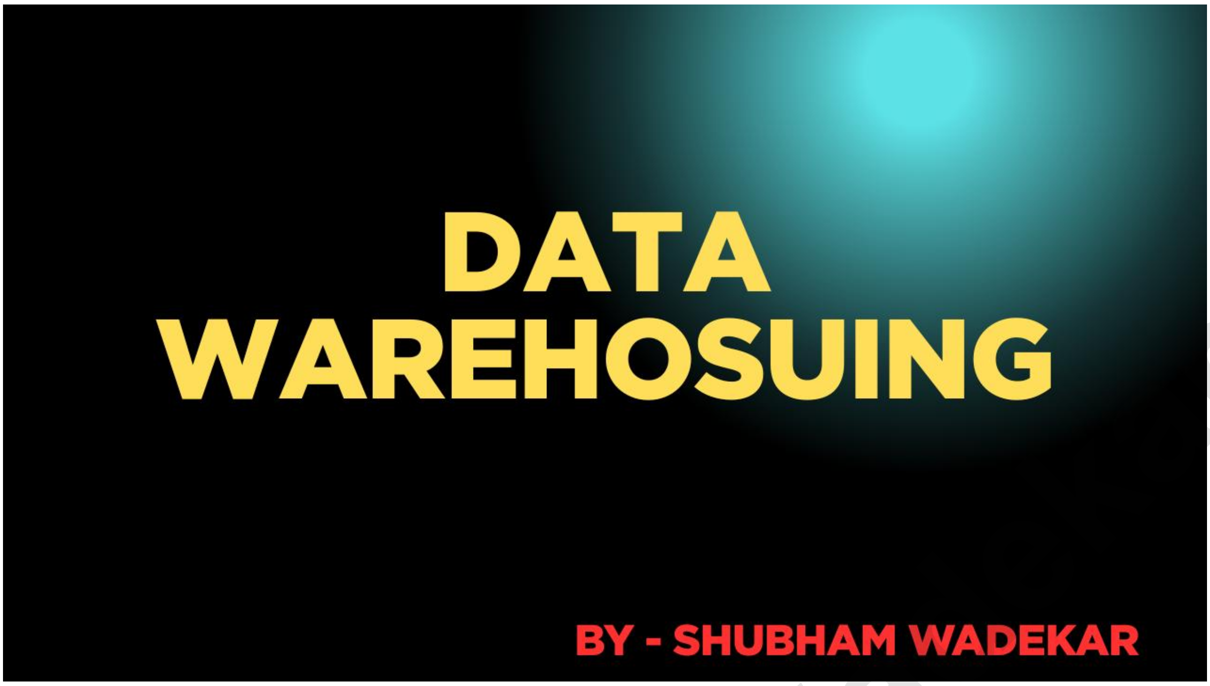

DATA
WAREHOSUING
BY - SHUBHAM WADEKAR

1. What is a Data Warehouse?

A data warehouse is a large, centralized repository of data that is used to support data-driven decision-making and business intelligence (BI) activities. It is designed to provide a single, comprehensive view of all the data in an organization, and to allow users to easily analyze and report on that data.

Data warehouses are typically used to store historical data and are optimized for fast query and analysis performance. They often contain data from multiple sources and may include both structured and unstructured data. Data in a data warehouse is imported from operational systems and external sources, rather than being created within the warehouse itself. Importantly, data is copied into the warehouse, not moved, so it remains in the source systems as well.

Data warehouses follow a set of rules proposed by Bill Inmon in 1990. These rules are:

1. Integrated: They combine data from different source systems into a unified environment.

2. Subject-oriented: Data is reorganized by subjects, making it easier to analyze specific topics or areas of interest.

3. Time-variant: They store historical data, not just current data, allowing for trend analysis and tracking changes over time.

4. Non-volatile: Data warehouses remain stable between refreshes, with new and updated data loaded periodically in batches. This ensures that the data does not change during analysis, allowing for consistent strategic planning and decision-making.

As data is imported into the data warehouse, it is often restructured and reorganized to make it more useful for analysis. This process helps to optimize the data for querying and reporting, making it easier for users to extract valuable insights from the data.

2. Why do we need a Data Warehouse?

Primary reasons for investing time, resources, and money into building a data warehouse:

1. Data-driven decision-making: Data warehouses enable organizations to make decisions based on data, rather than solely relying on experience, intuition, or hunches.

2. One-stop shopping: A data warehouse consolidates data from various transactional and operational applications into a single location, making it easier to access and analyze the data.

Data warehouses provide a comprehensive view of an organization's past, present, and potential future/forecast data. They also offer insights into unknown patterns or trends through advanced analytics and Business Intelligence (BI). In conclusion, Business Intelligence and data warehousing are closely related disciplines that provide immense value to organizations by facilitating data-driven decision-making and offering a centralized data repository for analysis.

3. Data Warehouse vs Data Lake

Let's discuss the similarities and differences between data warehouses and data lakes, two valuable tools in data management.

A data warehouse is often built on top of a relational database, such as Microsoft SQL Server, Oracle, or IBM DB2. These databases are used for both transactional systems and data warehousing, making them versatile data management tools. Sometimes, data warehouses are built on multidimensional databases called "cubes," which are specialized databases.

In contrast, a data lake is built on top of a big data environment rather than a traditional relational database. Big data environments allow for the management of extremely large volumes of data, rapid intake of new and changing data, and support a variety of data types (structured, semi-structured, and unstructured).

