
DWH

OLTP

Grow Data Skills

OLTP, or **Online Transaction Processing**, is a class of software applications capable of supporting transaction-oriented programs. Here are some key points about OLTP:

- **Transactional Operations**: OLTP systems handle a large number of short, atomic transactions. These transactions are typically CRUD operations - Create, Read, Update, Delete.
- **Concurrency**: Due to the high number of transactions, OLTP systems use multi-concurrency control techniques to prevent conflicts and ensure data consistency.
- **Data Integrity**: The systems are designed to ensure absolute data integrity through ACID (Atomicity, Consistency, Isolation, Durability) properties.
- **Real-Time Processing**: OLTP applications are designed for real-time transactional processing and quick response times.
- **High Availability**: Given the critical nature of many OLTP systems, they are designed for high availability and fault tolerance.
- **Operational vs. Analytical**: OLTP systems are focused on operational data and current transactional activity, as opposed to OLAP (Online Analytical Processing) systems, which are optimized for complex, analytical queries and historical and aggregated data reporting.
- **Performance Metrics**: The performance of OLTP systems is usually measured in transactions per second (TPS).
- **Database Design**: OLTP systems often use a relational database design with an extensive index to deliver rapid responses to SQL queries.

OLTP Systems

Grow Data Skills

- **ERP Systems**: Many ERP (Enterprise Resource Planning) systems, such as SAP ERP, Oracle ERP, Microsoft Dynamics, etc., are considered OLTP systems.
- **Banking Systems**: Online banking systems and ATM (Automated Teller Machines) software are examples of OLTP systems.
- **Airline Reservation Systems**: Software for reserving and selling tickets for airlines.
- **E-commerce Platforms**: Systems like Amazon and eBay, which handle numerous online transactions daily.
- **Telecommunication Network Systems**: They handle real-time transactions like call data records.
- **Retail POS Systems**: Point of Sale systems in retail stores.
- **Customer Relationship Management Systems**: CRM systems like Salesforce and HubSpot, where real-time data updates are crucial.
- **Online Service Applications**: Many apps, such as ride-sharing apps like Uber and Lyft, food delivery apps like DoorDash, and Airbnb, use OLTP systems for real-time transactions.

Popular Databases for OLTP

Grow Data Skills

- **Oracle Database**: A popular relational database management system from Oracle Corporation.
- **MySQL**: An open-source relational database management system owned by Oracle Corporation.
- **Microsoft SQL Server**: A relational database management system developed by Microsoft.
- **PostgreSQL**: A powerful, open-source object-relational database system.
- **IBM DB2**: A family of database server products developed by IBM.
- **MariaDB**: An open-source relational database management system, forked from MySQL.
- **SAP HANA**: An in-memory, column-oriented, relational database management system developed by SAP.
- **Amazon Aurora**: A relational database service from Amazon Web Services (AWS), compatible with MySQL and PostgreSQL.
- **Google Cloud Spanner**: A scalable, enterprise-grade, globally-distributed, and strongly consistent database service built for the cloud specifically to combine the benefits of relational database structure with non-relational horizontal scale.
- **CockroachDB**: An open-source distributed SQL database designed for global online transaction processing (OLTP).

OLAP

Grow Data Skills

OLAP, or **Online Analytical Processing**, is a category of software tools that provide analysis of data for business decisions. It is characterized by relatively low volume of transactions but complex queries involving aggregations, which need to be executed relatively quickly.

Here are the key characteristics of OLAP systems:

- **Multi-Dimensional Analysis**: OLAP tools allow users to analyze data along multiple dimensions, which is generally more intuitive for users. For example, a business person might want to analyze sales by product, by region, and by time period.
- **Speedy Query Performance**: Despite the size and complexity of the data sets, OLAP tools provide rapid results to queries due to their use of multidimensional data cubes and precalculation.
- **Aggregation and Computation**: OLAP systems are optimized to perform complex calculations and data aggregations, like sums, averages, ratios, ranks, etc., on the fly.
- **Support for Complex Queries**: OLAP tools support complex queries and enable users to perform 'what-if' type analysis.
- **Data Discovery**: They support data discovery by providing flexible, ad-hoc querying and multi-dimensional analysis.
- **Read-Optimized**: Unlike OLTP, which is write-optimized, OLAP is optimized for a high volume of read operations with fewer write operations.

OLAP Systems

Grow Data Skills

There are several databases and systems that are designed to work as OLAP systems, providing multi-dimensional analysis of data. Here are some of them:

- **Microsoft Analysis Services:** Also known as SSAS (SQL Server Analysis Services), it provides data mining and multi-dimensional analysis.
- **SAP BW (Business Warehouse):** SAP's data warehousing solution that includes OLAP capabilities.
- **Amazon Redshift:** A fully managed, petabyte-scale data warehouse service in the cloud from Amazon Web Services.
- **Google BigQuery:** A fully managed, serverless data warehouse that enables super-fast SQL queries and interactive analysis of massive datasets.
- **Snowflake:** A cloud-based data warehousing platform that supports multi-dimensional analysis of large volumes of data.

Difference between OLTP vs OLAP

Grow Data Skills

	OLTP	OLAP
Main Function	Manage day-to-day transaction of an organization.	Supports complex analysis and decision making.
Database Design	Normalized. Optimized for INSERTs and UPDATEs.	Denormalized. Optimized for data analysis and reporting.
Data	Detailed, operational data.	Summarized, consolidated, and historical data.
Transactions	Short and fast updates and queries.	Long, complex queries involving aggregations.
Number of Records Accessed	Accesses individual records (e.g., single row in a table).	Accesses large volumes of data, often entire tables or multiple tables.
Performance Metrics	Measured in transactions per second (TPS).	Measured in query response time.
Examples	Online banking, online airline ticket booking, etc.	Sales trend analysis, financial reporting, business intelligence, etc.
Users	Front line workers such as clerks, IT professionals, etc.	Managers, Business Analysts, Decision Makers.
Database Size	Smaller due to current operational data.	Larger due to storing historical data.
Nature of Queries	Simple standard queries.	Complex ad-hoc queries.
Consistency Requirement	High need for concurrency control and transaction consistency.	Lower consistency requirements, data frequently loaded in batches.

Normalized Data

Grow Data Skills

Data Normalization is a process in database design that organizes data to minimize redundancy and improve data integrity. It involves structuring data in accordance with a series of so-called normal forms, each with an increasing level of strictness.

