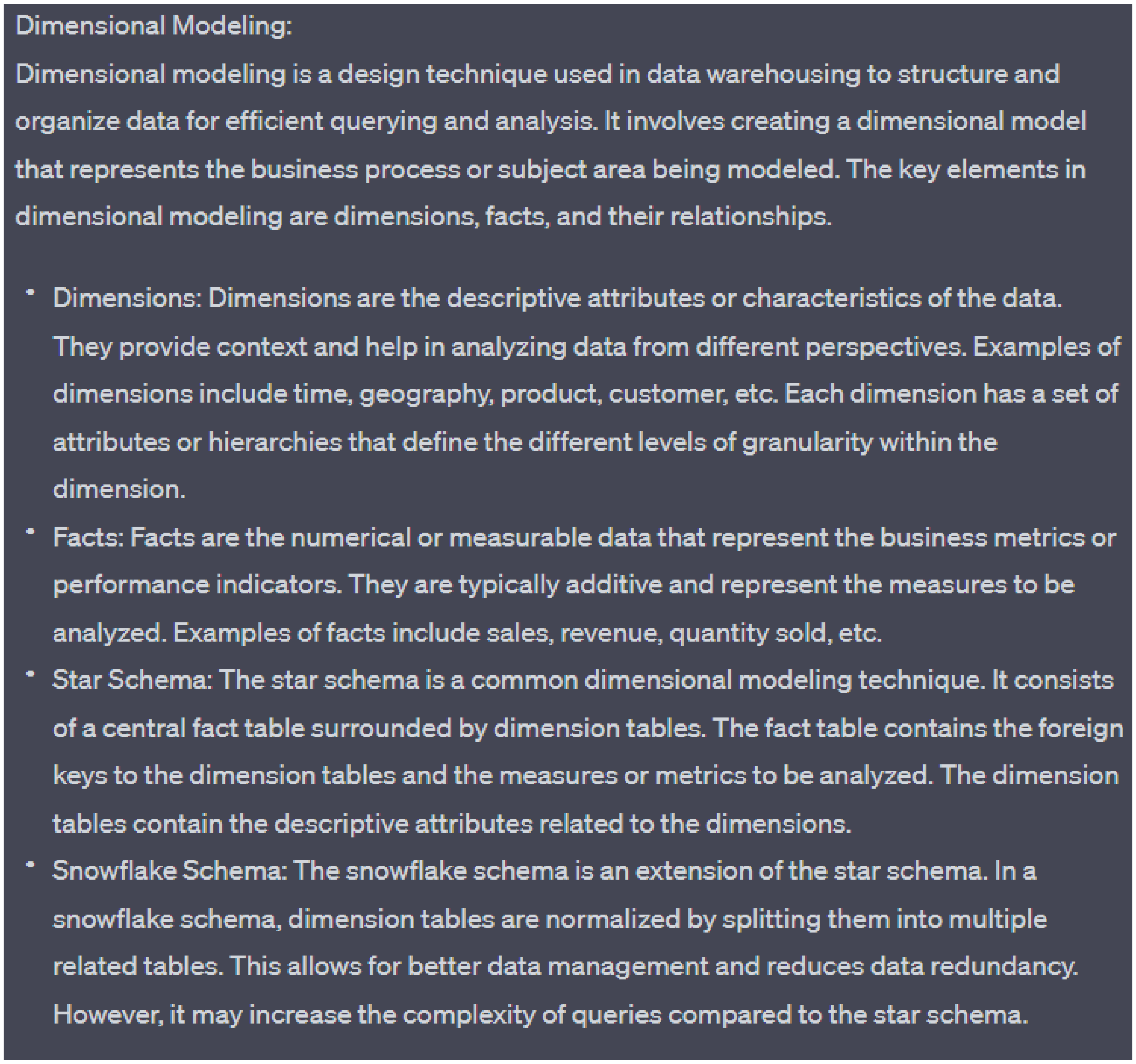
Mod-3

Unit-3.4

Dimensional modelling



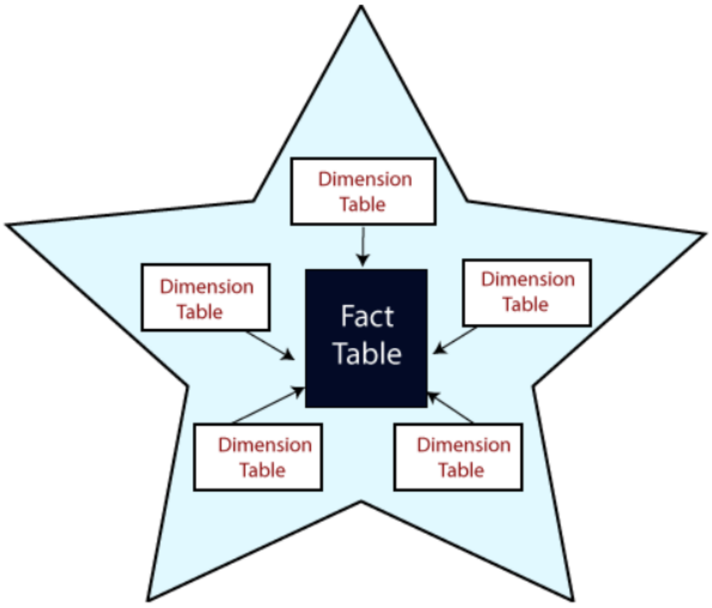
Dimensional Modeling:

Dimensional modeling is a design technique used in data warehousing to structure and organize data for efficient querying and analysis. It involves creating a dimensional model that represents the business process or subject area being modeled. The key elements in dimensional modeling are dimensions, facts, and their relationships.

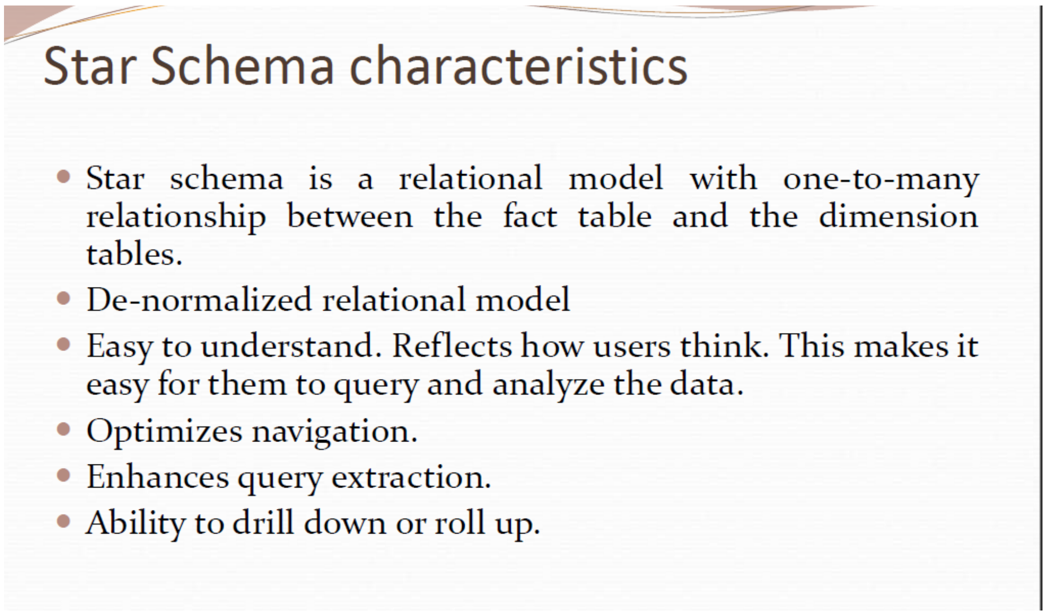
- Dimensions: Dimensions are the descriptive attributes or characteristics of the data. They provide context and help in analyzing data from different perspectives. Examples of dimensions include time, geography, product, customer, etc. Each dimension has a set of attributes or hierarchies that define the different levels of granularity within the dimension.
- Facts: Facts are the numerical or measurable data that represent the business metrics or performance indicators. They are typically additive and represent the measures to be analyzed. Examples of facts include sales, revenue, quantity sold, etc.
- Star Schema: The star schema is a common dimensional modeling technique. It consists of a central fact table surrounded by dimension tables. The fact table contains the foreign keys to the dimension tables and the measures or metrics to be analyzed. The dimension tables contain the descriptive attributes related to the dimensions.
- Snowflake Schema: The snowflake schema is an extension of the star schema. In a snowflake schema, dimension tables are normalized by splitting them into multiple related tables. This allows for better data management and reduces data redundancy. However, it may increase the complexity of queries compared to the star schema.