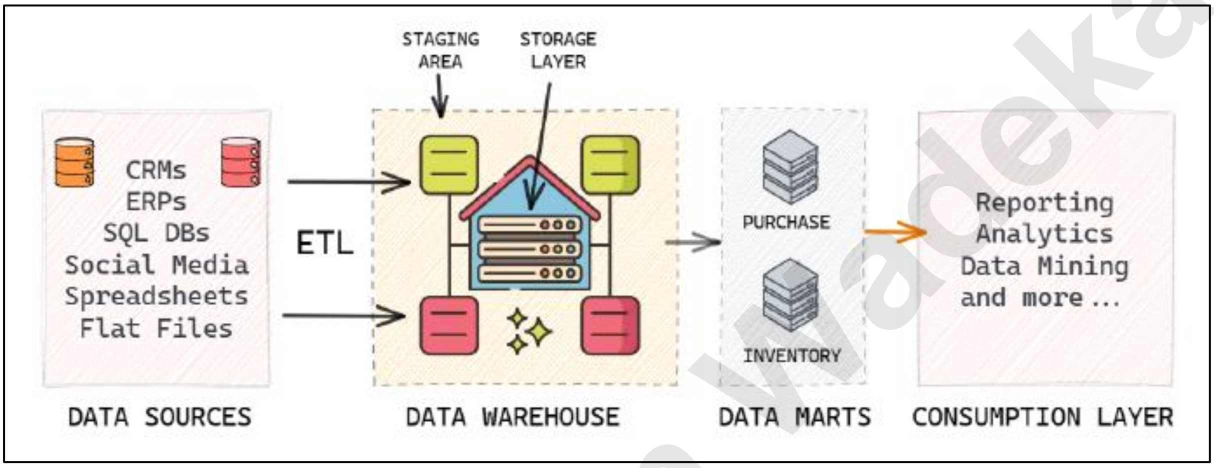
The lines between the two are increasingly blurred, as SQL, the standard relational database language, can be used on both data warehouses and data lakes. From a user perspective, traditional Business Intelligence (BI) can be performed against either a data warehouse or a data lake.

Feature	Data Warehouse	Data Lake
Data Structure	**Structured** (schema-on-write)	**Raw** (structured, semi-structured, unstructured)
Purpose	Business intelligence, reporting, analytics	Big data, ML, advanced analytics
Users	Business analysts, executives	Data scientists, engineers
Processing	Processed/cleaned before storage	Stored raw, processed when needed
Schema	Fixed schema (rigid)	Flexible schema (schema-on-read)
Storage Cost	Higher (processed data)	Lower (raw data)
Performance	Optimized for SQL queries	Optimized for big data processing
Data Quality	High (cleaned, validated)	Variable (may include raw, dirty data)
Best For	Structured reporting, dashboards	AI/ML, IoT, exploratory analysis
Example Tools	Snowflake, Redshift, BigQuery	Hadoop, Spark, Azure Data Lake

4. Simple End-to-End Data Warehouse Environment

A simple end-to-end data warehousing environment consists of data sources, a data warehouse, and sometimes, smaller environments called data marts. The process connecting data sources to the data warehouse is known as ETL (Extract, Transform, and Load), a critical aspect of data warehousing.

An analogy to understand this relationship is to think of data sources as suppliers, the data warehouse as a wholesaler that collects data from various suppliers, and data marts as data retailers. Data marts store specific subsets of data tailored for different user groups or business functions. Users typically access data from these data marts for their data-driven decision-making processes.



5. Data Mart

Data Mart is referred to as a pattern to get client data in a data warehouse environment. It's a data warehouse-specific structure that's employed by the team's business domain. Every company has its own data mart, which is kept in the data warehouse repository. Dependent, independent, and hybrid data marts are the three types of data marts. Independent data marts collect data from external sources and data warehouses, whereas dependent data marts take data that has already been developed. Data marts can be thought of as logical subsets of a data warehouse.

6. What are the different types of data marts in the context of data warehousing?

Dependent Data Mart: A dependent data mart can be developed using data from operational, external, or both sources. It enables the data of the source company to be accessed from a single data warehouse. All data is centralized, which can aid in the development of further data marts.

Independent Data Mart: There is no need for a central data warehouse with this data mart. This is typically established for smaller groups that exist within a company. It has no connection to Enterprise Data Warehouse or any other data warehouse. Each piece of information is self-contained and can be used independently. The analysis can also be carried out independently. It's critical to maintain a consistent and centralized data repository that numerous users can access.

Hybrid Data Mart: A hybrid data mart is utilized when a data warehouse contains inputs from multiple sources, as the name implies. When a user requires an ad hoc integration, this feature comes in handy. This solution can be utilized if an organization requires various database environments and quick implementation. It necessitates the least amount of data purification, and the data mart may accommodate huge storage structures. When smaller data-centric applications are employed, a data mart is most effective.

7. What do you mean by data mining? Differentiate between data mining and data warehousing.

Data mining is the process of collecting information in order to find patterns, trends, and usable data that will help a company to make data-driven decisions from large amounts of data. In other words, Data Mining is the method of analysing hidden patterns of data from various perspectives for categorization into useful data, which is gathered and assembled in specific areas such as data warehouses, efficient analysis, data mining algorithm, assisting decision making, and other data requirements, ultimately resulting in cost-cutting and revenue generation. Data mining is the process of automatically examining enormous amounts of data for patterns and trends that go beyond simple analysis. Data mining estimates the probability of future events by utilising advanced mathematical algorithms for data segments.