The primary advantages of normalization are:

- **Minimize Redundancy**: By ensuring that each piece of data is stored in only one place, normalization reduces the redundancy of data within the database.
- **Data Consistency**: Normalization improves data consistency as each data item is stored in only one place. Any update needs to be performed only in one place, reducing the chances of inconsistent data.
- **Efficient Use of Storage**: Normalized databases are more compact and often require less storage space, especially for large databases.
- **Simplified Data Management**: Normalization simplifies the management and updating of data as it ensures that a piece of information exists in one place only.

Normalized Data Systems, such as OLTP (Online Transaction Processing) systems, use normalized databases. These systems require high data integrity and support a large number of short online transactions (INSERT, UPDATE, DELETE). Examples include databases for managing sales transactions, customer relationship management (CRM) systems, airline reservation systems, and banking systems. These systems rely on data normalization to ensure data consistency and efficiency in processing a high volume of transactions.

Normalized Database

Employee

employeeID	employeeName	managerID	sectorID
1	David D.	1	4
2	Eugene E.	1	3
3	George G.	2	2
4	Henry H.	2	1
5	Ingrid I.	2	4
6	James J.	3	1
7	Katy K.	3	4

Sector

sectorID	sectorName
1	Administration
2	Security
3	IT
4	Finance

Manager

managerID	managerName	area
1	Adam A.	East
2	Betty B.	West
3	Carl C.	North

Denormalized Data

Grow Data Skills

Data Denormalization is the process of combining data from several normalized tables into one, for the purpose of improving read performance, simplifying queries, or preparing the data for specific analytical requirements. This is the opposite of normalization, where data is separated into multiple tables to minimize redundancy and improve data integrity.

Denormalization may lead to data redundancy and anomalies (like insert, update, or delete anomalies), but it reduces the need for complex joins and can therefore enhance the performance of read-heavy applications.

Denormalized Data Systems, such as **OLAP (Online Analytical Processing)** systems or data warehousing systems, use denormalized databases. These systems are designed for reporting and data analysis, and they support complex queries against large amounts of data. The data in these systems is typically read-only or infrequently updated, and the emphasis is on retrieval speed rather than transaction speed.

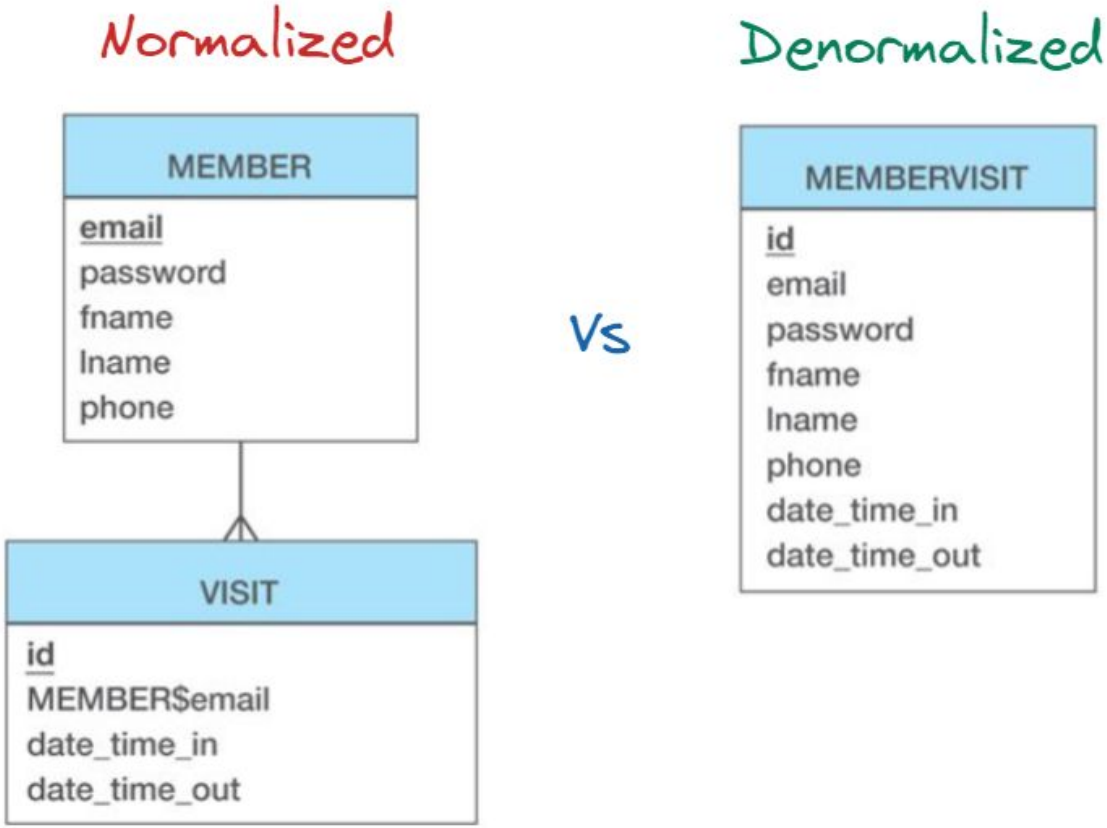
Here are some key characteristics of denormalized data systems:

- **Data Redundancy**: Since data is combined into fewer tables, some data can be repeated or redundant.
- **Read Optimization**: Denormalized databases can retrieve data faster because they often need fewer joins to answer a query.
- **Complexity**: By reducing the number of tables, the database structure becomes simpler, and complex queries can be more easily written.
- **Data Integrity**: In denormalized databases, maintaining data integrity can be more challenging due to potential redundancy.
- **Usage**: Denormalized data systems are commonly used for data analysis, data mining, data warehousing, and any system where reading data is more important than updating or inserting data.

It's important to note that denormalization doesn't mean that normalization principles are entirely abandoned. Rather, it's a strategic optimization for specific read-heavy use cases.

Denormalized Data

Grow Data Skills



Normalized vs Denormalized Data

Grow Data Skills

	Normalized Data	Denormalized Data
Structure	Data is spread across multiple tables to eliminate redundancy and dependency.	Data is combined into fewer tables to speed up data retrieval.
Redundancy	Redundancy is minimized. Each data item is stored in only one place.	Redundancy may be introduced to improve read performance.
Performance	Writing data (inserts, updates, deletes) is usually faster, while reading can be slower due to the need for multiple joins.	Reading data is typically faster due to fewer joins, while writing can be slower due to redundancy.
Data Integrity	Higher, as redundancy is eliminated.	Lower, as redundancy can lead to inconsistencies.
Use Case	Optimized for transactional systems like OLTP where write operations are common.	Optimized for analytical and reporting systems like OLAP where read operations are common.

Which is better depends on your use case:

- **Choose Normalized Data when**: You are dealing with transactional systems (like OLTP), where write operations (insert, update, delete) are common and the data integrity is crucial. This includes applications like online retail websites, banking systems, CRM, etc.