Star Schema:

A star schema is the elementary form of a dimensional model, in which data are organized into **facts** and **dimensions**. A fact is an event that is counted or measured, such as a sale or log in. A dimension includes reference data about the fact, such as date, item, or customer.

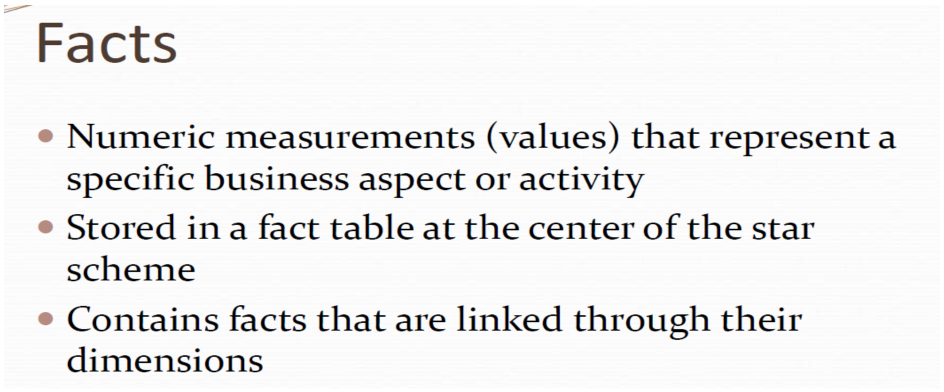


Star Schema



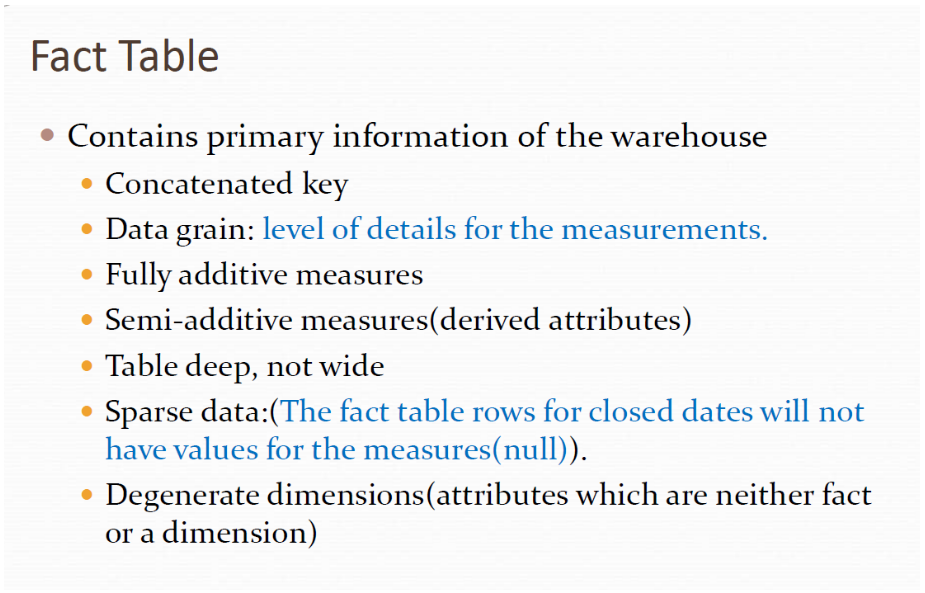
Star Schema characteristics

- Star schema is a relational model with one-to-many relationship between the fact table and the dimension tables.
- De-normalized relational model
- Easy to understand. Reflects how users think. This makes it easy for them to query and analyze the data.
- Optimizes navigation.
- Enhances query extraction.
- Ability to drill down or roll up.



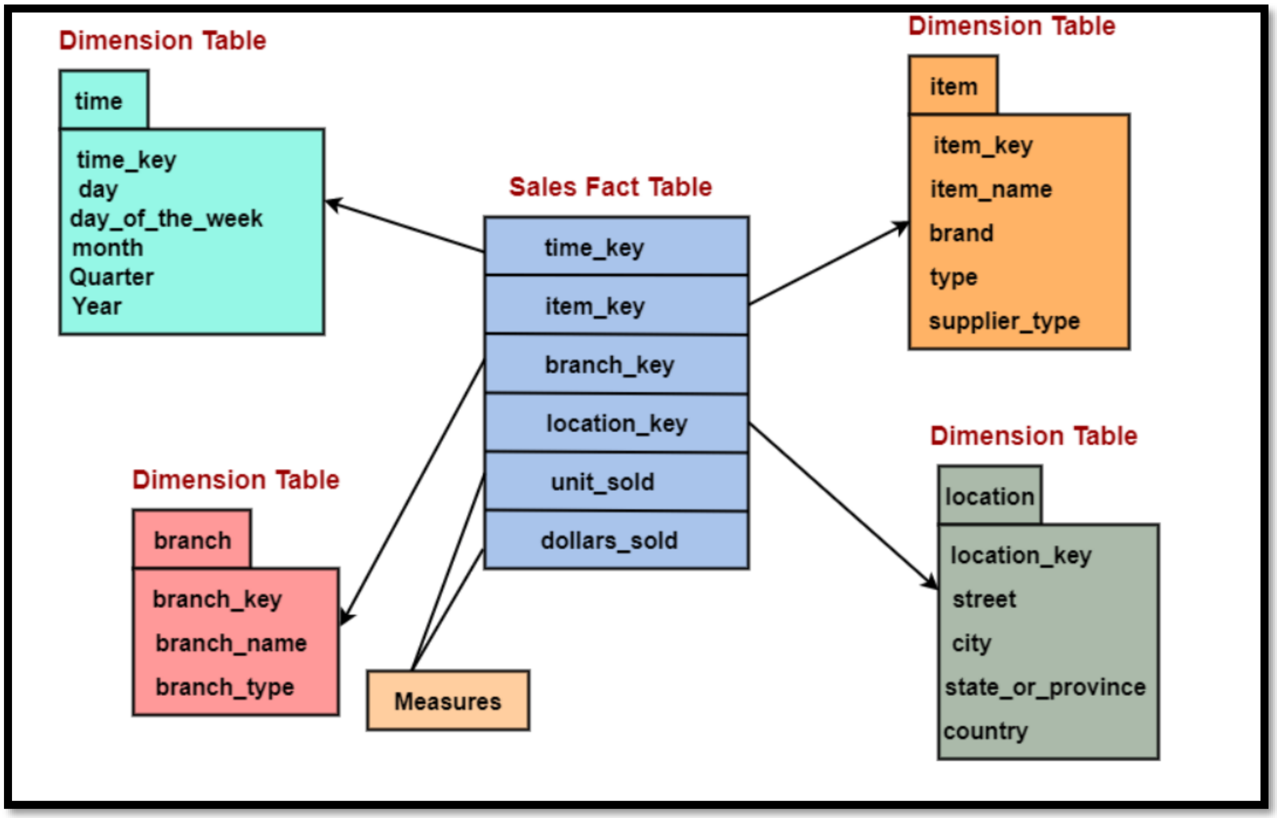
Facts

- Numeric measurements (values) that represent a specific business aspect or activity
- Stored in a fact table at the center of the star scheme
- Contains facts that are linked through their dimensions

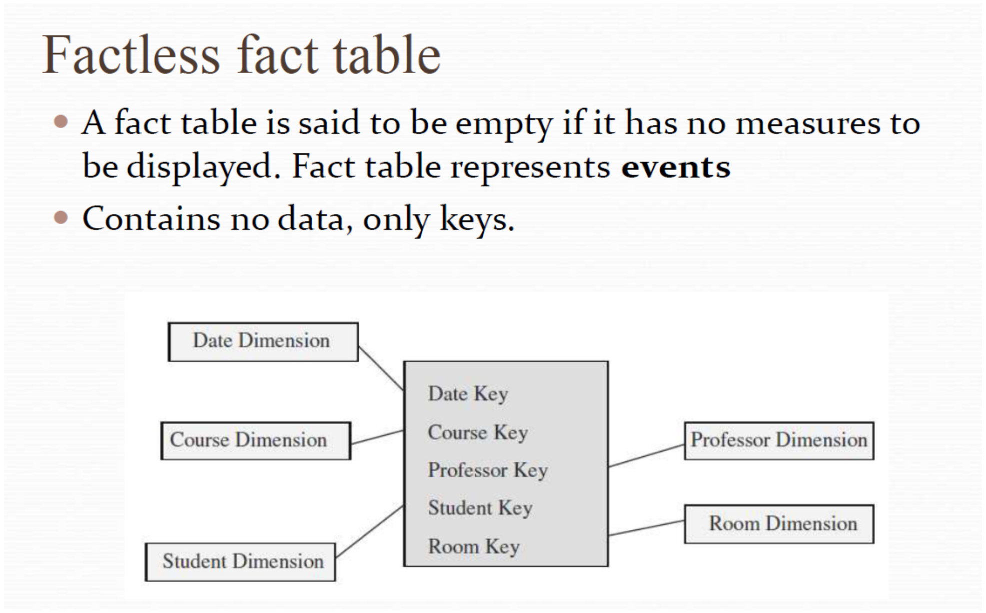


Fact Table

- Contains primary information of the warehouse
  - Concatenated key
  - Data grain: level of details for the measurements.
  - Fully additive measures
  - Semi-additive measures(derived attributes)
  - Table deep, not wide
  - Sparse data:(The fact table rows for closed dates will not have values for the measures(null)).
  - Degenerate dimensions(attributes which are neither fact or a dimension)



