Data Engineering Techniques (AID30010) — Unit 2

Data Warehousing

Table of Contents

1. Data Warehouse — Definition and Key Characteristics
    - Subject-Oriented
    - Integrated
    - Time-Variant
    - Nonvolatile
2. OLTP vs. OLAP
3. Data Warehouse Architecture (Three-Tier)
4. Data Warehouse Models
5. Multidimensional Data Model and Data Cube
6. Dimensional Modelling — Dimensions and Concept Hierarchies
7. Measures — Categorization and Computation
8. OLAP Operations on a Data Cube
9. Data Warehouse Design — Star, Snowflake, and Fact Constellation Schemas
10. Data Warehouse Design — Top-Down and Bottom-Up Approaches
11. Role of Metadata
12. Materialized Views
13. ETL Process
14. OLAP Server Architectures — ROLAP, MOLAP, HOLAP
15. Data Lakes
16. Data Warehouse vs. Data Lake
17. Data Lakehouses

---

1. Data Warehouse — Definition and Key Characteristics

1. A data warehouse is a single, complete, and consistent store of data obtained from a variety of different sources, made available to end users in a way they can understand and use in a business context.

2. It is a decision support database that is maintained separately from the organization's operational databases.
3. It supports information processing by providing a solid platform of consolidated, historical data for analysis.
4. W. H. Inmon's definition: "A data warehouse is a subject-oriented, integrated, time-variant, and nonvolatile collection of data in support of management's decision-making process."
5. The four key characteristics are: Subject-Oriented, Integrated, Time-Variant, and Nonvolatile.

Subject-Oriented

1. The data warehouse is organized around major subjects such as customer, product, and sales.
2. It focuses on modeling and analysis of data for decision makers — not on daily operations or transaction processing.
3. It provides a simple and concise view around particular subject issues.
4. It excludes data that is not useful in the decision support process.
5. Example: A customer subject area connects all customer-related tables by a common key — customer ID.

Integrated

1. The data warehouse is constructed by integrating multiple, heterogeneous data sources such as relational databases, flat files, and online transaction records.
2. Data cleaning and data integration techniques are applied before loading data into the warehouse.
3. It ensures consistency in naming conventions, encoding structures, and attribute measures among different data sources.
4. When data is moved to the warehouse, it is converted to a consistent form.
5. Example: Hotel price data from different sources may use different currencies and tax structures — all must be standardized.

Time-Variant

1. The time horizon for a data warehouse is significantly longer than for operational systems.
2. Operational databases hold current value data (60–90 day time horizon), while a data warehouse provides information from a historical perspective — e.g., past 5–10 years.
3. Every key structure in the data warehouse contains an element of time, explicitly or implicitly.
4. As a result, the data warehouse contains much more history than any other environment.

5. This historical perspective is what makes it useful for trend analysis and decision making.

Nonvolatile

1. A data warehouse is a physically separate store of data transformed from the operational environment.
2. Operational update of data does not occur in the data warehouse environment.
3. It does not require transaction processing, recovery, and concurrency control mechanisms.
4. It requires only two operations in data accessing: initial loading of data and access of data.
5. Data is loaded in a snapshot, static format — when changes occur, a new snapshot record is written, so history is maintained.

---

2. OLTP vs. OLAP

What is OLTP?

1. OLTP stands for On-Line Transaction Processing.
2. It handles day-to-day operations such as purchasing, inventory, banking, manufacturing, payroll, registration, and accounting.
3. It is oriented toward clerks and IT professionals.
4. It uses an ER model and application-oriented database design.
5. Data is current, detailed, and flat relational.

What is OLAP?

1. OLAP stands for On-Line Analytical Processing.
2. It is used for data analysis and decision making.
3. It is oriented toward knowledge workers and managers.
4. It uses a star schema and subject-oriented database design.
5. Data is historical, summarized, multidimensional, and integrated.

Comparison Table

Feature	OLTP	OLAP
Users	Clerks, IT professionals	Knowledge workers, managers
Function	Day-to-day operations	Decision support
DB Design	ER model + application-oriented	Star schema + subject-oriented

Feature	OLTP	OLAP
Data	Current, detailed, flat relational	Historical, summarized, multidimensional
Usage	Repetitive transactions	Ad-hoc complex queries
Access	Read/Write	Mostly read-only, complex queries
Records Accessed	Tens (one at a time)	Millions (in bulk)
DB Size	100 MB – GB	100 GB – TB

Why a Separate Data Warehouse?

1. High performance for both systems — the DBMS is tuned for OLTP transactions while the warehouse is tuned for OLAP analysis.
2. Decision support requires historical data spanning years, which operational databases do not typically maintain.
3. Decision support requires aggregation and summarization from heterogeneous sources that OLTP systems cannot easily provide.
4. Different sources use inconsistent representations, codes, and formats which must be reconciled before analysis.
5. Mixing analytical queries with operational queries degrades performance for both — a separate warehouse avoids this conflict.

---

3. Data Warehouse Architecture (Three-Tier)

1. A data warehouse uses a three-tier architecture to manage data from multiple operational sources and deliver it to end users.
2. The three tiers are: Data Sources (bottom), Data Storage / Warehouse Server (middle), and Front-End Tools (top).
3. Data flows from source systems through ETL processes to the warehouse, and then to end users.
4. This architecture separates storage, processing, and presentation for better performance and flexibility.
5. A metadata repository is also part of the architecture, storing information about the structure and content of the warehouse.

