

What’s the difference between OLAP and OLTP?


Online analytical processing (OLAP) and online transaction processing (OLTP) are data processing systems that help you store and analyze business data. You can collect and store data from multiple sources—such as websites, applications, smart meters, and internal systems.

- OLAP combines and groups the data so you can analyze it from different points of view. Conversely,
- OLTP stores and updates transactional data reliably and efficiently in high volumes. OLTP databases can be one among several data sources for an OLAP system.

Summary of differences: OLAP vs. OLTP

Criteria	OLAP	OLTP
Purpose	OLAP helps you analyze large volumes of data to support decision-making.	OLTP helps you manage and process real-time transactions.
Data source	OLAP uses historical and aggregated data from multiple sources.	OLTP uses real-time and transactional data from a single source.
Data structure	OLAP uses multidimensional (cubes) or relational databases.	OLTP uses relational databases.
Data model	OLAP uses star schema, snowflake schema , or other analytical models.	OLTP uses normalized or denormalized models.
Volume of data	OLAP has large storage requirements. Think terabytes (TB) and petabytes (PB).	OLTP has comparatively smaller storage requirements. Think gigabytes (GB).
Response time	OLAP has longer response times, typically in seconds or minutes.	OLTP has shorter response times, typically in milliseconds
Example applications	OLAP is good for analyzing trends, predicting customer behavior, and identifying profitability.	OLTP is good for processing payments, customer data management, and order processing.

Confused about different types of data models , please check below table you can understand .

Feature	Star Schema	Snowflake Schema	Normalized Model	Denormalized Mo 
Data Type	OLAP	OLAP	OLTP	OLAP
Dimension Tables	Denormalized	Normalized	Normalized	Denormalized
Query Speed	Fast	Medium	Slow	Fast
Storage Use	High	Low	Low	High
Design Complexity	Low	Medium	High	Low

Star Model:

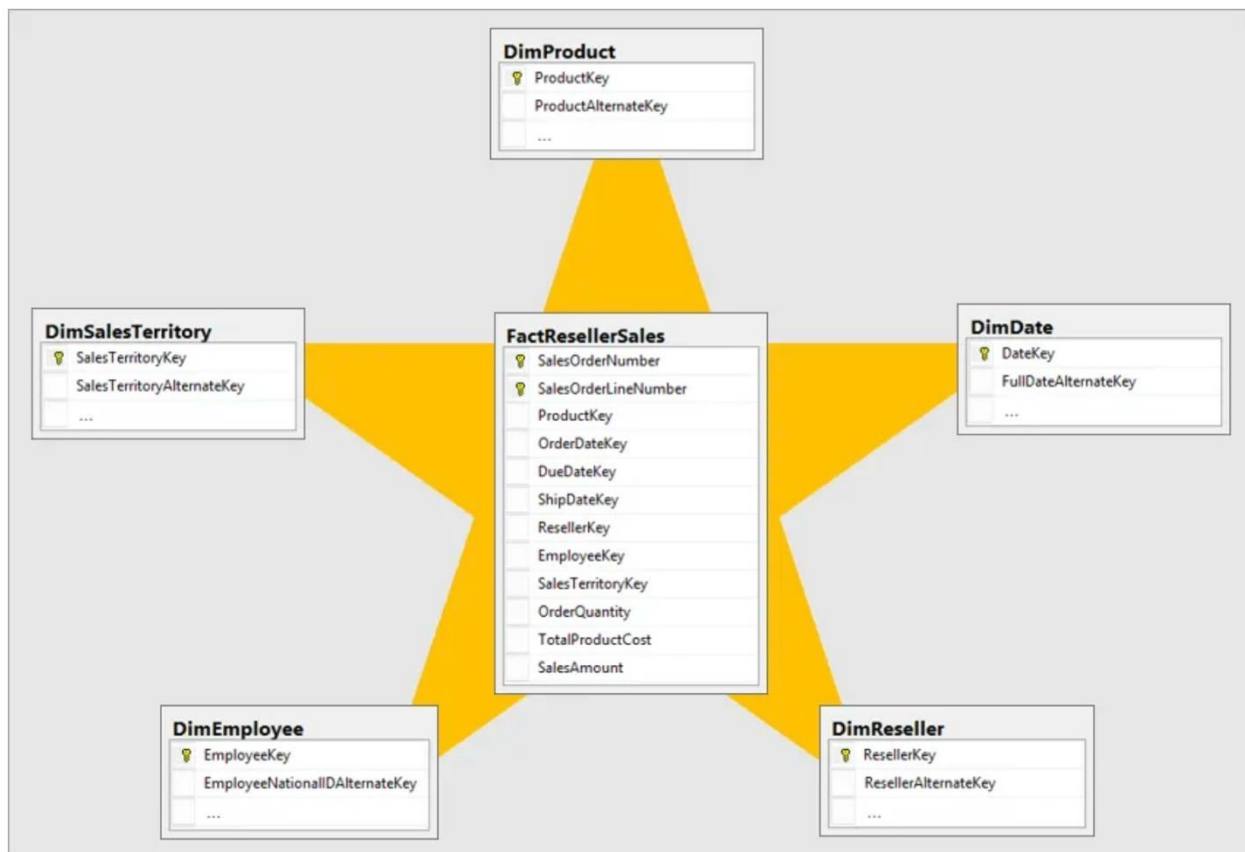
A **star schema** is a simple and popular data modeling technique used in **Data Warehousing** and **BI systems**. It consists of:

- A **central Fact Table** (contains measurable data).
- Connected to **Dimension Tables** (descriptive attributes).

The structure **resembles a star**, where the fact table is at the center and dimension tables radiate outward.

Structure

- **Fact Table:** Stores **quantitative data** (e.g., sales, revenue, quantity).
- **Dimension Tables:** Store **descriptive details** (e.g., product name, customer location, date).



Snowflake Model:

A **Snowflake Schema** is a logical arrangement of tables in a **multidimensional database** that resembles a snowflake shape. It is a **normalized version** of the **Star Schema**, where:

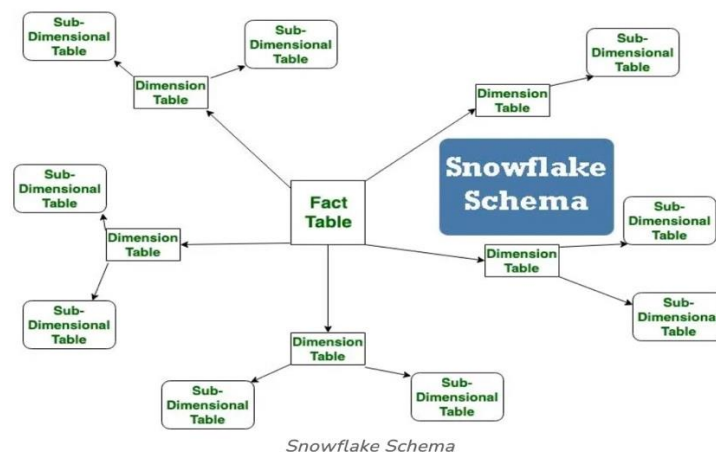
- **Dimension tables are split into sub-dimensions** (i.e., normalized into multiple related tables).
- This results in **more tables, more joins, and less data redundancy**.

Used in **data warehousing** when data storage efficiency and hierarchy modeling are prioritized over query performance.

Structure Overview

- **Fact Table:** Contains metrics/measures (quantitative data).
- **Dimension Tables:** Descriptive attributes but **split** into related hierarchical tables (normalized form).
- **Sub Dimension Tables:** Dimension tables are further normalized into smaller, more specific tables, creating a hierarchy.

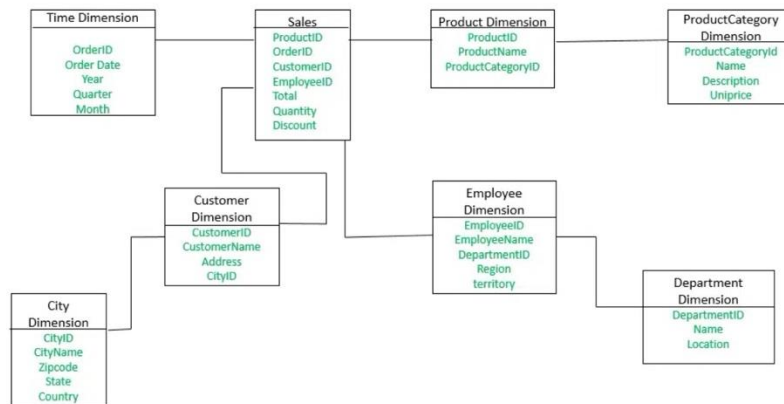
Model:



Example :

Example of Snowflake Schema

The **Employee** dimension table now contains the attributes: EmployeeID, EmployeeName, DepartmentID, Region, and Territory. The DepartmentID attribute links with the **Employee** table with the **Department** dimension table. The **Department** dimension is used to provide detail about each department, such as the Name and Location of the department. The **Customer** dimension table now contains the attributes: CustomerID, CustomerName, Address, and CityID. The CityID attributes link the **Customer** dimension table with the **City** dimension table. The **City** dimension table has details about each city such as city name, Zipcode, State, and Country.