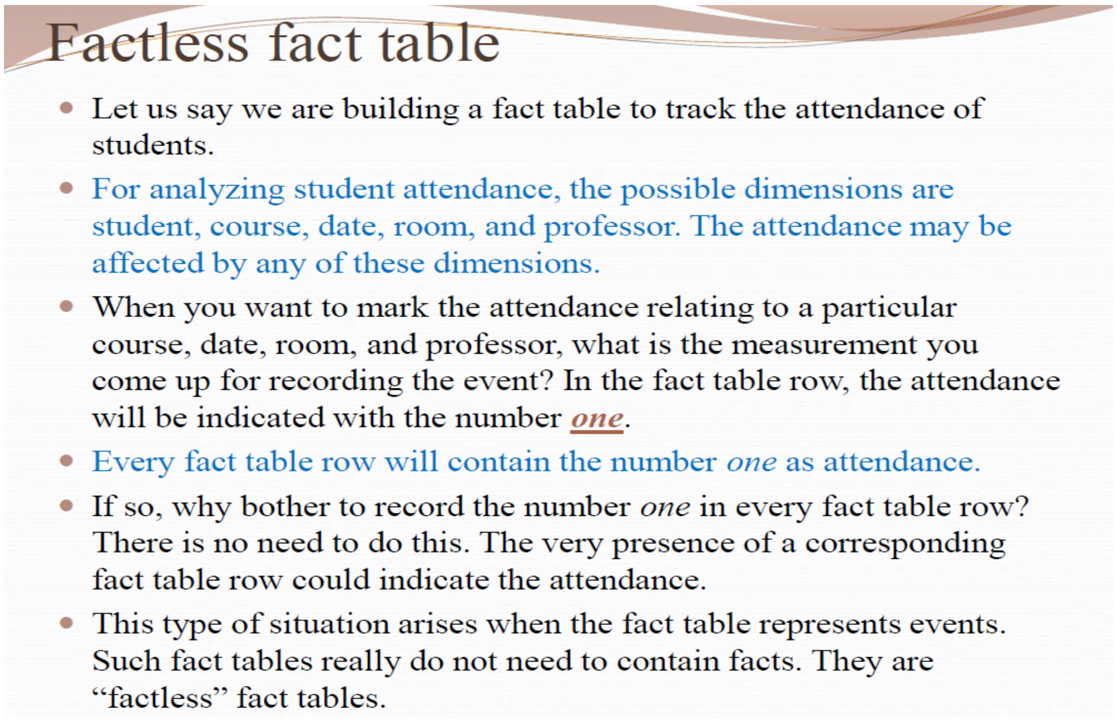
Factless Fact-Table:



Factless fact table

- A fact table is said to be empty if it has no measures to be displayed. Fact table represents **events**
- Contains no data, only keys.

Date Dimension
Course Dimension
Student Dimension
Date Key
Course Key
Professor Key
Student Key
Room Key
Professor Dimension
Room Dimension



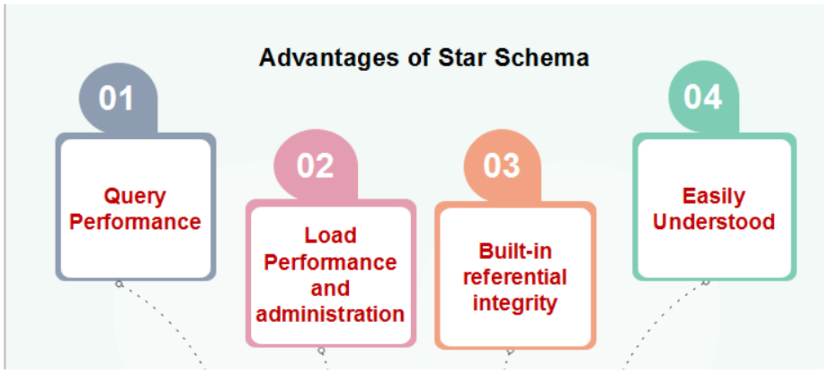
Factless fact table

- Let us say we are building a fact table to track the attendance of students.
- For analyzing student attendance, the possible dimensions are student, course, date, room, and professor. The attendance may be affected by any of these dimensions.
- When you want to mark the attendance relating to a particular course, date, room, and professor, what is the measurement you come up for recording the event? In the fact table row, the attendance will be indicated with the number ***one***.
- Every fact table row will contain the number *one* as attendance.
- If so, why bother to record the number *one* in every fact table row? There is no need to do this. The very presence of a corresponding fact table row could indicate the attendance.
- This type of situation arises when the fact table represents events. Such fact tables really do not need to contain facts. They are "factless" fact tables.

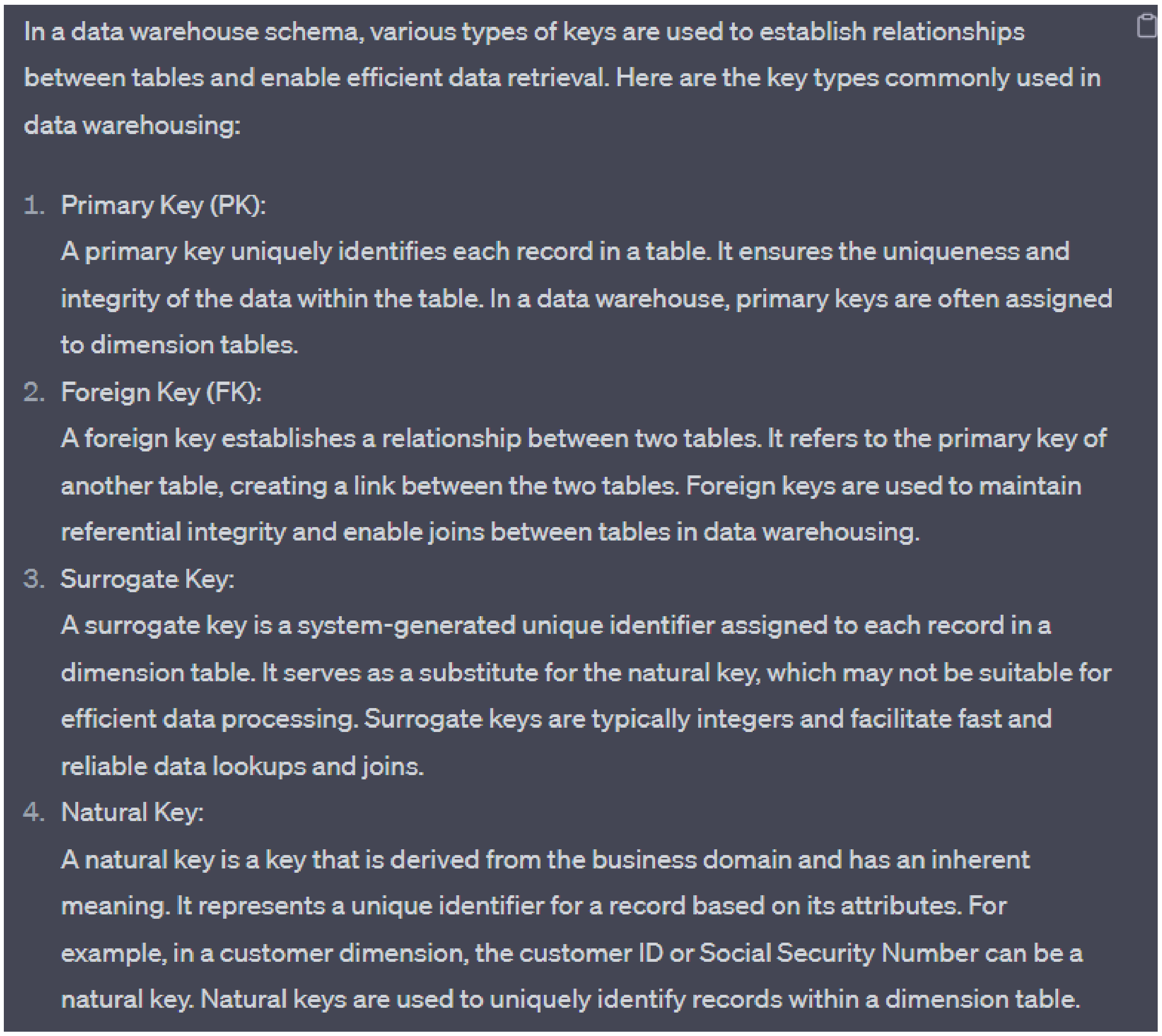
Advantages of Star Schema

Star Schemas are easy for end-users and application to understand and navigate. With a well-designed schema, the customer can instantly analyze large, multidimensional data sets.