- **Choose Denormalized Data when**: You are dealing with analytical systems (like OLAP), where read operations and speed of data retrieval are more important than write operations. This is common in reporting, data mining, and decision support systems.

It's important to remember that this is not an either/or situation. Often, in a single system, a part of the database might be normalized (to support transactional operations) and another part might be denormalized (to support analytical operations). It's all about choosing the right tool for the right job based on your specific needs.

What is DataLake?

Grow Data Skills

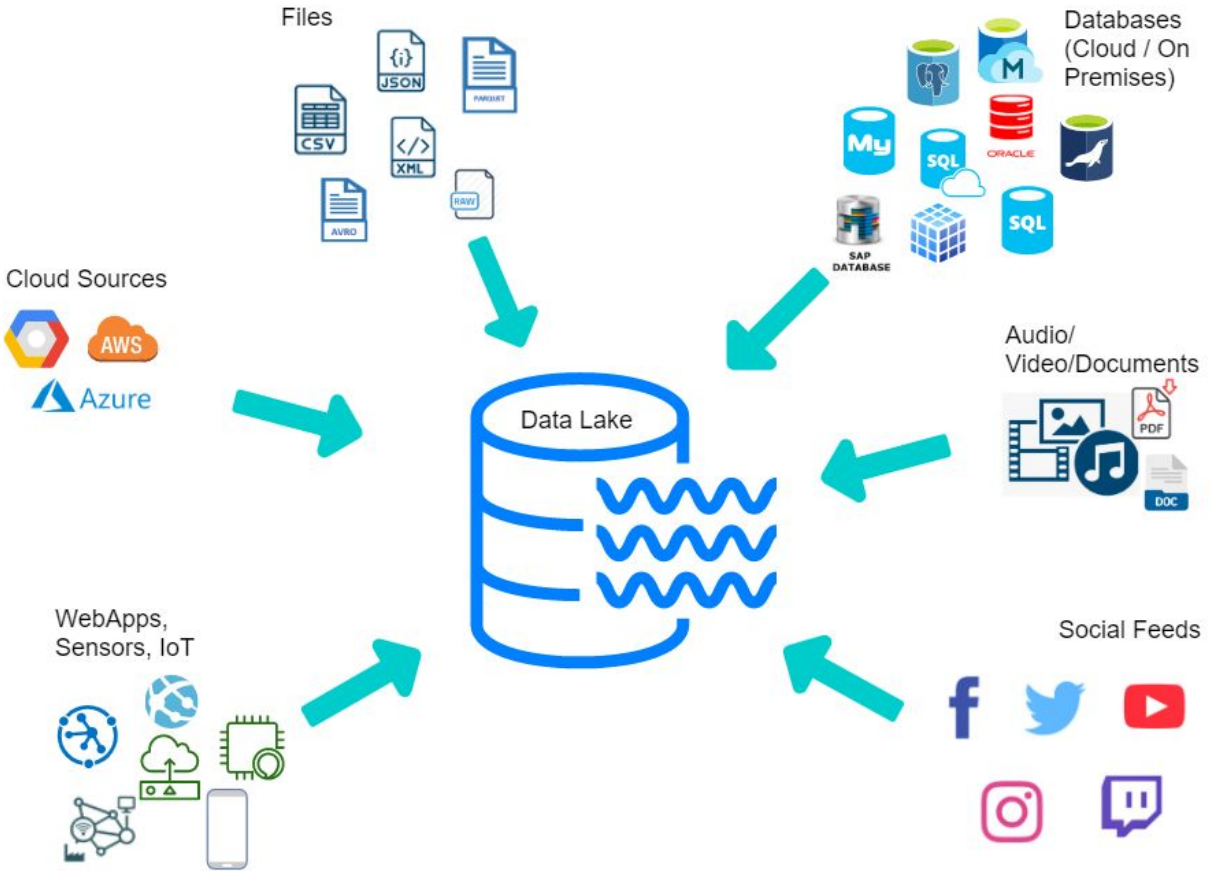
Data Lake is a storage repository, usually on a large scale, that holds a vast amount of raw data in its original, native format until it's needed. Unlike a traditional data warehouse, which stores data in a structured and often processed format, a data lake retains all data—structured, semi-structured, and unstructured—and does not require the schema to be defined before storing the data.

Here are some reasons why we need data lakes:

- **Data Variety**: Can store any type of data (structured, semi-structured, unstructured), accommodating the wide variety of data formats in big data.
- **Scalability**: Can handle massive volumes of data and quickly scale as data grows.
- **Cost-Effective**: Utilizes inexpensive, commodity hardware for storing large amounts of data.
- **Data Exploration**: Allows for exploratory data analysis, which is essential for data discovery and advanced analytics.
- **Data Integration**: Acts as a central hub for all organizational data, creating a unified data view.
- **Real-time Processing**: Supports both batch and real-time data processing, suitable for real-time analytics.
- **Schema on Read**: Doesn't require data modeling before storing, offering flexibility in data usage.

What is DataLake?

Grow Data Skills



What is DataWarehouse?

Grow Data Skills

A Data Warehouse is a large, centralized repository of data that combines information from different sources to support data analysis and reporting. It is specifically designed for Online Analytical Processing (OLAP), allowing for complex queries and analysis.

Here's why we need data warehouses:

- **Unified Information**: Data warehouses integrate data from multiple disparate sources, creating a central repository for all organizational data. This enables a unified view of the entire business, enhancing decision-making.
- **Improved Business Intelligence**: By providing data that has been cleaned, consolidated, and summarized, a data warehouse makes it easier to generate reports and dashboards, carry out data mining, and perform analytics, thus supporting better business decision-making.
- **Enhanced Data Quality and Consistency**: Data warehouses enforce data consistency by making sure that all data from different sources adhere to the same format. This ensures that the data is reliable and accurate, enhancing the quality of business insights derived from it.
- **Historical Data for Trend Analysis**: Data warehouses typically store large amounts of historical data, which is essential for trend analysis, forecasting, and decision-making.
- **High Performance**: Data warehouses are optimized for read operations, enabling quick retrieval of data. They typically use denormalized data, which reduces the need for complex joins and improves the speed of queries.
- **Separation of Operational and Analytical Processes:** Keeping a separate data warehouse allows organizations to offload their analytics workload from operational databases, preventing the performance of transactional systems from being affected by analytical queries.
- I**mproved Data Security**: Data warehouses provide a more secure environment for data as they include security features to protect data from unauthorized access.