Following are the differences between data warehousing and data mining –

Feature	Data Warehousing	Data Mining
Definition	A centralized repository for structured, historical data	The process of discovering patterns in large datasets
Primary Purpose	Data storage and organization for analysis	Extracting insights and knowledge from data
Focus	Data integration and management	Pattern recognition and prediction
Process	ETL (Extract, Transform, Load)	Statistical analysis, machine learning
Data Type	Structured, cleaned, and processed	Raw or processed data
Users	Business analysts, executives	Data scientists, analysts
Output	Reports, dashboards, visualizations	Predictive models, trends, correlations
Technologies	SQL, OLAP, ETL tools	Machine learning, clustering, classification

Feature	Data Warehousing	Data Mining
Time Orientation	Historical data storage	Future predictions and trends
Example Use Case	Sales reporting, financial consolidation	Customer segmentation, fraud detection

8. OLTP VS OLAP

Feature	OLTP (Online Transaction Processing)	OLAP (Online Analytical Processing)
Purpose	Handles **real-time operational transactions** (e.g., orders, payments).	Supports **complex analytics & reporting** (e.g., trends, forecasts).
Data Type	**Current, detailed, and frequently updated** (short-term).	**Historical, aggregated, and read-only** (long-term).
Database Design	**Normalized** (3NF) to avoid redundancy.	**Denormalized** (star/snowflake schema) for fast queries.
Query Complexity	Simple, frequent, and fast (e.g., INSERT, UPDATE, DELETE).	Complex, read-heavy, and slow (e.g., multi-table joins, aggregations).
Performance Focus	**High-speed writes**, low latency for transactions.	**Optimized for reads**, handles large datasets efficiently.
User Base	Front-line staff (e.g., cashiers, clerks).	Analysts, executives, data scientists.
Example Systems	POS systems, banking transactions, e-commerce orders.	Data warehouses, BI tools (Power BI, Tableau), dashboards.
Backup Needs	**Critical** (daily/hourly backups due to live data).	**Less frequent** (historical data changes rarely).
Storage Size	Smaller (stores only current operational data).	Larger (retains years of historical data).

9. What are the advantages of a data warehouse?

Following are the advantages of using a data warehouse:

Helps you save time:

- To stay ahead of your competitors in today's fast-paced world of cutthroat competition, your company's ability to make smart judgments quickly is critical.
- A Data warehouse gives you instant access to all of your essential data, so you and your staff don't have to worry about missing a deadline. All you have to do now is deploy your data model to start collecting data in a matter of seconds. You can do this with most warehousing solutions without utilising a sophisticated query or machine learning.
- With data warehousing, your company won't have to rely on a technical professional to troubleshoot data retrieval issues 24 hours a day, seven days a week. You will save a lot of time this way.

Enhances the quality of data:

- The high-quality data ensures that your company's policies are founded on accurate information about your operations.
- You can turn data from numerous sources into a shared structure using data warehousing. You can assure the consistency and integrity of your company's data this way. This allows you to spot and eliminate duplicate data, inaccurately reported data and disinformation.
- For your firm, implementing a data quality management program may be both costly and time-consuming. You can easily use a data warehouse to reduce the number of these annoyances while saving money and increasing the general productivity of your company.

Enhances Business Intelligence (BI):

- Throughout your commercial endeavours, you can use a data warehouse to gather, absorb, and derive data from any source. As a result of the capacity to easily consolidate data from several sources, your BI will improve by leaps and bounds.

Data standardization and Consistency are achieved:

- The uniformity of huge data is another key benefit of having central data repositories. In a similar manner, a data storage or data mart might benefit your company. Because data warehousing stores data from various sources in a consistent manner, such as a transactional system, each source will produce results that are synchronized with other sources. This ensures that data is of higher quality and homogeneous. As a result, you and your team can rest assured that your data is accurate, resulting in more informed corporate decisions.

Enhances Data Security:

- A data warehouse improves security by incorporating cutting-edge security features into its design. For any business, consumer data is a vital resource. You can keep all of your data sources integrated and properly protected by adopting a warehousing solution. The risk of a data breach will be greatly reduced as a result of this.

Ability to store historical data:

- Because a data warehouse can hold enormous amounts of historical data from operational systems, you can readily study different time periods and inclinations that could be game-changing for your business. You can make better corporate judgments about your business plans if you have the correct facts in your hands.

10. What are the disadvantages of using a data warehouse?

Following are the disadvantages of using a data warehouse: -

Loading time of data resources is undervalued

- We frequently underestimate the time it will take to gather, sanitize, and post data to the warehouse. Although some resources are in place to minimize the time and effort spent on the process, it may require a significant amount of the overall production time.

Source system flaws that go unnoticed

- After years of non-discovery, hidden flaws linked with the source networks that provide the data warehouse may be discovered. Some fields, for example, may accept nulls when entering new property information, resulting in workers inputting incomplete property data, even if it was available and relevant.

Homogenization of data

- Data warehousing also deals with data formats that are comparable across diverse data sources. It's possible that some important data will be lost as a result.

11. What are the different types of data warehouse?

Enterprise Data Warehouse:

An enterprise database is a database that brings together the various functional areas of an organisation in a cohesive manner. It's a centralised location where all corporate data from various sources and apps can be accessed. They can be utilised for analytics and by everyone in the organisation once they've been saved. The data can be categorised by subject, and access is granted according to the necessary division. The tasks of extracting, converting, and conforming are taken care of in an Enterprise Datawarehouse.

Enterprise Datawarehouse's purpose is to provide a comprehensive overview of any object in the data model. This is performed by finding and wrangling the data from different systems. This is then loaded into a model that is consistent and conformed. The data is acquired by Enterprise Datawarehouse, which can provide access to a single site where various tools can be used to execute analytical functions and generate various predictions. New trends or patterns can be identified by research teams, which can then be focused on to help the company expand.

Operational Data Store (ODS):

An operational data store is utilised instead of having an operational decision support system application. It facilitates data access directly from the database, as well as transaction processing. By checking the associated business rules, the data in the Operational Data Store may be cleansed, and any redundancy found can be checked and rectified. It also aids in the integration of disparate data from many sources so that business activities, analysis, and reporting may be carried out quickly and effectively while the process is still ongoing.