The main advantage of star schemas in a decision-support environment are:


Advantages of Star Schema
01
Query Performance
02
Load Performance and administration
03
Built-in referential integrity
04
Easily Understood

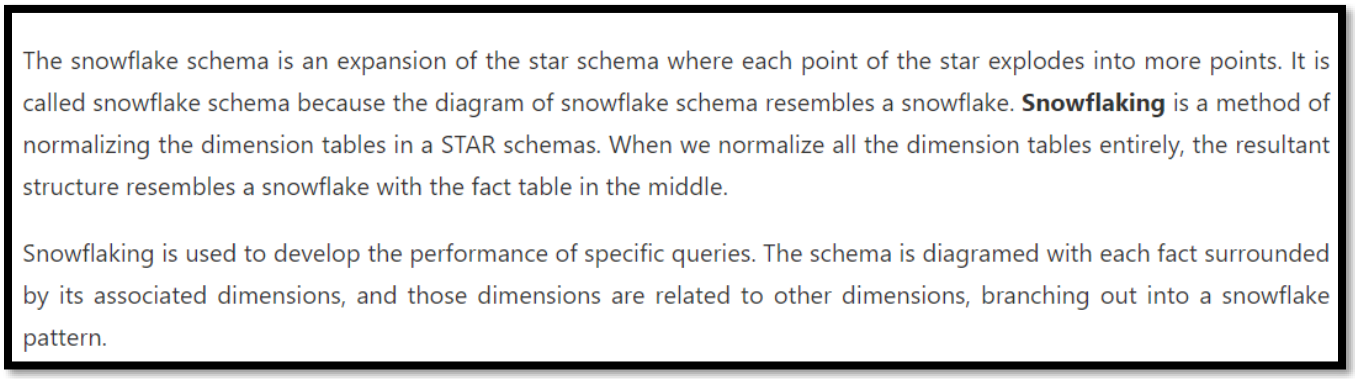
Keys in data warehouse schema:



In a data warehouse schema, various types of keys are used to establish relationships between tables and enable efficient data retrieval. Here are the key types commonly used in data warehousing:

1. Primary Key (PK):
   A primary key uniquely identifies each record in a table. It ensures the uniqueness and integrity of the data within the table. In a data warehouse, primary keys are often assigned to dimension tables.
2. Foreign Key (FK):
   A foreign key establishes a relationship between two tables. It refers to the primary key of another table, creating a link between the two tables. Foreign keys are used to maintain referential integrity and enable joins between tables in data warehousing.
3. Surrogate Key:
   A surrogate key is a system-generated unique identifier assigned to each record in a dimension table. It serves as a substitute for the natural key, which may not be suitable for efficient data processing. Surrogate keys are typically integers and facilitate fast and reliable data lookups and joins.
4. Natural Key:
   A natural key is a key that is derived from the business domain and has an inherent meaning. It represents a unique identifier for a record based on its attributes. For example, in a customer dimension, the customer ID or Social Security Number can be a natural key. Natural keys are used to uniquely identify records within a dimension table.

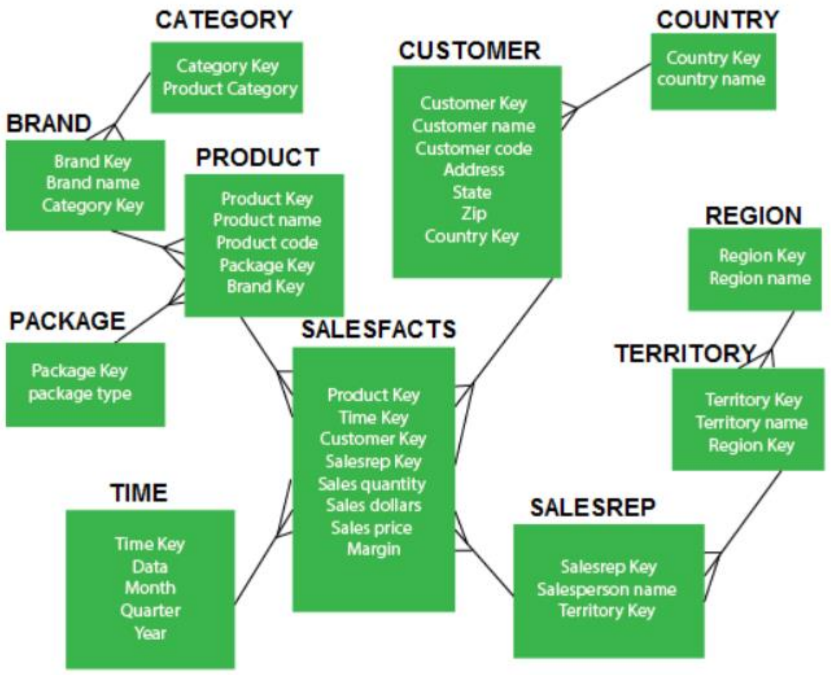
Snowflake schema:



The snowflake schema is an expansion of the star schema where each point of the star explodes into more points. It is called snowflake schema because the diagram of snowflake schema resembles a snowflake. **Snowflaking** is a method of normalizing the dimension tables in a STAR schemas. When we normalize all the dimension tables entirely, the resultant structure resembles a snowflake with the fact table in the middle.

Snowflaking is used to develop the performance of specific queries. The schema is diagramed with each fact surrounded by its associated dimensions, and those dimensions are related to other dimensions, branching out into a snowflake pattern.

The STAR schema for sales, as shown above, contains only five tables, whereas the normalized version now extends to eleven tables. We will notice that in the snowflake schema, the attributes with low cardinality in each original dimension tables are removed to form separate tables. These new tables are connected back to the original dimension table through artificial keys.



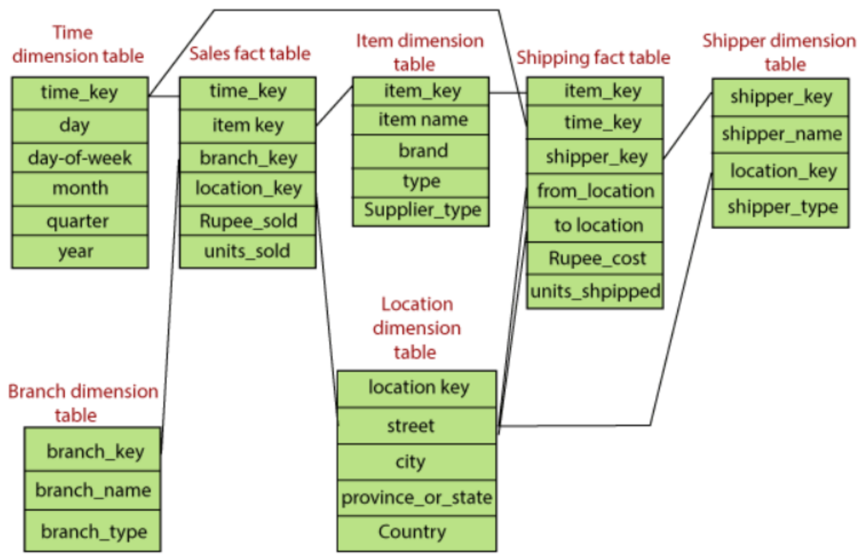
Snowflake Schema

Fact Constellation Schema:

A Fact constellation means two or more fact tables sharing one or more dimensions. It is also called **Galaxy schema**.