It's important to note that implementing a data warehouse is not a trivial task. It involves data cleaning, data integration, and data transformation tasks that can be complex and time-consuming. Therefore, the decision to create a data warehouse should take into consideration the specific needs of the organization, the availability of resources, and the potential return on investment.

We can only store Structured Data in Data Warehouse?

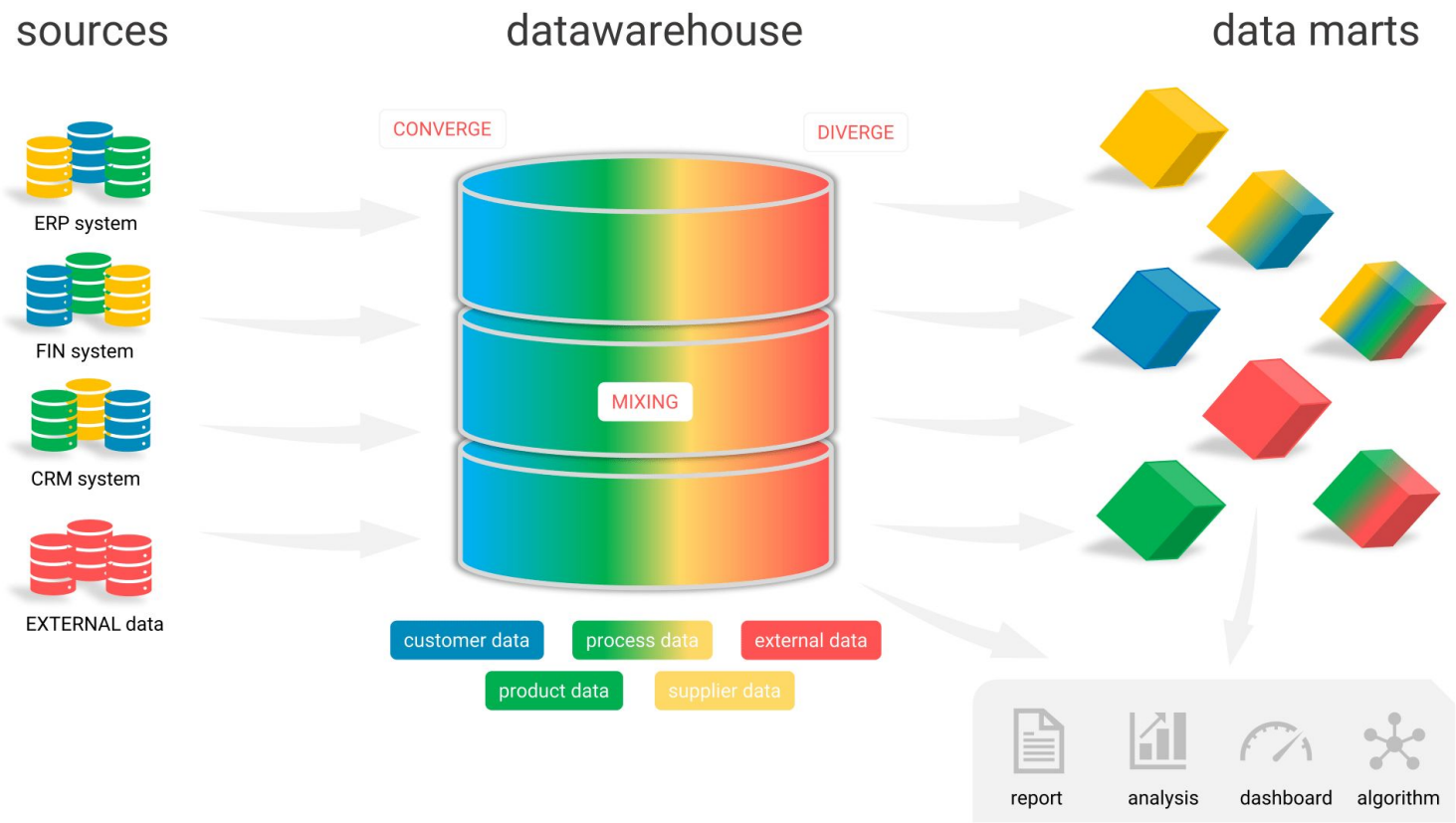
Grow Data Skills

Traditionally, data warehouses were designed to handle structured data - that is, data that fits neatly into a table with rows and columns, like you would find in a relational database or an Excel spreadsheet. This structured data could be analyzed using SQL or similar querying languages.

However, with the advent of Big Data, the nature of data has evolved. Today's organizations also need to handle semi-structured data (like JSON or XML files) and unstructured data (like text documents, images, videos, etc.) to extract insights. Consequently, the concept of data warehousing has also evolved.

Modern data warehouses have extended their capabilities and can now accommodate, process, and analyze semi-structured and even unstructured data. For example, solutions like Google's BigQuery, Amazon's Redshift, and Snowflake allow users to query non-relational data types including nested and repeated data.

However, while it's possible to store and query unstructured and semi-structured data in a modern data warehouse, often it's not the most efficient or cost-effective way to handle such data. For many use cases, it may be more appropriate to use other types of data storage and processing systems, like data lakes or NoSQL databases, which are specifically designed to handle these types of data. These systems can then work in conjunction with a data warehouse as part of a broader data architecture.


sources
datawarehouse
data marts
ERP system
FIN system
CRM system
EXTERNAL data
CONVERGE
DIVERGE
MIXING
customer data
process data
external data
product data
supplier data
report
analysis
dashboard
algorithm

What is Data Mart?

Grow Data Skills

A Data Mart is a subset of a data warehouse that is focused on a specific area of business, such as sales, marketing, finance, or HR. Essentially, a data mart is a condensed version of a data warehouse that is tailored to fit the needs of a specific business unit or team.