The majority of current operations are stored here before being migrated to the data warehouse for a longer period of time. It is particularly useful for simple searches and little amounts of data. It functions as short-term or temporary memory, storing recent data. The data warehouse keeps data for a long time and also keeps information that is generally permanent.

Data Mart:

Data Mart is referred to as a pattern to get client data in a data warehouse environment. It's a data warehouse-specific structure that's employed by the team's business domain. Every company has its own data mart, which is kept in the data warehouse repository. Dependent, independent, and hybrid data marts are the three types of data marts. Independent data marts collect data from external sources and data warehouses, whereas dependent data marts take data that has already been developed. Data marts can be thought of as logical subsets of a data warehouse.

12. Datawarehouse vs database

Feature	Data Warehouse	Database
Purpose	Analytical processing (OLAP)	Transactional processing (OLTP)
Data Type	Historical, aggregated data	Current, operational data
Data Structure	Optimized for complex queries	Optimized for fast transactions
Schema Design	Denormalized (star/snowflake schema)	Normalized (3NF or higher)
Data Volume	Large (TB-PB scale)	Small-Medium (GB-TB scale)
Write Operations	Batch loads (ETL processes)	Continuous real-time updates
Read Operations	Complex analytical queries	Simple, frequent record lookups
Users	Business analysts, data scientists	Application users, clerks

Feature	Data Warehouse	Database
Query Performance	Optimized for read-heavy workloads	Optimized for write-heavy workloads
Data Timeframe	Years of historical data	Days/weeks of current data
Example Systems	Snowflake, Redshift, BigQuery	MySQL, PostgreSQL, Oracle

13. What are the characteristics of a data warehouse?

Subject-oriented: Because it distributes information about a theme rather than an organization's actual operations, a data warehouse is always subject-oriented. It is possible to do so with a certain theme. That is to say, the data warehousing procedure is intended to deal with a more defined theme. These themes could include sales, distribution, and marketing, for example. The focus of a data warehouse is never solely on present activities. Instead, it concentrates on demonstrating and analyzing evidence in order to reach diverse conclusions. It also provides a simple and precise demonstration around a specific theme by removing info that isn't needed to make conclusions.

Integrated: It is similar to subject orientation in that it is created in a dependable format. Integration entails the creation of a single entity to scale all related data from several databases. The data has to be stored in several data warehouses in a shared and widely accessible manner. A data warehouse is created by combining information from a variety of sources, such as a mainframe and a relational database. It must also have dependable naming conventions, formats, and codes. The utilization of a data warehouse allows for more effective data analysis. The consistency of name conventions, column scaling, and encoding structure, among other things, should be validated. The data warehouse integration handles a variety of subject-related warehouses.

Time-Variant: Data is kept in this system at various time intervals, such as weekly, monthly, or annually. It discovers a number of time limits that are structured between massive datasets and held in the online transaction process (OLTP). Data warehouse time limitations are more flexible than those of operational systems. The data in the data warehouse is predictable over a set period of time and provides information from a historical standpoint. It contains explicit or implicit time elements. Another property of time-variance is that data cannot be edited, altered, or updated once it has been placed in the data warehouse.

Non-volatile: The data in a data warehouse is permanent, as the name implies. It also means that when new data is put, it is not erased or removed. It incorporates a massive amount of data that is placed into logical business alteration between the designated quantity. It assesses the analysis in the context of warehousing technologies. Data is read-only and refreshed at scheduled intervals. This is useful for analyzing historical data and understanding how things work. It is not required to have a transaction process, a recapture mechanism, or a concurrency control mechanism. In a data warehouse environment, operations like delete, update, and insert that are performed in an operational application are lost

14. Enlist a few data warehouse solutions that are currently being used in the industry.

Some of the major data warehouse solutions currently being used in the industry are as follows

- Snowflakes
- Oracle Exadata
- Apache Hadoop
- SAP BW4HANA
- Microfocus Vertica
- Teradata
- AWS Redshift
- GCP Big Query

15. Enlist some of the renowned ETL tools currently used in the industry.

Some of the renowned ETL tools currently used in the industry are as follows

- Informatica
- Talend
- Pentaho
- Abnitio
- Oracle Data Integrator
- Xplenty
- Skyvia
- Microsoft – SQL Server Integrated Services (SSIS)

16. What do you understand about a data cube in the context of data warehousing?

A data cube is a multidimensional data model that stores optimized, summarized, or aggregated data for quick and easy analysis using OLAP technologies. The precomputed data is stored in a data cube, which makes online analytical processing easier. We all think of a cube as a three-dimensional structure, however in data warehousing, an n-dimensional data cube can be implemented. A data cube stores information in terms of dimensions and facts.

Data Cubes have two categories. They are as follows:

Multidimensional Data Cube: Data is stored in multidimensional arrays, which allows for a multidimensional view of the data. A multidimensional data cube aids in the storage of vast amounts of information. A multidimensional data cube uses indexing to represent each dimension of the data cube, making it easier to access, retrieve, and store data.

Relational Data Cube: The relational data cube can be thought of as an "expanded version of relational DBMS." Data is stored in relational tables, and each relational table represents a data cube's dimension. The relational data cube uses SQL to produce aggregated data, although it is slower than the multidimensional data cube in terms of performance. The relational data cube, on the other hand, is scalable for data that grows over time.