Normalized Model:

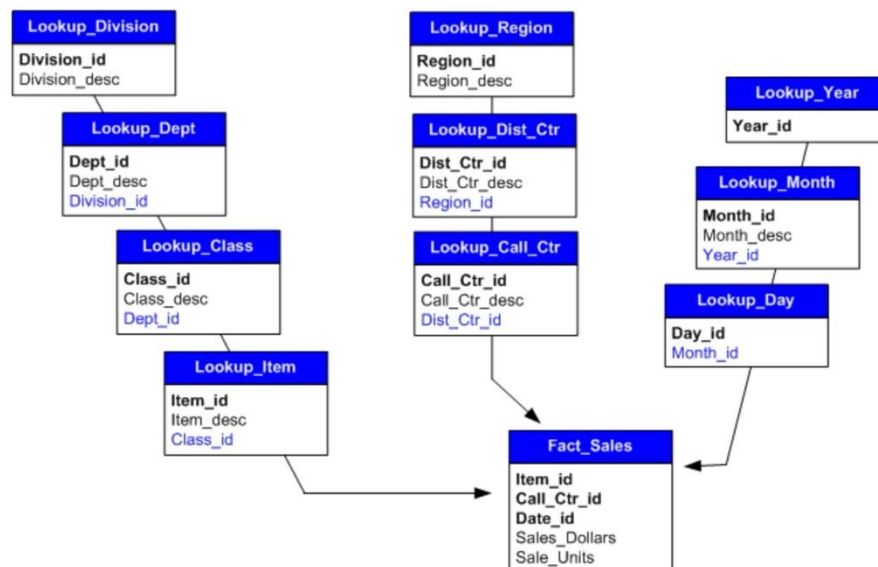
A **Normalized Schema** is a data model used primarily in **transactional (OLTP)** systems. Its design follows **normalization rules** (1NF, 2NF, 3NF, etc.) to **reduce redundancy** and **ensure data integrity** by organizing data into **multiple related tables**.

Feature: Eliminate data anomalies, improve consistency, and support high-speed insert/update/delete operations.

Feature	Description
Redundancy	Minimized by splitting data into logical entities
Relationships	Data stored across multiple related tables
Joins	Requires more joins for queries
Storage Efficiency	High (less duplicate data)
integrity	Strong enforcement of referential integrity
Use Case	OLTP systems (e.g., banking, retail POS, ERP systems)

Highly normalized schema: Minimal storage space

The following diagram is an example of a highly normalized schema. In highly normalized schemas, lookup tables contain unique developer-designed attribute keys, such as `Call_Ctr_id`, `Dist_Ctr_id`, and `Region_id`, as shown in the figure below. They also contain attribute description columns, such as `Call_Ctr_desc`, `Dist_Ctr_desc`, and `Region_desc`. Also, the lookup table for an attribute contains the ID column of the parent attribute, such as `Dist_Ctr_id` in the `Lookup_Call_Ctr` table.