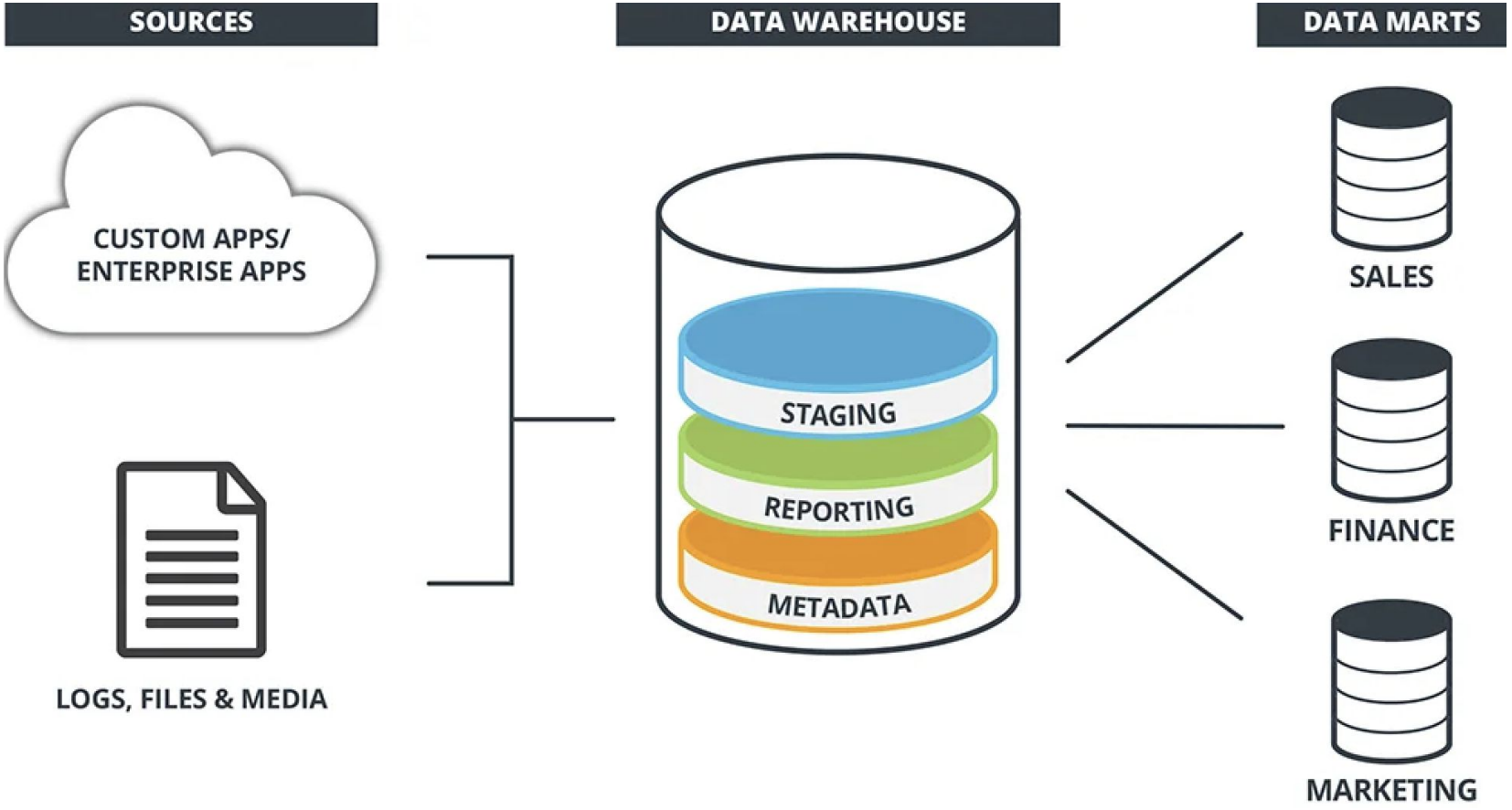
Here are the reasons why we need data marts:

- **Focused Business Insights:** Since data marts focus on a specific area of business, they can provide more detailed and relevant insights for a specific department or team.
- **Improved Performance**: With less data to process, queries run faster in data marts, improving response times for users.
- **Ease of Use:** Data marts are simpler and more intuitive for end users because they only contain data relevant to a specific business area. Users don't need to sift through unrelated data, which makes their analysis faster and more efficient.
- **Autonomy**: By having their own data mart, a business unit can maintain control over their data, making decisions on data handling, and access based on their unique needs and priorities.
- **Less Risk:** With data marts, you can start small and grow over time. This approach reduces the risk and cost compared to implementing a full data warehouse at once.
- **Increased Security:** It's easier to secure data because data marts only hold a limited set of data. Therefore, even if a breach occurs, the amount of exposed data would be less compared to a full data warehouse.

Despite these advantages, it's essential to ensure that data marts are well-coordinated within an organization to avoid issues like data duplication or data inconsistencies. Sometimes, an organization may choose a hybrid approach, having a central data warehouse for the entire organization, and then creating data marts for specific departments or functions.

What is Data Mart?

Grow Data Skills



Difference between Data Warehouse Design & Data Modelling?

Grow Data Skills

Data Warehouse Design and Data Modelling are interrelated concepts in the field of data management, but they focus on different aspects. Here's a brief comparison:

Data Warehouse Design refers to the process of creating a blueprint for the data warehouse system. It includes the decision-making and planning for:

- **Data sources**: Identifying the sources from where data will be pulled.
- **Data transformation:** Defining how data will be cleaned, transformed, and integrated.
- **Data storage:** Choosing the structure in which data will be stored, such as deciding between a normalized or denormalized schema, or choosing between different architectures like a Star Schema, Snowflake Schema, etc.
- **Data retrieval:** Planning for how data will be queried and retrieved, such as creating indexes, views, or OLAP cubes.
- **Security and backup:** Defining how the data will be secured, how privacy will be maintained, and how data will be backed up.

Data Modelling, on the other hand, is a specific part of the data warehouse design process. It involves creating a conceptual model of how data should be structured in the database.

Data Modelling Concepts - Fact Tables

Grow Data Skills

A **Fact Table** is a central component of a **star schema** or a **snowflake schema** in a dimensional modelling. It is the main table in a dimensional model, which holds the **quantitative** or **measurable data** that a business wishes to analyze.

Fact tables typically contain two types of columns:

- **Fact**: These are the measurable, quantitative data points that are relevant to the business. They are usually numerical values that can be aggregated (like summed, averaged, etc.). Examples of facts could be 'Sales Amount', 'Quantity Sold', 'Profit', etc.

- **Foreign Keys:** These are the keys that connect the fact table to its associated dimension tables. These keys enable you to categorize your facts and look at them from different perspectives.

For instance, in a retail business scenario, a fact table might have a structure like this:

Date Key	Product Key	Store Key	Sales	Quantity Sold
20230725	123	001	5000	100

In this case, 'Sales' and 'Quantity Sold' are the facts. 'Date Key', 'Product Key', and 'Store Key' are the foreign keys linking the fact table to the dimension tables for 'Date', 'Product', and 'Store', respectively. This design allows you to analyze your sales and quantities from different dimensions like time, product type, and location.

Data Modelling Concepts - Dimension Tables

Grow Data Skills

The main purpose of a dimension table is to provide context and descriptive attributes for facts stored in the Fact Table in the same dimensional data model. This allows users to understand and interpret the quantitative values in the fact table.