17. ETL VS ELT

Feature	ETL (Extract, Transform, Load)	ELT (Extract, Load, Transform)
Processing Order	Transform before loading	Load raw data first, then transform
Transformation	Happens in a separate processing engine	Happens within the target data system
Data Volume	Best for small/medium datasets	Handles large/big data efficiently
Flexibility	Less flexible (schema-on-write)	More flexible (schema-on-read)
Implementation	Requires staging area	No staging area needed
Latency	Higher latency (transforms first)	Lower latency (loads first)
Cost	Higher (requires processing power)	Lower (leverages target system power)
Use Cases	Traditional data warehousing	Modern data lakes/cloud warehouses
Tools	Informatica, SSIS, Talend	Snowflake, BigQuery, Databricks
Data Quality	Ensures clean data before loading	May contain raw/uncleaned data
Maintenance	Complex to maintain	Easier to maintain

18. Data warehousing data loads –

Data loading is the process of moving data from source systems into a data warehouse. The approach you choose impacts performance, storage costs, and data freshness. Here's a detailed breakdown:

1. Full Load

Definition: Complete replacement of all existing data in the target table with fresh data from the source.

How it Works:

1. Truncates the target table
2. Extracts ALL records from source
3. Loads entire dataset into target

Characteristics:

- Simple to implement
- Guarantees data consistency
- Resource-intensive (processes all data every time)
- No tracking of changes needed

When to Use:
✓ Small datasets
✓ Initial load
✓ When source doesn't track changes
✓ When data changes completely between loads

Example:

TRUNCATE TABLE customers;

INSERT INTO customers SELECT * FROM source_customers

2. Upsert (Merge Load)

Definition: Combination of update existing records and insert new records (UPDATE + INSERT = UPSERT).

How it Works:

1. Compares source and target using key fields
2. Updates matching records
3. Inserts non-matching records

Characteristics:

- Maintains data history
- More complex than full load
- Requires unique key for matching
- Efficient for slowly changing dimensions

When to Use:
✓ When you need to maintain history
✓ For slowly changing dimensions
✓ When only some records change

Example (SQL MERGE):

```
MERGE INTO target_table t

USING source_table s

ON t.id = s.id

WHEN MATCHED THEN UPDATE SET t.col1 = s.col1, t.col2 = s.col2

WHEN NOT MATCHED THEN INSERT (id, col1, col2) VALUES (s.id, s.col1, s.col2);
```

3. Incremental Load (Delta Load)

Definition: Only loads new or changed records since last load.

How it Works:

1. Identifies changed records using:
    - Timestamps (last_modified_date)
    - Change Data Capture (CDC)
    - Log-based tracking
2. Extracts only delta (changes)
3. Applies changes to target

Characteristics:

- Most efficient for large datasets
- Requires robust change tracking
- Complex to implement
- Risk of missing changes if not properly tracked

When to Use:
✓ Large datasets
✓ Frequent loads
✓ When source tracks changes

Example:

```
INSERT INTO sales
SELECT * FROM source_sales
WHERE sale_date > (SELECT MAX(sale_date) FROM sales);
```

Feature	Full Load	Upsert	Incremental Load
Data Volume	Processes all data	Processes changed data	Processes only new/changed data
Performance	Slowest	Moderate	Fastest
Complexity	Simplest	Moderate	Most complex
Storage	Replaces all data	Updates existing	Adds new records
Risk	Data loss if failed	Complex logic	Missed changes if tracking fails
Best For	Small/static datasets	Slowly changing dimensions	Large/dynamic datasets

19. Data Warehousing Storage Layouts: Row vs. Column vs. Hybrid Stores

1. Row-Store (Row-Oriented Storage)

How It Works

- Stores data row by row (all columns of a record are stored together)
- Traditional approach used in OLTP databases (e.g., MySQL, PostgreSQL)

Characteristics

✅ Fast for transactional operations (INSERT/UPDATE/DELETE)
✅ Efficient for retrieving full rows (e.g., fetching a customer's complete record)
❌ Inefficient for analytical queries (scans entire rows even if only a few columns are needed)
❌ Poor compression (redundant data in columns isn't optimized)

When to Use?

OLTP systems (transaction-heavy workloads)
Queries fetching entire rows (e.g., SELECT * FROM customers WHERE id = 100)

Example Storage Layout

RowID	CustomerID	Name	Age	City
1	1001	Alice	30	New York
2	1002	Bob	25	London

2. Column-Store (Column-Oriented Storage)

How It Works

- Stores data column by column (all values of a single column are stored together)
- Optimized for OLAP workloads (e.g., Snowflake, Redshift, BigQuery)

Characteristics

✅ Superior compression (similar data values in columns compress well)
✅ Fast for analytical queries (only reads required columns)
✅ Efficient aggregations (e.g., SUM(sales), AVG(price))
❌ Slower for row-level updates (modifying a single record requires rewriting columns)

When to Use?

✓ Data warehousing & analytics
✓ Queries scanning large datasets but few columns (e.g., SELECT SUM(revenue) FROM sales)

Example Storage Layout

CustomerID	1001	1002	1003
Name	Alice	Bob	Carol
Age	30	25	28
City	NY	London	Paris

3. Hybrid Store (Row + Column Storage)

How It Works

- Combines row-store and column-store in a single system
- Example: SQL Server (with columnstore indexes), Oracle (In-Memory Column Store)

Characteristics

✅ Balances transactional & analytical performance
✅ Supports both OLTP & OLAP workloads
❌ More complex to manage
❌ Higher storage overhead

When to Use?

✓ Mixed workloads (both transactions and analytics)
✓ Real-time analytics on transactional data

Example Implementation

- Transactional queries use row-store for fast updates.
- Analytical queries use column-store for fast scans.

20. ACID Properties in Databases

The ACID (Atomicity, Consistency, Isolation, Durability) properties are fundamental principles that ensure reliable transaction processing in databases. These properties collectively guarantee that database transactions are processed securely, even in cases of system failures or concurrent operations.

1. Atomicity

Atomicity ensures that a database transaction is treated as an indivisible unit. This means that either all operations within the transaction are completed successfully, or none are applied at all. If any part of the transaction fails, the entire transaction is rolled back to its original state, leaving no partial updates.

