You will be introduced to the basic concepts of a data warehouse, star schema, OLAP (Online Analytical Processing), OLTP (Online Transactional Processing), SETL (Select, Extraction, Transformation, and Loading), constraints and ERD (Entity Relationship Diagrams).

**What is a Data Warehouse?**

Think of yourself as a data analyst working for a company that has the following three departments: Marketing, Sales and Finance. Now, let’s assume that each department maintains a separate database.

This could lead to a situation wherein each department has its own version of the facts. For a question such as 'What is the total revenue of the last quarter?', every department might have a different answer. This is because each department draws information from a different database.

This is where a data warehouse can prove to be useful. It can help with creating a single version of the truth and the facts. A data warehouse would thus be the central repository of data of the entire enterprise.

So, a data warehouse is a collection of data. It has the following properties:

* **Subject-oriented:** A data warehouse should contain information about a few well-defined subjects rather than the enterprise.
* **Integrated:** A data warehouse is an integrated repository of data. It contains information from various systems within an organisation.
* **Non-volatile:** The data values in a database cannot be changed without a valid reason.
* **Time-variant:** A data warehouse contains historical data for analysis.

**Structure of a Data Warehouse**

In the previous segment, you learnt about the basic concepts of data warehousing. Now, one of the primary methods of designing a data warehouse is **dimensional modelling**.

The two key elements of dimensional modelling include **facts**and **dimensions**, which are basically the different types of variables that are used to design a data warehouse. They are arranged in a specific manner, known as a **schema diagram**.So, in the following video, you will learn more about facts and dimensions.

1. The numerical variables/attributes are called fact variables.
2. Dimension variables add life to the fact variables.
   1. Examples: investor\_name, company\_name, country\_name etc
3. Dimensions can be thought of metadata of the fact variables
4. So, essentially, facts are the numerical data in a data warehouse and dimensions are the metadata (that is, data explaining some other data) attached to the fact variables. Both facts and dimensions are equally important for generating actionable insights from a data set.

**Star Schema**

In the previous segment, you learnt about **facts** and **dimensions**, which are the two key elements of dimension modelling. Now, a typical problem might involve multiple databases with many different variables, but we may not be interested in all of them. Hence, only some facts and dimensions are combined in a specific manner to build the structure of a data warehouse, called a schema diagram.

A schema is anoutline of the entire data warehouse. It shows how different data sets are connected and how the different attributes of each data set are used for the data warehouse.

In the upcoming video, you will see an example of how an e-commerce company can design a data warehouse.

**OLAP vs OLTP**

Now that you have developed a fair understanding of databases, you might be wondering how a data warehouse and a database differ from each other.

So, in this segment, you will learn exactly which features of a data warehouse differentiate it from a regular database. In the upcoming video, our expert will talk about the differences between a transactional database and a data warehouse.

OLTP : online transaction processing systems

OLAP: online analytical processing systems

**Key Differences:**

* **Purpose**:
  + **Database**: General-purpose, stores and manages data.
  + **Transactional Database**: Focused on high-speed, reliable transaction processing.
  + **Data Warehouse**: Focused on analyzing large volumes of data and supporting decision-making processes.
* **Data Structure**:
  + **Database**: Can be either relational or non-relational.
  + **Transactional Database**: Relational with strict normalization.
  + **Data Warehouse**: Often denormalized, optimized for complex queries.
* **Use Case**:
  + **Database**: Versatile, used across various applications.
  + **Transactional Database**: Used in environments with frequent data transactions (e.g., e-commerce).
  + **Data Warehouse**: Used for data analysis, business intelligence, and reporting.
* you learnt about the differences between a transactional database (i.e., **OLTP, or Online Transactional Processing**) and a data warehouse (which is often referred to as **OLAP, or Online Analytical Processing**).  Notice that the major difference between OLAP and OLTP is apparent in the names themselves: OLTP is used for day-to-day transactions, whereas OLAP is used for analytical purposes.
* Earlier, you were introduced to the terms **dimensional modelling** and **star schema.** They are essential for creating the structure of a data warehouse. These techniques involve finding out the variables on which analysis can be performed and then combining them with the metadata to derive meaningful insights. You will learn more about them in the next session.

|  |  |
| --- | --- |
| **OLTP System  Online Transaction Processing  (Operational System)** | **OLAP System  Online Analytical Processing  (Data Warehouse)** |
| Source of data | Operational data; OLTPs are the original source of the data. | Consolidation data; OLAP data comes from the various OLTP Databases |
| Purpose of data | To control and run fundamental business tasks | To help with planning, problem solving, and decision support |
| What the data | Reveals a snapshot of ongoing business processes | Multi-dimensional views of various kinds of business activities |
| Inserts and Updates | Short and fast inserts and updates initiated by end users | Periodic long-running batch jobs refresh the data |
| Queries | Relatively standardized and simple queries Returning relatively few records | Often complex queries involving aggregations |
| Processing Speed | Typically very fast | Depends on the amount of data involved; batch data refreshes and complex queries may take many hours; query speed can be improved by creating indexes |
| Space Requirements | Can be relatively small if historical data is archived | Larger due to the existence of aggregation structures and history data; requires more indexes than OLTP |
| Database Design | Highly normalized with many tables | Typically de-normalized with fewer tables; use of star and/or snowflake schemas |
| Backup and Recovery | Backup religiously; operational data is critical to run the business, data loss is likely to entail significant monetary loss and legal liability | Instead of regular backups, some environments may consider simply reloading the OLTP data as a recovery method |