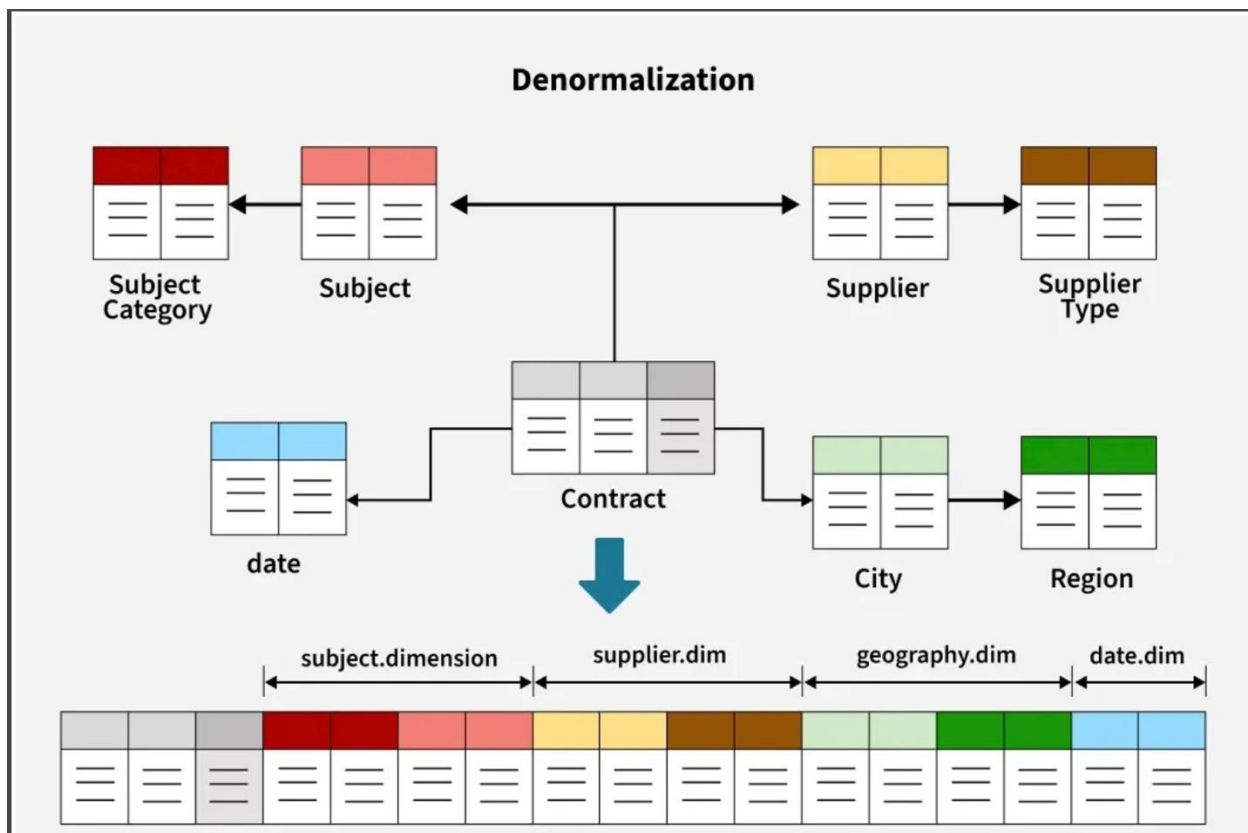


Denormalized Model:

A **Denormalized Model** is a database design where data is **intentionally duplicated** across tables to improve **read/query performance** by reducing the need for joins.

Used primarily in **OLAP**, data warehouses, reporting systems, and data marts, where **speed of access** is more important than **storage efficiency**.

Feature	Description
Redundancy	High – data is repeated
Joins	Minimized or eliminated
Read Performance	High – faster queries
Write Operations	Slower and more complex (due to repeated data updates)
Use Case	Analytical systems, dashboards, BI tools
Complexity	Lower from a querying perspective



Step 1: Unnormalized Table

This is the starting point where all the data is stored in a single table.

Unnormalized Structure

StudentID	StudentName	ClassID	ClassName	TeacherName	Subject
1	Alice	C101	Math	Mr. Smith	Algebra
1	Alice	C101	Math	Mr. Smith	Geometry
2	Bob	C102	Science	Mrs. Johnson	Physics
2	Bob	C102	Science	Mrs. Johnson	Chemistry

What's wrong with it?

- **Redundancy:** For example, "Alice" and "Math" are repeated multiple times. Similarly, "Mr. Smith" is stored twice for the same class.
- **Update Anomalies:** If "Mr. Smith" changes to "Mr. Brown," we have to update multiple rows. Missing one row could lead to inconsistencies.
- **Inefficient Storage:** Repeated information takes up unnecessary space.

Step 2: Normalized Structure

To eliminate redundancy and avoid anomalies, we split the data into smaller, related tables. This process is called normalization. Each table now focuses on a specific aspect, such as students, classes, or subjects.

Normalized Structure

StudentID	StudentName
1	Alice
2	Bob

ClassID	ClassName	TeacherName
C101	Math	Mr. Smith
C102	Science	Mrs. Johnson

StudentID	ClassID
1	C101
2	C102

ClassID	Subject
C101	Algebra
C101	Geometry
C102	Physics
C102	Chemistry

Why is this better?

- **No Redundancy:** "Mr. Smith" appears only once in the Classes Table, even if multiple subjects are associated with the class.
- **Easier Updates:** If "Mr. Smith" changes to "Mr. Brown," you only update the Classes Table, and it automatically reflects everywhere.
- **Efficient Storage:** Repeated data is eliminated, saving space.

Step 3: Denormalized Table

In some cases, normalization can make querying complex and slow because you need to join multiple tables to get the required information. To optimize performance, we can denormalize the data by combining related tables into a single table.

Denormalized Structure

StudentID	StudentName	ClassName	TeacherName	Subject
1	Alice	Math	Mr. Smith	Algebra
1	Alice	Math	Mr. Smith	Geometry
2	Bob	Science	Mrs. Johnson	Physics
2	Bob	Science	Mrs. Johnson	Chemistry

What's happening here?

- All related information (student name, class name, teacher, and subject) is stored in a single table.
- This simplifies querying because you don't need to join multiple tables.

=