For example, in a bank transfer transaction, where money is debited from one account and credited to another, atomicity ensures that both operations succeed together. If the credit operation fails after the debit has occurred, the entire transaction is reversed, preventing data inconsistency. Databases achieve atomicity using transaction logs, which record changes so they can be undone (rolled back) in case of failure.

2. Consistency

Consistency ensures that a transaction brings the database from one valid state to another, adhering to all defined rules, constraints, and schemas. This means that any data written to the database must comply with integrity constraints, such as foreign keys, unique constraints, and check conditions.

For instance, if a database enforces a rule that an employee's salary cannot be negative, any transaction attempting to insert a negative salary will be aborted, keeping the database in a valid state. Consistency is maintained through constraints, triggers, and validation checks before and after transactions.

3. Isolation

Isolation ensures that concurrent transactions (multiple transactions happening simultaneously) do not interfere with each other. Each transaction executes as if it were the only one running, preventing issues like dirty reads, non-repeatable reads, and phantom reads.

Databases implement isolation using locking mechanisms or Multi-Version Concurrency Control (MVCC). Different isolation levels (Read Uncommitted, Read Committed, Repeatable Read, Serializable) control the degree of strictness in isolating transactions. For example, the Serializable level ensures complete isolation by executing transactions sequentially, while Read Committed allows higher concurrency but may permit some anomalies.

4. Durability

Durability guarantees that once a transaction is committed, its effects remain permanent even in the event of system crashes, power failures, or other disruptions. This is achieved by writing transaction logs to non-volatile storage (e.g., disk) before acknowledging the commit.

For example, if a customer places an online order, durability ensures that the order remains recorded even if the database crashes immediately afterward. Techniques like write-ahead logging (WAL) and replication enhance durability by ensuring data is safely stored before confirming the transaction.

Why ACID Matters

ACID properties are crucial for applications requiring data integrity and reliability, such as:

- Banking systems (transfers must be atomic and durable).
- E-commerce (inventory updates must be consistent).
- Healthcare records (transactions must be isolated to prevent errors).

While ACID is essential for traditional relational databases (e.g., MySQL, PostgreSQL), some NoSQL systems (e.g., MongoDB, Cassandra) relax these properties for scalability, opting for BASE (Basically Available, Soft state, Eventual consistency) in distributed environments.

21. Normalization vs Denormalization

Normalization

Normalization is a systematic approach to organizing data in a relational database by breaking down large tables into smaller, related tables while minimizing redundancy. The primary goal is to structure data logically to reduce data duplication and improve data integrity. This is achieved by ensuring that each piece of data is stored in only one place, with relationships between tables established through foreign keys. Normalization follows a series of rules called normal forms (such as 1NF, 2NF, 3NF) that progressively refine the database structure to eliminate anomalies like insertion, update, and deletion inconsistencies. By decomposing tables into well-defined structures, normalization helps maintain accuracy and consistency while optimizing storage efficiency.

Denormalization

Denormalization is the intentional process of combining tables or adding redundant data to a database to improve read performance, often at the expense of increased storage and potential data inconsistency. Unlike normalization, which prioritizes data integrity and minimal redundancy, denormalization deliberately introduces duplication to speed up query execution, particularly in analytical or reporting systems where complex joins can slow down performance. This technique is commonly used in data warehouses, OLAP systems, and read-heavy applications where fast retrieval is more critical than storage efficiency. While denormalization reduces the need for joins and simplifies queries, it requires careful management to ensure data consistency, often through triggers or application logic

Category	Normalization	Denormalization
1. Definition	Process of structuring data to minimize redundancy	Process of intentionally introducing redundancy
2. Primary Purpose	Ensure data integrity and eliminate anomalies	Optimize query performance
3. Data Organization	Data split into multiple related tables	Data combined into fewer tables
4. Storage Efficiency	High (no duplicate data)	Low (contains redundant data)
5. Query Complexity	Complex queries with multiple joins	Simple queries with fewer joins
6. Read Performance	Slower (due to joins)	Faster (reduced join operations)
7. Write Performance	Faster (single-point updates)	Slower (multiple updates needed)
8. Data Consistency	High (ACID compliant)	Potential inconsistencies
9. Maintenance	Easier to maintain	Harder to maintain
10. Space Usage	Efficient	Inefficient
11. Best For	OLTP systems (transaction processing)	OLAP systems (analytics/reporting)
12. Update Anomalies	Prevents anomalies	May introduce anomalies
13. Join Operations	Frequent joins required	Minimal joins needed
14. Implementation	Follows normal forms (1NF-5NF)	No strict rules
15. Data Redundancy	Eliminated	Introduced

Category	Normalization	Denormalization
16. Example Systems	MySQL, PostgreSQL (transactional)	Data warehouses, BI tools
17. Scalability	Vertical scaling preferred	Horizontal scaling possible
18. Flexibility	More flexible for schema changes	Less flexible for modifications
19. Backup/Restore	Faster and smaller backups	Slower and larger backups
20. Typical Use Case	Banking systems, ERP software	Business intelligence, dashboards

Types of Normalisations - 1NF, 2NF, 3NF, BCNF

Different stages or levels of normalization are termed as 'Normal Forms.' The progression from one form to the next involves applying specific rules and making design modifications:

- 1NF (First Normal Form): Every column contains atomic, indivisible values, and each entry in the table has a unique identifier or primary key.
- 2NF (Second Normal Form): Meets all 1NF criteria and ensures that non-key attributes are functionally dependent on the entire primary key in case of composite primary keys.
- 3NF (Third Normal Form): Satisfies both 1NF and 2NF, and also ensures that non-key attributes are functionally dependent only on the primary key, not on other non-key attributes.
- BCNF (Boyce-Codd Normal Form): A more stringent form than 3NF, BCNF ensures that for every non-trivial functional dependency, the left-hand side is a superkey.