Characteristics of dimension tables include:

- **Descriptive**: Dimension tables store descriptive or textual data, often referred to as "attribute data." For example, the Product dimension table might include details like product name, product type, product category, etc.
- **Granularity**: The granularity of a dimension table is at the level of the individual instance. For instance, each row in a Product dimension table would represent a single product.
- **Primary Key**: Each dimension table has its own primary key, which is used to connect the dimension table with the fact table(s).
- **Hierarchical**: Dimensions often include a hierarchy of categories. For instance, a "Time" dimension might include fields for Year, Quarter, Month, and Day.
- **Stability**: While fact table data changes frequently, dimension table data is relatively stable. However, dimension tables can change in response to business changes, such as adding a new product or a new sales region.

Data Modelling Concepts - Dimension Tables

Grow Data Skills

Store_ID	Store_Name	Store_Type	Location
001	Store A	Supermarket	New York
002	Store B	Convenience Store	Boston
003	Store C	Supermarket	Chicago

In this example, 'Store_ID' would be the primary key that connects this dimension table to the fact table, and the other columns ('Store_Name', 'Store_Type', 'Location') provide descriptive details about each store.

Dimension tables help in performing meaningful analysis of the fact data. They allow the business users to look at the data from various perspectives and help in slicing and dicing the fact data.

Data Modelling Concepts - Star Schema

Grow Data Skills

The Star Schema is a simple yet powerful database architecture used in data warehousing and business intelligence reporting. The star schema gets its name from the physical model's resemblance to a star shape with a fact table in the middle surrounded by dimension tables.

Here is a thorough explanation of the Star Schema:

- **Fact Table:** At the center of the star schema is the fact table. This table contains the quantitative or measurable data (the "facts") that the business wants to analyze. For example, in a retail business, the fact table might store transaction data like units sold and total sales. Fact tables often contain a large number of rows, reflecting the detailed level of data they store.

- **Dimension Tables**: Surrounding the fact table are dimension tables. These tables contain descriptive data that provide context to the facts. For example, in the retail business scenario, the dimension tables might include information on products, customers, stores, and dates. Each row in a dimension table represents a unique instance of that dimension, like a specific product or a specific store.

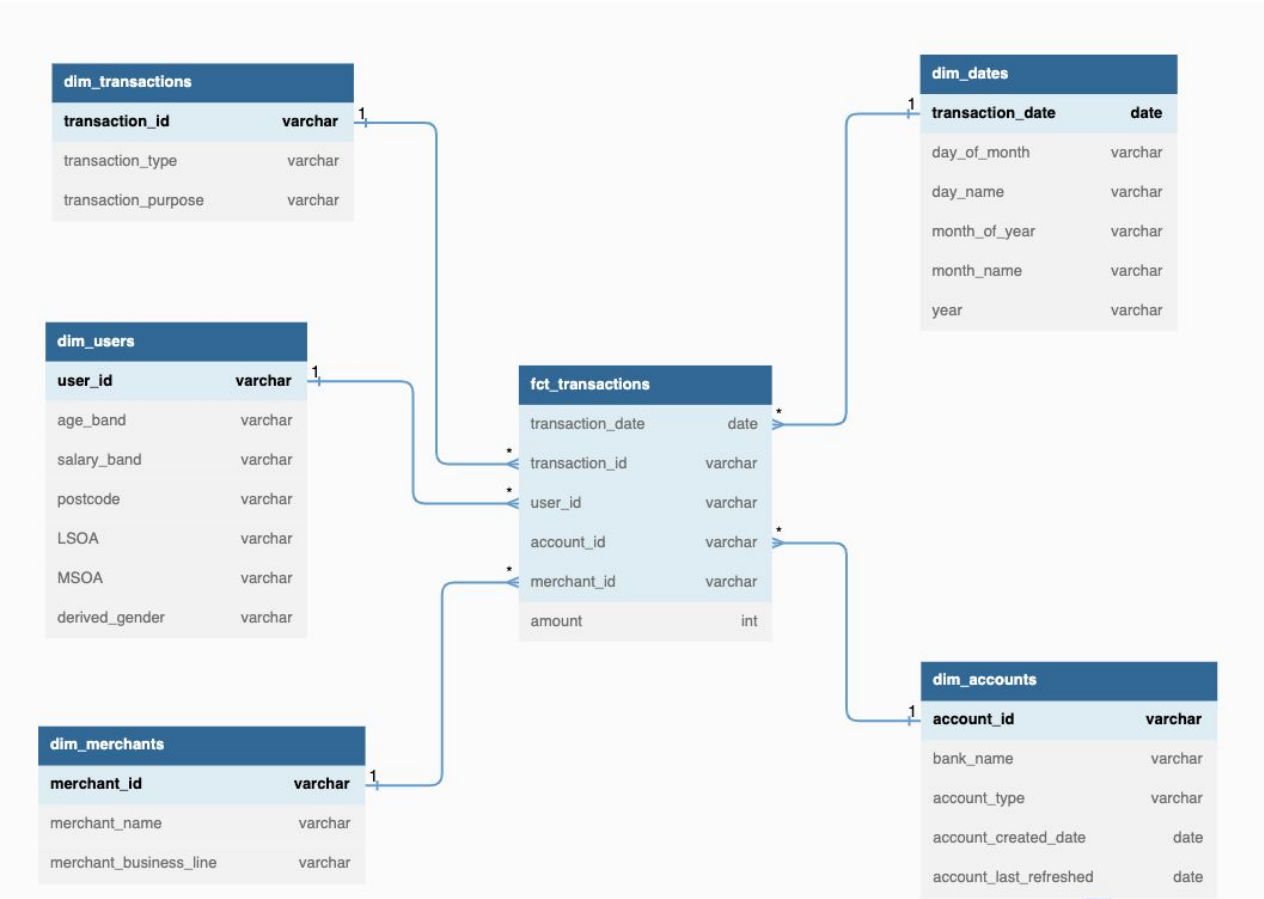
- **Schema Layout:** In a star schema, each dimension table is directly connected to the fact table via a foreign key relationship. The fact table includes a foreign key for each dimension table that it's linked to. These foreign keys enable the joining of the fact table with dimension tables.

- **Simplicity**: The star schema is denormalized, which means it does not strictly enforce certain data integrity rules that are required in a normalized database design. This denormalization simplifies the database design and makes it easier to write and run queries.

- **Performance**: The star schema is designed for efficient data retrieval and is the standard schema for a reason. The simple, predictable queries, reduced number of tables, and minimized joins make it ideal for handling complex business queries and aggregations.

Data Modelling Concepts - Star Schema

Grow Data Skills



Data Modelling Concepts - Snowflake Schema

Grow Data Skills

The Snowflake Schema is a type of database schema that is used in data warehousing. It's a variant of the star schema, but with additional levels of complexity due to further normalization of the dimension tables. It's called a "snowflake" schema because its diagram resembles a snowflake with spokelike arms branching out from a center.