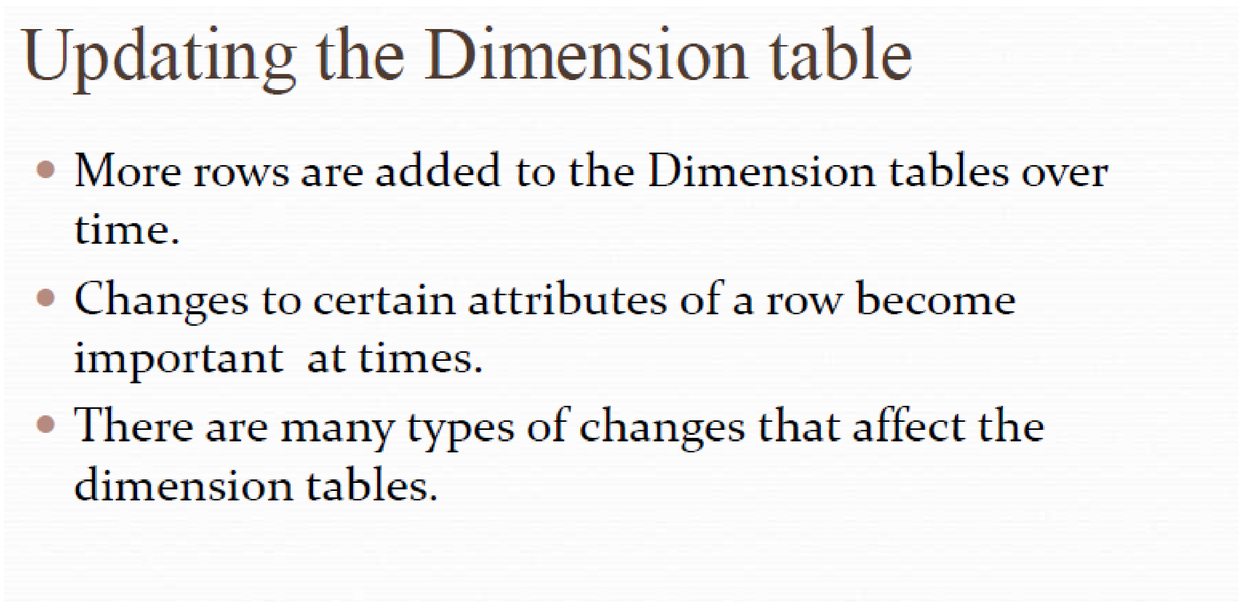
Fact Constellation Schema describes a logical structure of data warehouse or data mart. Fact Constellation Schema can design with a collection of de-normalized FACT, Shared, and Conformed Dimension tables.

Example: A fact constellation schema is shown in the figure below.



This schema defines two fact tables, sales, and shipping. Sales are treated along four dimensions, namely, time, item, branch, and location. The schema contains a fact table for sales that includes keys to each of the four dimensions, along with two measures: Rupee_sold and units_sold. The shipping table has five dimensions, or keys: item_key, time_key, shipper_key, from_location, and to_location, and two measures: Rupee_cost and units_shipped.

The primary disadvantage of the fact constellation schema is that it is a more challenging design because many variants for specific kinds of aggregation must be considered and selected.



Updating the Dimension table

- More rows are added to the Dimension tables over time.
- Changes to certain attributes of a row become important at times.
- There are many types of changes that affect the dimension tables.

Slowly changing Dimensions:

Slowly Changing Dimensions

What is a Slowly Changing Dimension?

A Slowly Changing Dimension (SCD) is a dimension that stores and manages both current and historical data over time in a data warehouse. It is considered and implemented as one of the most critical ETL tasks in tracking the history of dimension records.

Type 1 changes: Correction of Errors

- These changes relate to correction of errors in source systems.
- These changes do not have any significance in the source systems.
- The old value needs to be discarded.
- New value overwrites the old value in the source system.
- Such changes in the source system need not be preserved in the data warehouse.

Customer key	Customer ID	Customer Name	Marital status	Address
-	-	-	-	-
22522134	C12345	July Michael	Single	AAAAAA A

Customer key	Customer ID	Customer Name	Marital status	Address
-	-	-	-	-
22522134	C12345	July Michel	Single	AAAAAA A

Method for applying Type 1 change

Type 2 Changes: Preservation of History

- These changes relate to true changes in source system.
- The history must be preserved in the data warehouse.
- These changes cause the history to be partitioned in the data warehouse.
- Every change that occurs in the attribute value must be preserved.

Customer key	Customer ID	Customer Name	Marital status	Address
-	-	-	-	-
22522134	C12345	Jenny david	Single	AAAAAAA
-	-	-	-	-

Customer key	Customer ID	Customer Name	Marital status	Address	Effective Date
-	-	-	-	-	-
22522134	C12345	Jenny david	Single	AAAAAAA	1/Mar/03
-	-	-	-	-	-
22522134	C12345	Jenny david	Married	AAAAAAA	16/Jan/06

Type 3 changes: Tentative Soft Revisions

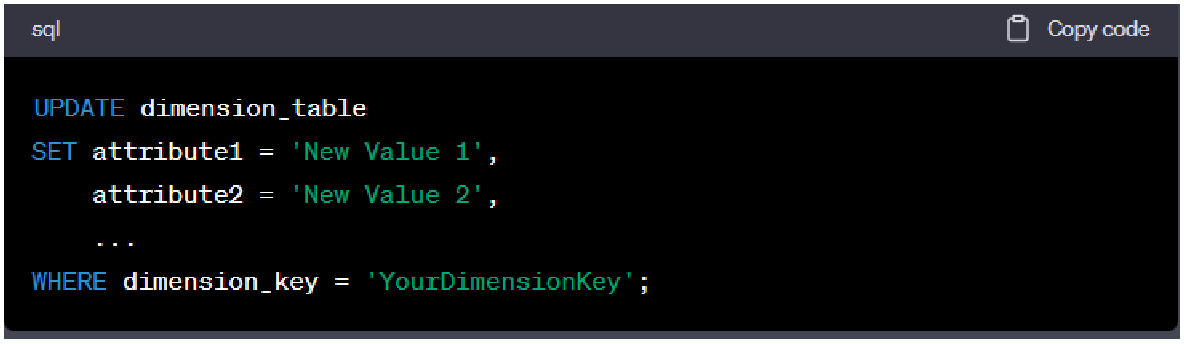
How Type 3 changes applied to the data stored in the datawarehouse

- There is a need to track history with both old and new value of the same attribute.
- Type 3 changes are used to compare performance across the transition.
- They enable the users to track data in both forward and backward directions.
- An "old" field is added in the dimension table for the affected attribute.
- The existing value of the attribute is pushed down from the "current" field to the "old" field.
- The new value of the attribute is kept in the "current" field.

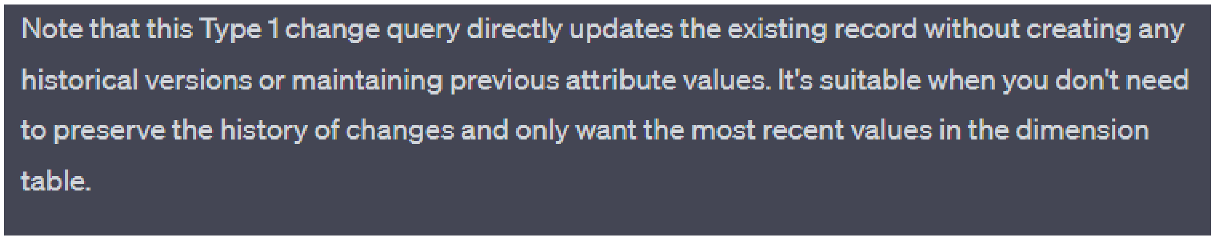
Salesperson key	Salesperson ID	Salesperson Name	Old Location	Current Location	Effective date
-	-	-	-	-	
22522134	C12345	Jenny david		Delhi	1/Mar/03
-	-	-	-	-	

Salesperson key	Salesperson ID	Salesperson Name	Old Location	Current Location	Effective date
-	-	-	-	-	
22522134	C12345	Jenny david	Delhi	Mumbai	1/Mar/03
-	-	-	-	-	