Imagine we have a table storing information about students, courses, and the grades students receive in those courses:

Initial Table:

```
| StudentID | StudentName | Course | Instructor | Grade |
| - - - - - -| - - - - - - -| - - - - - - - | - - - - - - - | - - - -|
| 101 | Alice | Math | Mr. Smith | A |
| 101 | Alice | History | Ms. Doe | B |
| 102 | Bob | Math | Mr. Smith | B |
```

1. 1NF (First Normal Form):

Ensure atomic values and have a unique identifier for each entry.

The table is already in 1NF because:

- Each column contains atomic (indivisible) values.
- The combination of StudentID and Course can serve as a unique identifier.

2. 2NF (Second Normal Form):

Remove partial dependencies of any column on the primary key.

We'll break down the table to remove columns that are dependent only on a part of the primary key:

```
Students Table

| StudentID | StudentName |
| - - - - - | - - - - - - |
| 101       | Alice       |
| 102       | Bob         |



Courses Table
| Course  | Instructor |
| - - - - | - - - - - -|
| Math    | Mr. Smith  |
| History | Ms. Doe    |


Grades Table
| StudentID  | Course  | Grade  |
| - - - - - -| - - - - | - - - -|
| 101        | Math    | A      |
| 101        | History | B      |
| 102        | Math    | B      |
```

3. 3NF (Third Normal Form):

Remove transitive dependencies of non-key attributes.

In our 2NF tables, there's no transitive dependency. If, however, our Courses Table included a column like "InstructorOffice," which is dependent on the Instructor (and not on the course), it would be a transitive dependency, and we would need to remove it to another table to achieve 3NF.

4. BCNF (Boyce-Codd Normal Form):

For every non-trivial functional dependency, the left-hand side is a superkey.

In the current design, our tables already meet BCNF. However, suppose there was a scenario where a course could be taught by multiple instructors. The course alone would no longer determine the instructor, and we would need further normalization to satisfy BCNF.These are simplistic examples, and actual normalization in a real-world scenario would consider many more attributes and complexities.

22. Data Warehouse Architecture

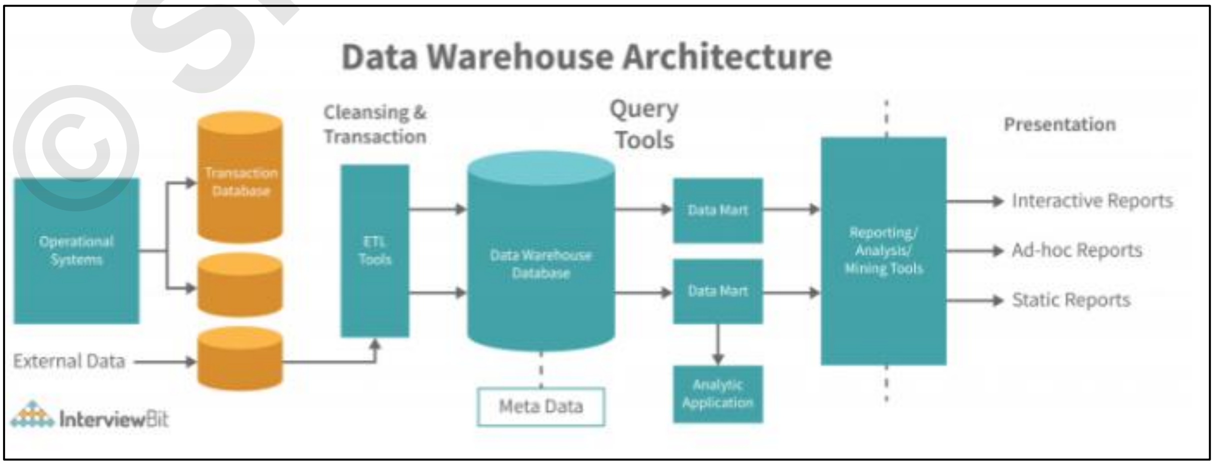
The single-tier Data Warehouse architecture is composed of a single hardware layer. This hardware layer is composed of a single hardware layer. There are three approaches to creating a data warehouse layer: Single-tier, two-tier, and three-tier.

Single-tier architecture: A single-layer structure aimed at keeping data space minimal. This structure is rarely used in real life.

Two-tier architecture: Data warehouse is the aggregation of data in a format that is easy to transform and load into a database. Data warehouses can be implemented in a number of different ways, and it is important to pick the right one for your business needs. The most important thing to consider is scalability. If you want to store large amounts of data in a small amount of space, then you should consider using a data warehouse.

Three-Tier Data Warehouse Architecture: The Top, Middle, and Bottom Tiers of this Architecture of Data Warehouse are collectively referred to as the Top Tier.

1. The bottom tier of the Datawarehouse is a relational database system. This database system typically contains a relational database system. Back-end tools clean, transform, and load data into this layer.
2. A middle tier OLAP server is either ROLAP or MOLAP-based. It abstracts OLAP from the end user by serving as a middle tier OLAP server. Data warehouses that facilitate end-user interaction with the database and middle tier OLAP servers that abstract OLAP from the end user are known as middle tier OLAP servers.
3. The front-end client layer of the top-tier is important because it is the first point of interaction with the data. It is where data is presented to the end user, and decisions are made with the data. The front-end client layer of top-tier must work with real-time data and must be able to process data quickly. It is also important to work with data that is in a format that top-tier can understand and use. Typically, top-tier data is in a relational database format, but it could be a file or a stream. Top-tier data must be well-structured, must be validated, and must be structured in a way that allows for easier data profiling and analytics.