**SETL: Select, Extract, Transform and Load**.

* Select: Identification of the data that you want to analyse
* Extract: Connecting to the particular data source and pulling out the data
* Transform: Modifying the extracted data to standardise it
* Load: Pushing the data into the data warehouse

This process includes the typical operations that are involved in selecting the required data, extracting data from multiple sources, operating on the data so that data from multiple sources is compatible, and loading this data into a data warehouse for analytical purposes.

* **Unique:** This constraint is used for columns that need unique values. For example, 'employee ids' should be unique in an 'employees' table.
* **Null:** This constraint is used to determine the columns that can have null values. For example, an employee may not need to specify their location, which means the 'location' column can have null values in an 'employees' table.
* **Primary Key:** This constraint is used to determine the column that uniquely identifies a table. For example, 'employee ids' uniquely identify every employee. Two employees may have the same name or the same salary, but not the same employee id.

Entity constraints : unique, null check, primary key

Referential constraints: foreign key

Semantic constraints: extra check on the columns

An **Entity-Relationship Diagram**,**or ERD**,can be thought of as a map of the database schema. We can visualise the structure of the entire schema and answer the following questions just by looking at the ERD:

* What are the tables that it contains?
* What are the columns that each table contains?
* What is/are the data types and constraint/s (if any) for each column?
* What are the relationships between the various tables?

An **Entity-Relationship Diagram**,**or ERD**,can be thought of as a map of the database schema. We can visualise the structure of the entire schema and answer the following questions just by looking at the ERD:

* What are the tables that it contains?
* What are the columns that each table contains?
* What is/are the data types and constraint/s (if any) for each column?
* What are the relationships between the various tables?

So, as you learnt in the video, the constituents of an ERD are as follows:

* **Entity Type/Entity:** It is nothing but a table in the schema. For example, 'orders' and 'payments' are both entity types.
* **Attribute:** It is a column in an entity type. For example, 'orderNumber' is an attribute in the 'orders' entity type.
* **Relationship Types:** They are the lines between the tables. They define the relationships among the tables. These can be of various types based on their cardinalities, i.e., **one-to-one**, **one-to-many**, **many-to-many**, etc

SQL supports several wildcards that can be used in conjunction with the `LIKE` operator to search for patterns in text data. The common wildcards used in SQL are:

### 1. \*\*Percent (%)\*\*

- \*\*Purpose\*\*: Represents zero, one, or multiple characters.

- \*\*Example\*\*:

```sql

SELECT \* FROM Employees WHERE Name LIKE 'J%';

```

- Finds all names starting with 'J', such as "John", "Jane", "Jack".

### 2. \*\*Underscore (\_)\*\*

- \*\*Purpose\*\*: Represents exactly one character.

- \*\*Example\*\*:

```sql

SELECT \* FROM Employees WHERE Name LIKE '\_a%';

```

- Finds all names where the second letter is 'a', such as "Mark", "Jack", "Jane".

### 3. \*\*Square Brackets ([ ])\*\*

- \*\*Purpose\*\*: Represents any single character within the brackets.

- \*\*Example\*\*:

```sql

SELECT \* FROM Employees WHERE Name LIKE '[Jj]ohn';

```

- Finds all names that match "John" or "john".

### 4. \*\*Caret (^) within Square Brackets\*\*

- \*\*Purpose\*\*: Represents any single character not within the brackets.

- \*\*Example\*\*:

```sql

SELECT \* FROM Employees WHERE Name LIKE '[^J]ohn';

```

- Finds all names that do not start with 'J', but end with "ohn", such as "Kohn", "Bohn".

### 5. \*\*Hyphen (-) within Square Brackets\*\*

- \*\*Purpose\*\*: Represents a range of characters.

- \*\*Example\*\*:

```sql

SELECT \* FROM Employees WHERE Name LIKE '[A-C]%';

```

- Finds all names starting with 'A', 'B', or 'C'.

### Common Wildcards in SQL:

1. \*\*%\*\* - Zero, one, or many characters.

2. \*\*\_\*\* - Exactly one character.

3. \*\*[ ]\*\* - Any single character within the brackets.

4. \*\*[^ ]\*\* - Any single character not within the brackets.

5. \*\*[-]\*\* - Range of characters within the brackets.

These wildcards are used within the `LIKE` clause to perform flexible pattern matching in SQL queries.