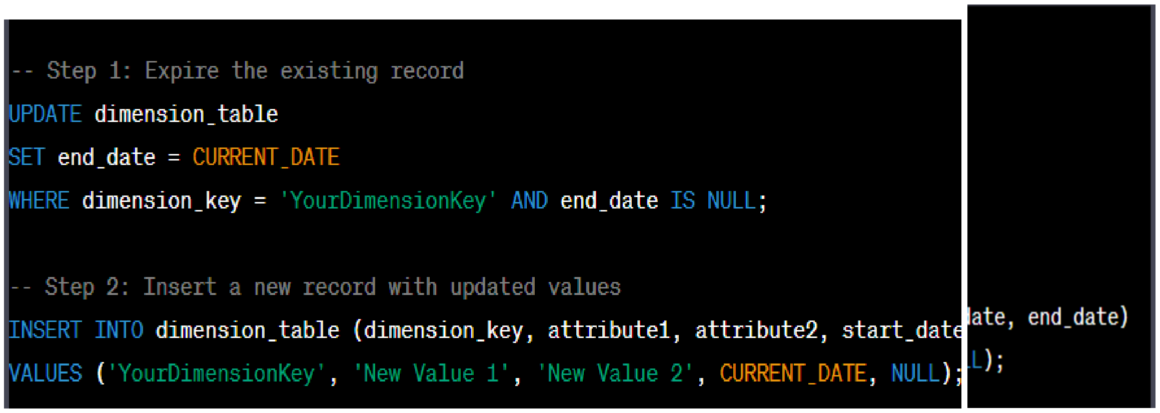
Method for applying type 3 changes



```sql
UPDATE dimension_table
SET attribute1 = 'New Value 1',
    attribute2 = 'New Value 2',
    ...
WHERE dimension_key = 'YourDimensionKey';
```

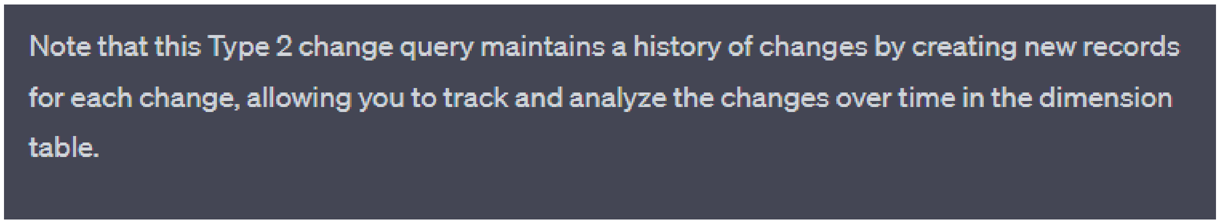


Note that this Type 1 change query directly updates the existing record without creating any historical versions or maintaining previous attribute values. It's suitable when you don't need to preserve the history of changes and only want the most recent values in the dimension table.

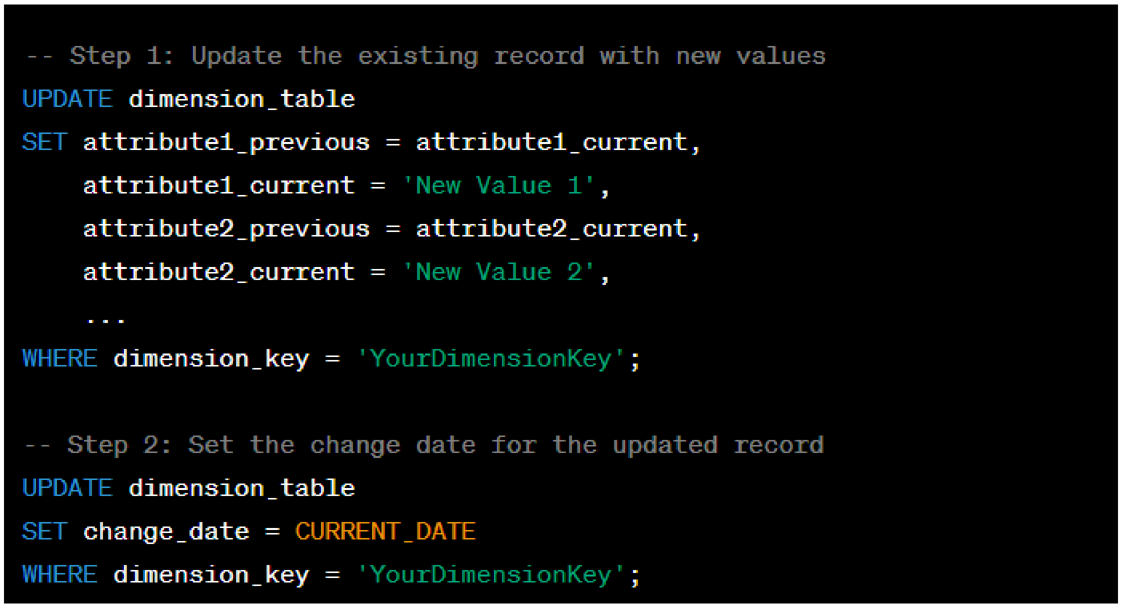


```sql
-- Step 1: Expire the existing record
UPDATE dimension_table
SET end_date = CURRENT_DATE
WHERE dimension_key = 'YourDimensionKey' AND end_date IS NULL;

-- Step 2: Insert a new record with updated values
INSERT INTO dimension_table (dimension_key, attribute1, attribute2, start_dateate, end_date)
VALUES ('YourDimensionKey', 'New Value 1', 'New Value 2', CURRENT_DATE, NULL);L);
```

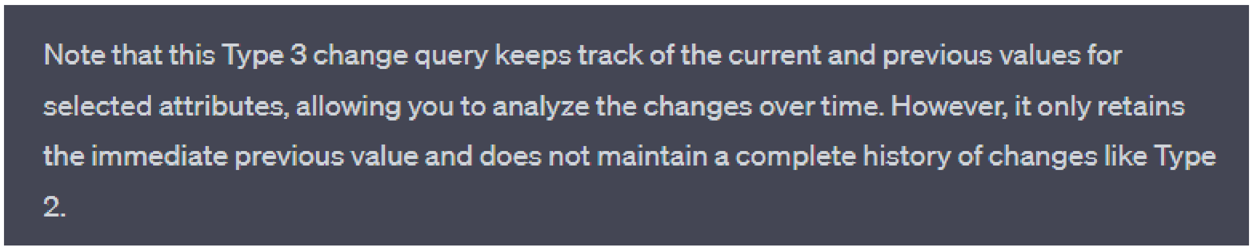


Note that this Type 2 change query maintains a history of changes by creating new records for each change, allowing you to track and analyze the changes over time in the dimension table.



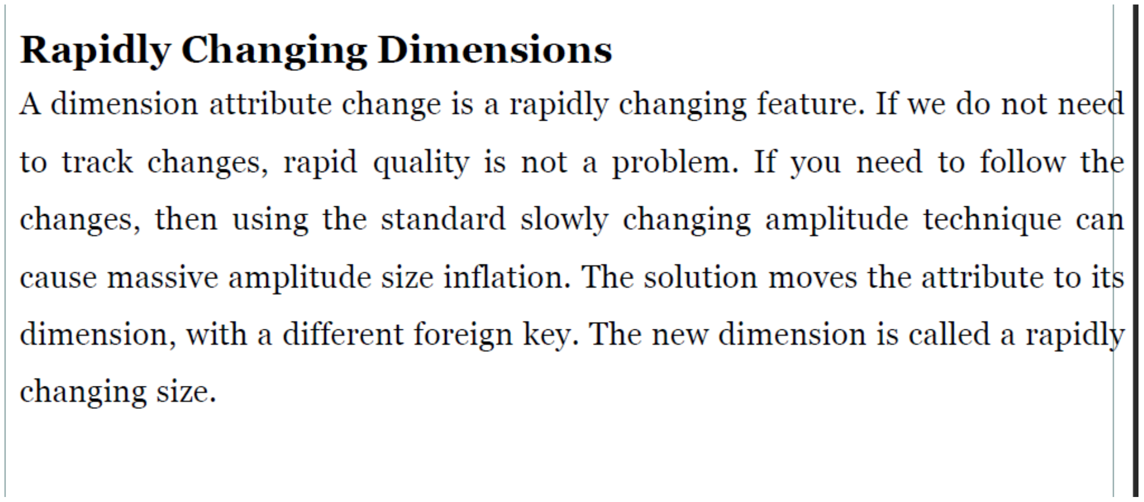
```sql
-- Step 1: Update the existing record with new values
UPDATE dimension_table
SET attribute1_previous = attribute1_current,
    attribute1_current = 'New Value 1',
    attribute2_previous = attribute2_current,
    attribute2_current = 'New Value 2',
    ...
WHERE dimension_key = 'YourDimensionKey';

-- Step 2: Set the change date for the updated record
UPDATE dimension_table
SET change_date = CURRENT_DATE
WHERE dimension_key = 'YourDimensionKey';
```



Note that this Type 3 change query keeps track of the current and previous values for selected attributes, allowing you to analyze the changes over time. However, it only retains the immediate previous value and does not maintain a complete history of changes like Type 2.