Data Warehouse Architecture Properties

- We sometimes wish to keep analytical and transactional processing as far away as possible.
- The scalability of the solution should be demonstrated by the ability to process a huge volume of data and stream it to different destinations, at high speed, in various formats. The data stream should be processed and presented in the required format, at the right time and location, with the minimum impact to the existing infrastructure. The data stream must be protected and managed with the highest level of confidentiality and integrity. The size of the data stream and the rate at which the data is being generated must be determined by the business requirements, and the available hardware and software resources must be utilized to the fullest extent possible.
- The architecture should be extensible; new functionality can be implemented in an existing service by extending the service's APIs. For example, an insurance company could extend their customer service platform to provide a new feature that allows customers to obtain a personalized quote based on their preferences. Newer technologies, such as artificial intelligence, can be implemented in an existing service by extending the service's APIs. For example, an insurance company could extend their customer service platform to provide a new feature that allows customers to obtain a personalized quote based on their preferences. Newer technologies, such as artificial intelligence, should be implemented in the core services; the core services can be extended for new business functions, such as customer relationship management.
- Data security is a critical aspect of the data governance strategy. Data security controls at the source include establishing data access controls and data encryption. Data security controls at the perimeter include data security policies and monitoring access to the data.
- It should be simple and straightforward, and users should be able to work with the data in an efficient and effective manner. Data Warehouse management should be easy to understand and implement. Data Warehouse management should not be complicated and difficult for beginners should not find their way into data warehouse management. It should be simple to use and easy to understand.

23. Types of Data Warehouse Architectures

There are basically three different data warehouse architectures.

Single-Tier Architecture

Single-tier architectures are not implemented in real-time systems. They are used for batch and real-time processing. The data is first transferred to a single-tier architecture where it is converted into a format that is suitable for real-time processing. This architecture is known as "single-threaded". After this, the data is transferred to a real-time system. Single-tier architectures are currently the most preferred way to process operational data. It is important to note that single-tier architectures are not implemented in real time systems.

The data storage and processing middleware should be able to determine the quality of the data before the data is accepted by the analytical engine and transformed into relevant information. If these steps are not performed, then the middleware can be penetrated by malicious or faulty code. As an example, consider a credit score calculation. If a malicious hacker controls the middleware, then the hacker can modify the score and extract valuable data.

Two-Tier Architecture

In a two-tier data warehouse, an analytical process is separated from a business process. This allows for greater levels of control and efficiency. A two-tier system also provides a better understanding of the data and allows for more informed decisions.

Two-layer architecture describes a four-stage data flow in which physical sources are separated from data warehouses by a two-layered architecture.

- The source of the data is critical to the data warehouse's integrity. The integrity of the data stored in the data warehouse must be guaranteed. Data integrity is the degree to which data values in a database record are true or accurate. A data warehouse is a system that stores information in a database so that it can be searched and analyzed.
- Data staging is a key process in the ETL process, and one that can significantly reduce the time it takes to extract, transform, and load (ETL) a large data set. ETL tools can extract data from various storage sources, transform the data with corporate-specific functions, and load the data into a data warehouse. Data warehouse functions such as monitoring the system, provisioning new data, and making decisions on the basis of the data are all done through data warehouse functions such as ETL. Data warehouse functions such as ETL can be implemented through a data warehouse.
- Data warehouse metadata is a critical component of the data warehouse. It is the information that helps a data warehouse administrator decide which data to delete, which data to retain, and which data to use in future reports. It is also important to maintain data warehouse consistency. Data warehouse administrators must determine which data should be updated or deleted when new data arrives, and which data should be left untouched. When data warehouse consistency is not guaranteed, application developers and users must be careful about which tables and reports they create.

- Data profiling is also very important for this level as it helps in validating data integrity and presentation standards. It also comes with advanced analytics such as real-time and batch reporting, data profiling and visualizations, and rating functions. It is important to keep in mind that this is not just a data warehouse but a live data platform that receives and analyzes massive amounts of data. This is why it is important to keep track of data changes, scalability, and performance of the system.

Three-Tier Architecture

A three-tier structure is employed in the source layer, the reconciled layer, and the data warehouse layer. The reconciled layer sits between the source data and data warehouse. The main disadvantage of the reconciled layer is the fact that it is not possible to completely ignore the problems of the data before it is reconciled. Therefore, the main focus of the reconciler should be on data integrity, accuracy, and consistency. For example, assume that the data warehouse contains a collection of company data elements that are updated frequently, such as order book information. In such a case, the best approach would be to use a web-based data warehouse refresh tool, which extracts the latest data from the data warehouse and refreshes the data in the corporate application. This architecture is appropriate for systems with a long-life cycle. Whenever a change occurs in the data, an extra layer of data review and analysis is done to ensure that no erroneous data was entered. This architecture is also known as data-driven architecture. This structure is mainly used for large-scale systems. It is important to note that the extra layers of data review and analysis created by this structure does not consume any extra space in the storage device.

24. What are the advantages of a cloud-based data warehouse?

Total cost of ownership is low: The low cost of cloud data warehouses is one of the reasons they are becoming more popular. On-premises data warehouses necessitate high-cost technology, lengthy upgrades, ongoing maintenance, and outage management.

Increased performance and speed: To keep up with the expanding number of data sources, cloud data warehouses are crucial. Cloud data warehouses can easily and quickly integrate with additional data sources as needed, and deploy the updated solution to production. Cloud data warehouses significantly improve speed and performance, allowing IT to focus on more innovative projects.

Enhanced Security: Cloud security engineers can create and iterate on precise data-protection measures. Furthermore, cloud encryption technologies such as multi-factor authentication make data transfer between regions and resources extremely safe.

Improved Disaster Recovery: Physical assets are not required to prepare cloud data warehouses for disasters. Instead, almost all cloud data warehouses offer asynchronous data duplication and execute automatic snapshots and backups. This data is kept across multiple nodes, allowing duplicate data to be accessed at any time without stopping present activity.