Here's a detailed explanation of the Snowflake Schema:

- **Fact Table:** Similar to the star schema, the snowflake schema also consists of a central fact table which contains the business data (facts) that are being analyzed. This could include things like sales amounts, quantities, and so forth.

- **Normalized Dimension Tables**: In the snowflake schema, the dimension tables are normalized. This means that the data in the dimension tables is split or segregated into additional tables. For example, instead of having a single "Product" table that includes all product-related information, you might have a main "Product" table linked to separate tables for "Product Category" and "Product Manufacturer". This normalization reduces data redundancy.

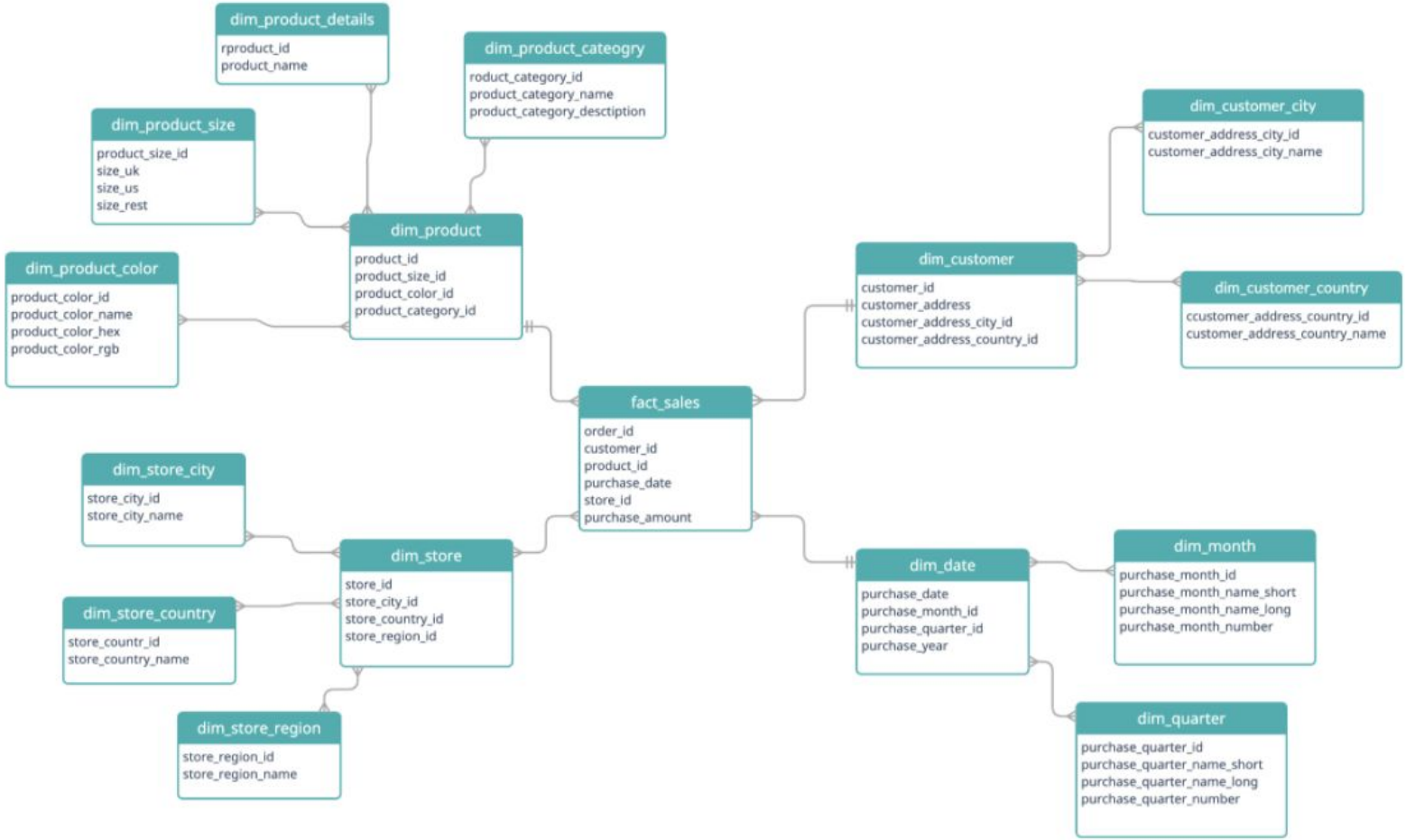
- **Schema Layout:** In a snowflake schema, the fact table is at the center with multiple branching dimension tables, which can have other tables branching off of them. This multi-level relationship between the fact table and dimension tables gives the schema its snowflake-like appearance.

- **Query Complexity**: Due to the additional level of normalization, queries in snowflake schemas can become more complex as they may involve more tables and joins. This might lead to longer query times, but it can also lead to more efficient storage.

- **Storage Efficiency**: One of the main advantages of the snowflake schema is the reduction in storage required due to the normalization of dimension tables. This can also lead to improved data integrity as it helps to eliminate redundancy and inconsistencies.