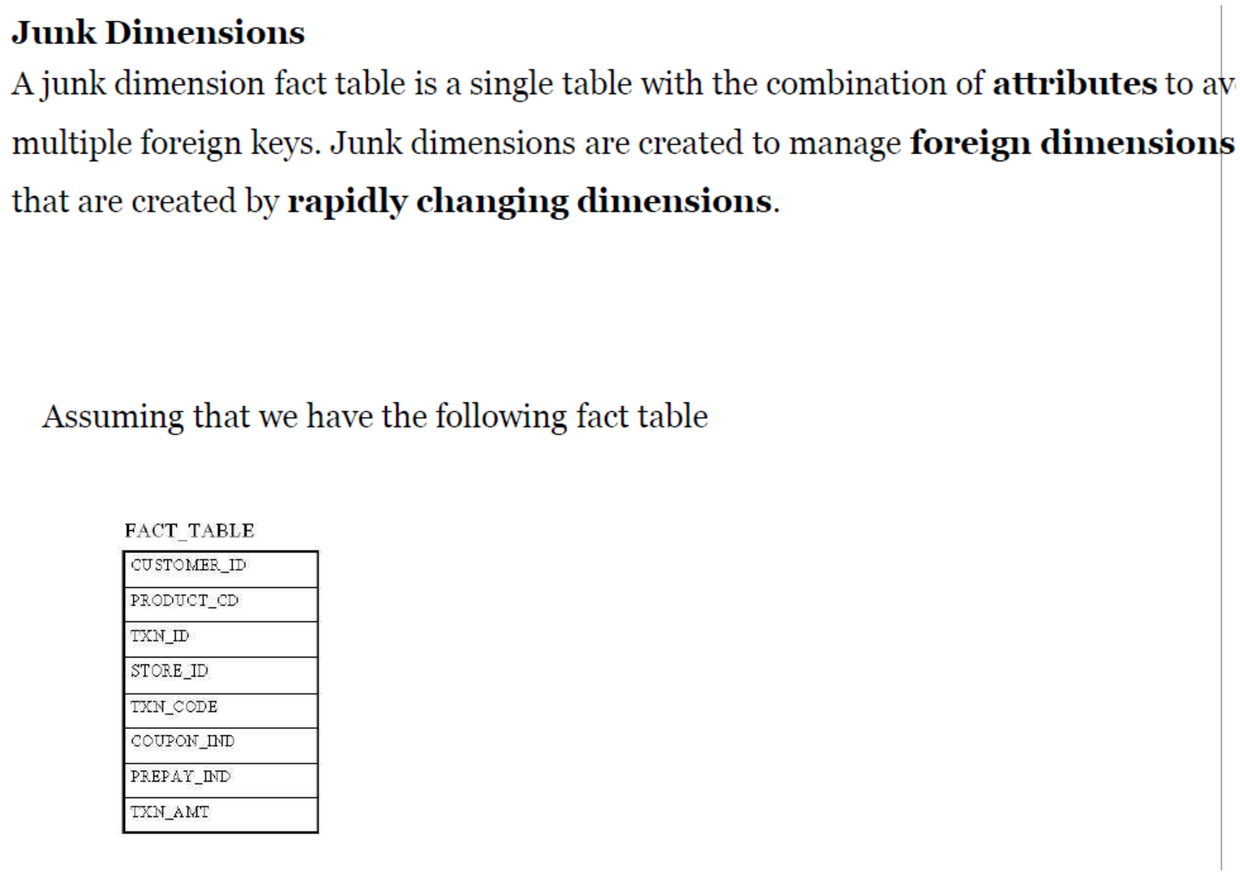
Rapidly changing Dimensions:



Rapidly Changing Dimensions

A dimension attribute change is a rapidly changing feature. If we do not need to track changes, rapid quality is not a problem. If you need to follow the changes, then using the standard slowly changing amplitude technique can cause massive amplitude size inflation. The solution moves the attribute to its dimension, with a different foreign key. The new dimension is called a rapidly changing size.

Junk changing Dimensions:

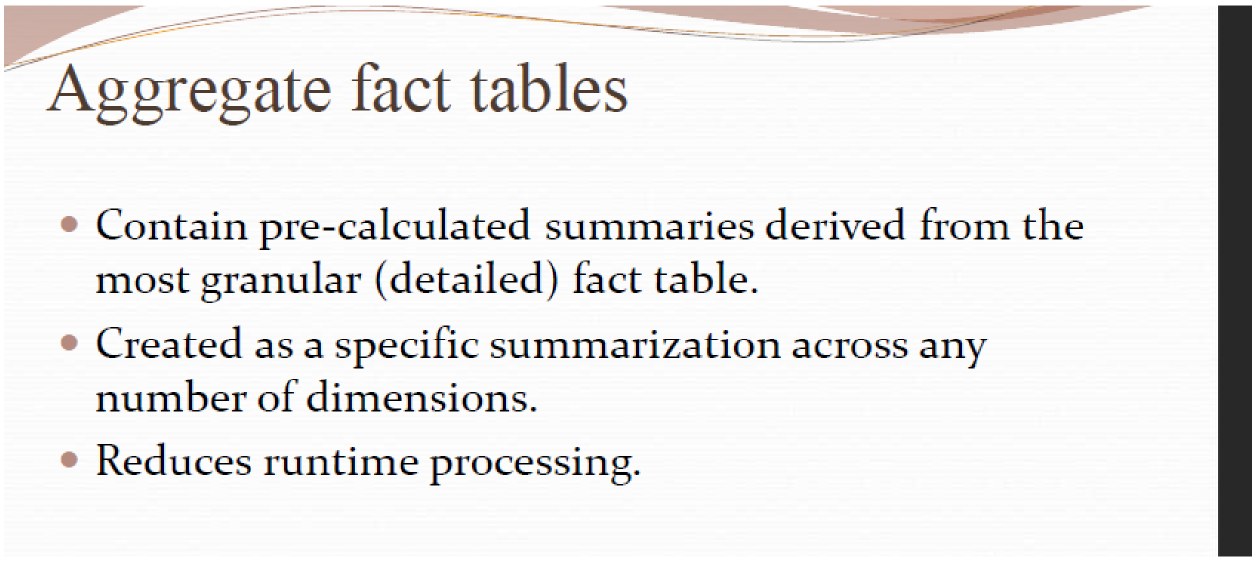


Junk Dimensions

A junk dimension fact table is a single table with the combination of **attributes** to av multiple foreign keys. Junk dimensions are created to manage **foreign dimensions** that are created by **rapidly changing dimensions**.

Assuming that we have the following fact table

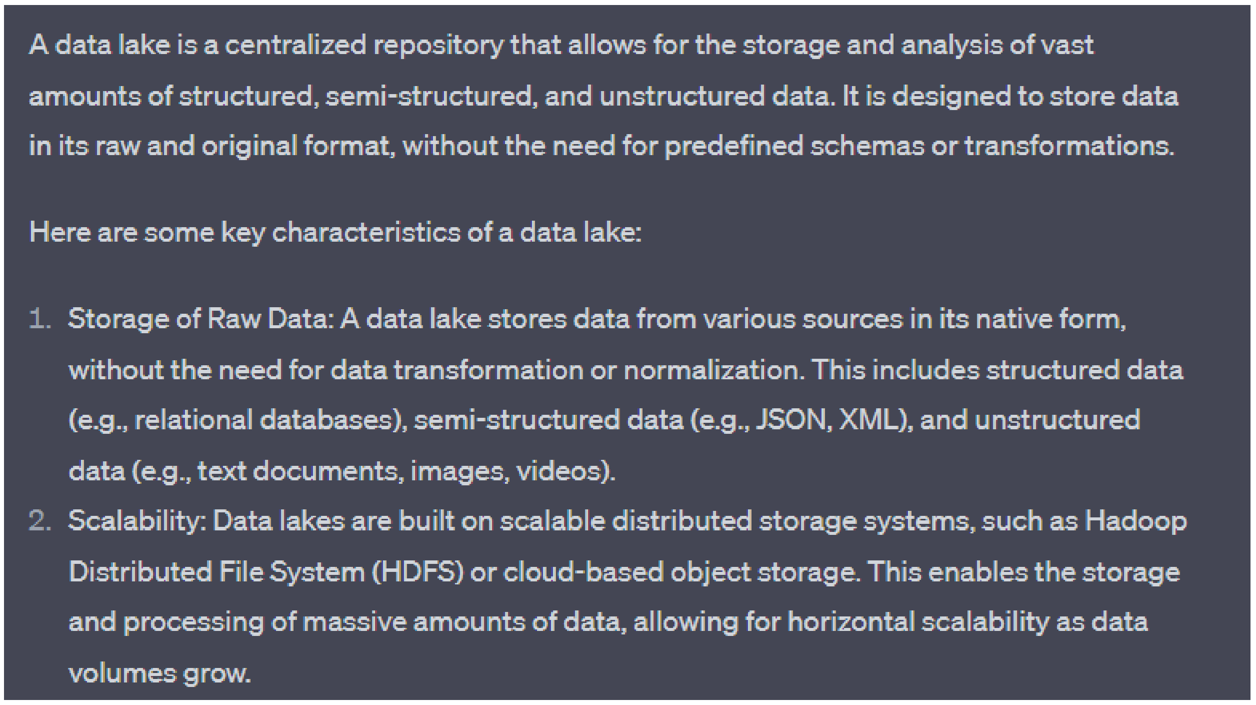
FACT_TABLE
CUSTOMER_ID
PRODUCT_CD
TXN_ID
STORE_ID
TXN_CODE
COUPON_IND
PREPAY_IND
TXN_AMT



Aggregate fact tables

- Contain pre-calculated summaries derived from the most granular (detailed) fact table.
- Created as a specific summarization across any number of dimensions.
- Reduces runtime processing.

Data Lake:

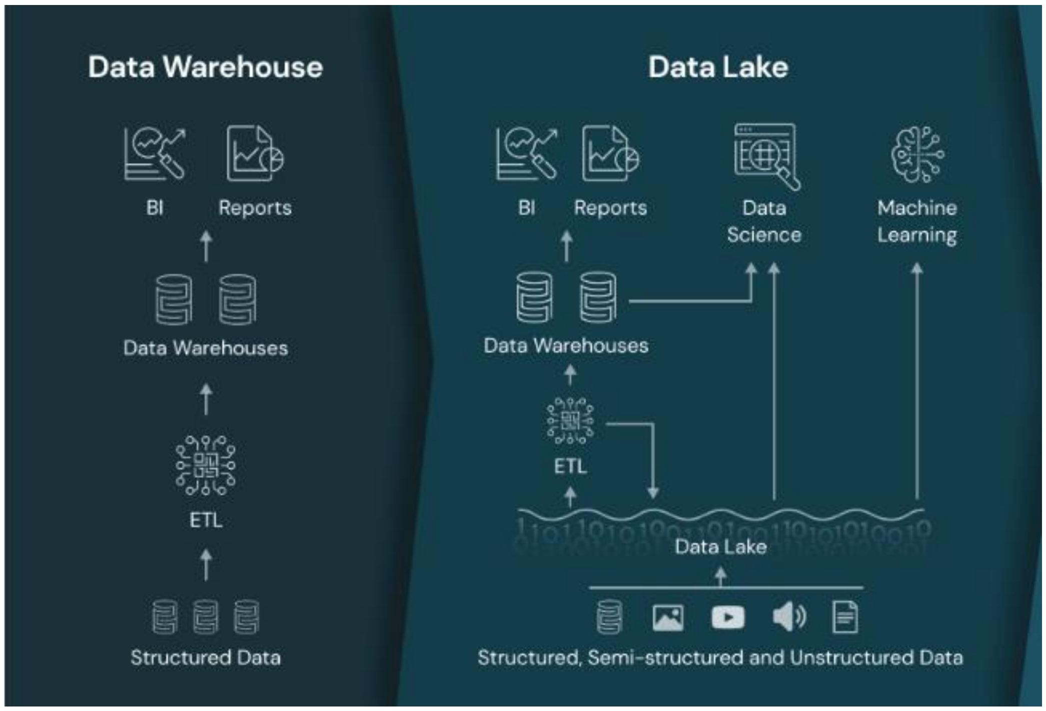


A data lake is a centralized repository that allows for the storage and analysis of vast amounts of structured, semi-structured, and unstructured data. It is designed to store data in its raw and original format, without the need for predefined schemas or transformations.

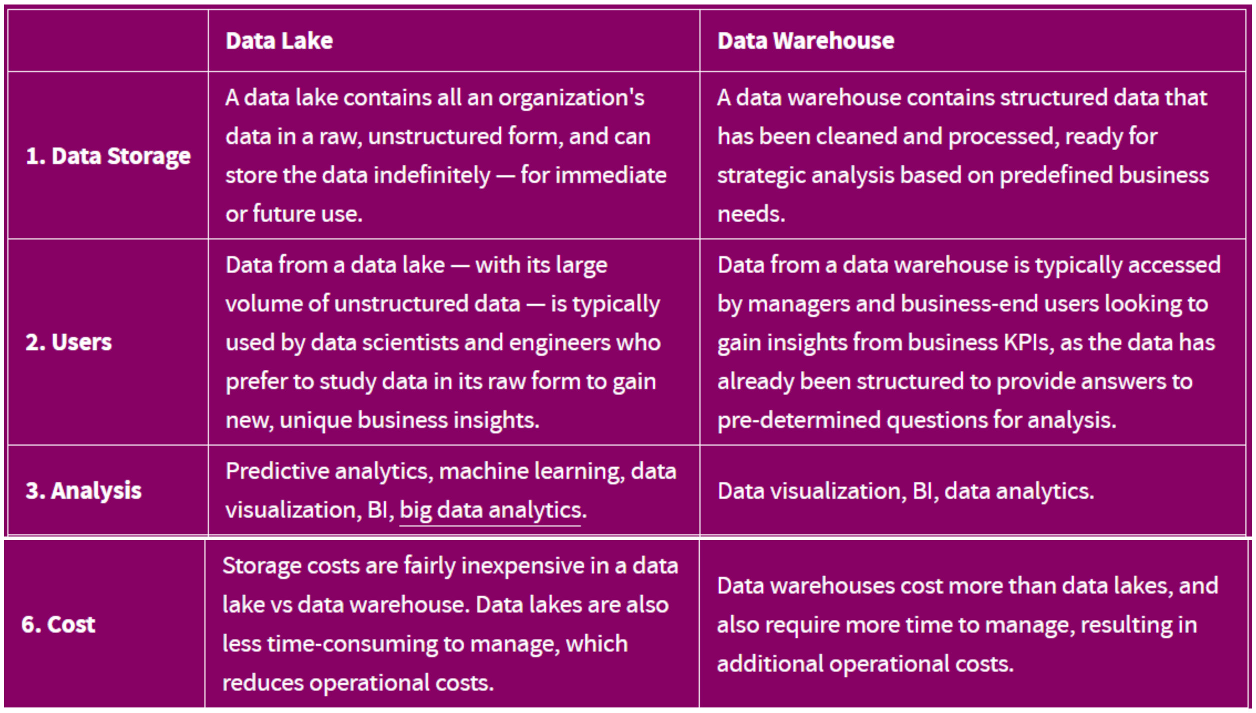
Here are some key characteristics of a data lake:

1. Storage of Raw Data: A data lake stores data from various sources in its native form, without the need for data transformation or normalization. This includes structured data (e.g., relational databases), semi-structured data (e.g., JSON, XML), and unstructured data (e.g., text documents, images, videos).
2. Scalability: Data lakes are built on scalable distributed storage systems, such as Hadoop Distributed File System (HDFS) or cloud-based object storage. This enables the storage and processing of massive amounts of data, allowing for horizontal scalability as data volumes grow.

Data Lake Architecture:



Data Lake Vs data warehouse



	Data Lake	**Data Warehouse**
1. Data Storage	A data lake contains all an organization's data in a raw, unstructured form, and can store the data indefinitely — for immediate or future use.	A data warehouse contains structured data that has been cleaned and processed, ready for strategic analysis based on predefined business needs.
2. Users	Data from a data lake — with its large volume of unstructured data — is typically used by data scientists and engineers who prefer to study data in its raw form to gain new, unique business insights.	Data from a data warehouse is typically accessed by managers and business-end users looking to gain insights from business KPIs, as the data has already been structured to provide answers to pre-determined questions for analysis.
3. Analysis	Predictive analytics, machine learning, data visualization, BI, big data analytics.	Data visualization, BI, data analytics.
6. Cost	Storage costs are fairly inexpensive in a data lake vs data warehouse. Data lakes are also less time-consuming to manage, which reduces operational costs.	Data warehouses cost more than data lakes, and also require more time to manage, resulting in additional operational costs.