Data Modelling Concepts - Snowflake Schema

Grow Data Skills



Data Modelling Concepts - Galaxy Schema

Grow Data Skills

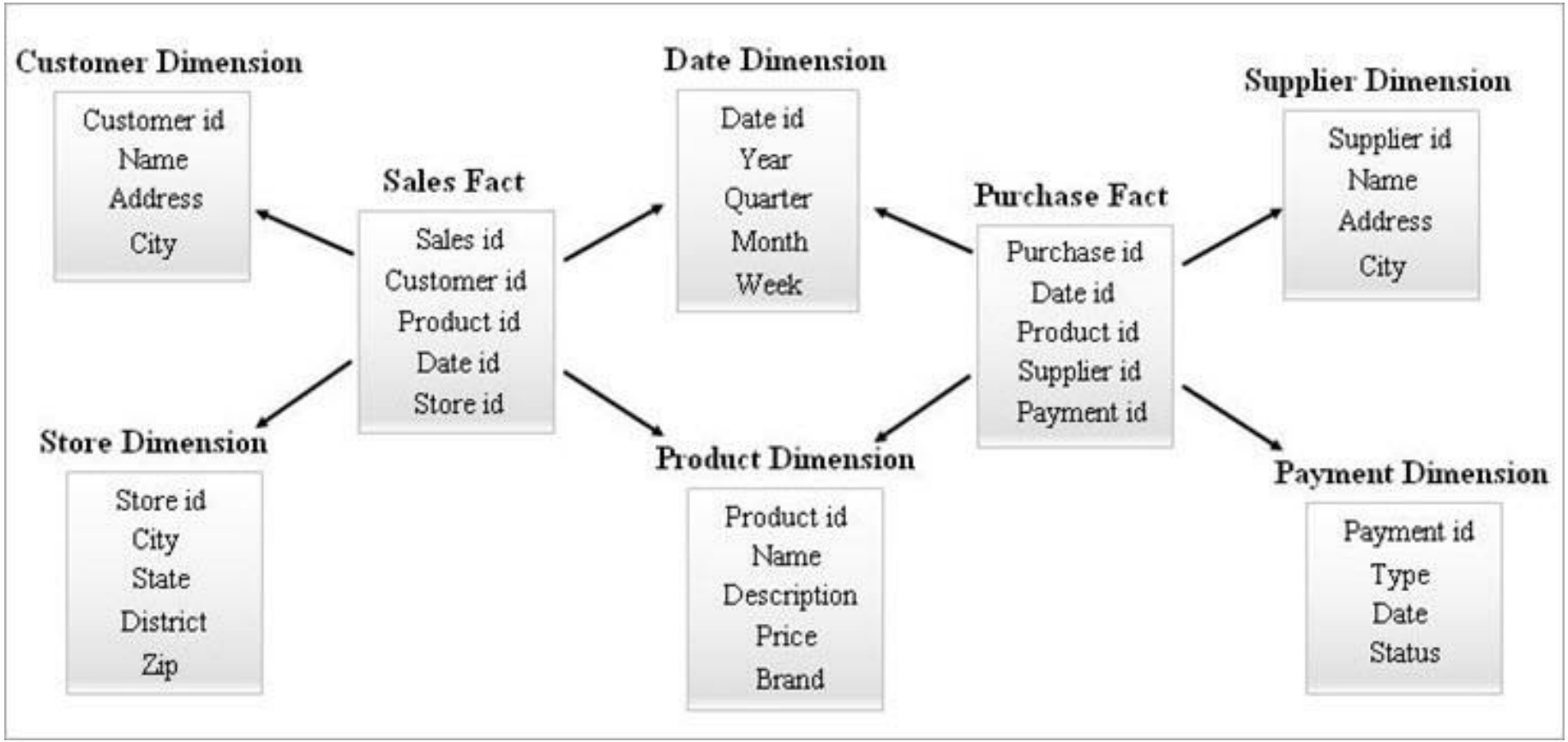
A **Galaxy Schema,** also known as a Fact Constellation Schema, is a complex type of schema used in data warehouse environments. It extends the concepts of the Star and Snowflake schemas by allowing multiple fact tables to share dimension tables.

Here is a thorough explanation of the Galaxy Schema:

- **Multiple Fact Tables**: Unlike the star and snowflake schemas, which have a single fact table, a galaxy schema can have multiple fact tables. Each fact table in a galaxy schema represents a different business process or event.
- **Shared Dimension Tables**: The dimension tables in a galaxy schema are often shared among fact tables. This makes it possible to analyze facts from different fact tables across shared dimensions.
- **Schema Layout**: The galaxy schema looks like multiple star or snowflake schemas combined, where the fact tables form the center of each star or snowflake, and the dimension tables branch out from there. The shared dimensions form the intersecting points between these multiple stars or snowflakes.
- **Complexity**: Galaxy schemas are typically more complex than star or snowflake schemas. They involve more tables, more relationships, and more complex queries. However, they also offer more analytical capabilities due to the ability to compare and analyze multiple facts across shared dimensions.
- **Flexibility**: One of the main advantages of the galaxy schema is its flexibility. It allows for complex analyses and queries across multiple business processes or events. This can be particularly useful in large organizations with complex data needs.

Data Modelling Concepts - Galaxy Schema

Grow Data Skills



Data Modelling Concepts - SCD

Grow Data Skills

SCD stands for **Slowly Changing Dimensions,** which are a common concept in Data Warehousing, Business Intelligence, and data modeling. These dimensions are the aspects of the business that can change over time, but not at a high frequency, hence "slowly changing."

- **SCD Type 1 (Overwrite):** In this type, when changes occur in attribute values, the old values are overwritten with new ones. No history is kept in this scenario, only the current state is stored. This approach is typically used for minor or correctional changes where tracking the history is not important.

  Example: Let's take an example of a customer dimension where the customer's address is an attribute. If a customer changes their address, in a Type 1 SCD, the new address would simply replace the old address in the customer dimension table.

- **SCD Type 2 (Add a new row):** This type is used when it is necessary to maintain a full history of data changes. When changes occur, instead of updating the existing records, a new row is added with the new values, and the original record is marked as inactive or retained with a flag indicating that it is the old version.

  Example: In the same customer dimension example, if the customer changes their address and we are using a Type 2 SCD, a new row would be added for the customer with the new address, and the old address row would be marked as inactive or have a flag indicating it's an old version. This way, we have a complete history of all addresses the customer has had over time.

- **SCD Type 3 (Add a new column):** In this type, when changes occur, a new column is added to the table to track the changes. This is generally used when we are only interested in maintaining the current value and the immediate previous value.

  Example: In the customer dimension example, if the customer changes their address and we are using a Type 3 SCD, a new column would be added (e.g., "Previous_Address") to store the old address, and the "Address" column would be updated with the new address. This method allows us to see the current and previous address but doesn't keep a full history of all addresses.

Data Modelling Concepts - SCD

Grow Data Skills

- **SCD Type 4 (Using History Table):** This method involves the use of a separate history table to track the changes. When a change happens, the current table (dimension table) gets updated, and the old record gets pushed into the history table. This method keeps the dimension table lightweight but provides full historical context in the separate table.

  Example: If a customer changes their address in a Type 4 SCD setup, the customer's current address in the main customer table would be updated with the new address, and the old address would be stored in a separate customer address history table.

- **SCD Type 6 (Combination of Type 1, 2, and 3):** Also known as a hybrid method, this combines elements of Type 1, Type 2, and Type 3 SCDs. This method usually holds one or more current attributes (Type 1), one or more historical attributes (Type 2), and may include a "previous value" column for specific attributes (Type 3).

  Example: If a customer changes their address in a Type 6 SCD setup, a new row would be added with the new address (like Type 2), and the old row may be kept as it is or updated in a Type 1 style for certain attributes. Additionally, there might be a column in the new row capturing the previous address (like Type 3).

Please note that Type 4 and Type 6 are less commonly used due to their complexity, but they do provide additional options for handling changes over time in a data warehouse environment. The decision on which type to use depends on the specific needs of your business and the resources available for managing the data warehouse.