Tier 1 — Data Sources

1. The bottom layer contains operational databases and external data.

2. Sources include relational databases, flat files, online transaction records, and external data.
3. This is where raw data originates before it is processed.
4. Data from these sources may be inconsistent, incomplete, or formatted differently.
5. All this data is extracted, cleaned, and loaded into the warehouse.

Tier 2 — Data Storage (Warehouse Server)

1. The middle layer is where actual data is stored and processed.
2. Data is extracted, cleaned, transformed, and loaded (ETL process) from source systems.
3. An OLAP Server processes analytical queries on warehouse data.
4. This layer also contains Data Marts — smaller, subject-specific subsets of the warehouse.
5. Pre-computed aggregations and summaries are stored here for fast query performance.

Tier 3 — Front-End Tools

1. The top layer is what end users interact with.
2. Tools include Analysis, Query & Reporting, and Data Mining tools.
3. It allows users to perform OLAP operations, generate reports, and discover patterns.
4. Users do not directly access raw data — they work through this layer.
5. Examples include dashboards, business intelligence tools, and visualization platforms.

---

4. Data Warehouse Models

Enterprise Warehouse

1. Collects all of the information about subjects spanning the entire organization.
2. Provides a comprehensive, organization-wide view of data.
3. Serves as the central warehouse for all business areas.
4. It is large in scale and requires significant planning and resources to build.
5. All data marts in an organization can be built from the enterprise warehouse.

Data Mart

1. A data mart is a subset of corporate-wide data that is of value to a specific group of users.
2. Its scope is confined to specific, selected groups — e.g., a marketing data mart or a finance data mart.
3. **Independent Data Mart:** Built directly from source data, without a central enterprise warehouse.

4. **Dependent Data Mart:** Built directly from an enterprise data warehouse — more consistent and reliable.
5. Data marts are smaller and faster to build than full enterprise warehouses.

Virtual Warehouse

1. A virtual warehouse is a set of views over operational databases.
2. Only some of the possible summary views may be materialized.
3. It is easy to build but requires excess capacity on operational database servers.
4. It does not physically store data — it provides a virtual view of existing data.
5. Performance can be a concern since queries still run against operational databases.

Cloud-Based Data Warehouse

1. Hosted on cloud infrastructure (e.g., AWS Redshift, Google BigQuery, Snowflake).
2. Can typically perform complex analytical queries much faster because they use Massively Parallel Processing (MPP).
3. Highly scalable, cost-effective, and easily accessible.
4. No need to manage physical hardware — the cloud provider handles infrastructure.
5. Suitable for organizations with large and growing data needs.

---

5. Multidimensional Data Model and Data Cube

Multidimensional Data Model

1. A data warehouse is based on a multidimensional data model which views data in the form of a data cube.
2. Managers think of business in terms of business dimensions.
3. Example: A marketing VP is interested in revenue broken down by month, division, customer, product version, sales office, etc.
4. These breakdowns are the business dimensions.
5. The multidimensional model makes it easy to analyze data from multiple perspectives at the same time.

What is a Data Cube?

1. A data cube allows data to be modeled and viewed in multiple dimensions.
2. Example: Sales can be viewed across the dimensions of Product, Location, and Time.

3. Each combination of dimension values defines a cell in the cube that contains a measure (e.g., dollars sold).
4. A data cube can have more than three dimensions — it is a general term for multi-dimensional data.
5. It allows fast access to summarized data at different levels of detail.

Dimension Tables and Fact Tables

1. **Dimension tables** contain descriptive attributes about the dimensions (e.g., item name, brand, type).
2. **Fact tables** contain measures (such as dollars sold) and foreign keys to each dimension table.
3. The fact table is at the center and is connected to all dimension tables.
4. Time, Product, and Location are common examples of dimensions.
5. The combination of the fact table and dimension tables forms the basis of the data warehouse model.

Cuboids and the Lattice

1. For n dimensions, an n-D base cube is called the **base cuboid** — it holds the most detailed data.
2. The topmost 0-D cuboid (highest level of summarization) is called the **apex cuboid**.
3. The lattice of cuboids (from base to apex) forms the full data cube.
4. Each cuboid represents data summarized at a different combination of dimensions.
5. **Formula for number of cuboids:** Total cuboids = $\prod(Li + 1)$ for i = 1 to n (where Li = number of levels in dimension i).

Example for {product, date, country}:

- 0-D (apex): all
- 1-D: {product}, {date}, {country}
- 2-D: {product, date}, {product, country}, {date, country}
- 3-D (base): {product, date, country}

---

6. Dimensional Modelling — Dimensions and Concept Hierarchies

What are Dimensions?

1. Dimensions are the perspectives or entities with respect to which an organization wants to keep records.

2. They define the context for the measures (facts) stored in the fact table.
3. Example: A sales data cube may have dimensions — time, item, branch, and location.
4. Different roles in an organization focus on different dimensions based on their responsibilities.
5. Dimensions help managers slice, filter, and analyze data from the angle that matters most to them.

Business Dimensions — Examples

1. **Marketing VP:** Revenue broken down by month, division, customer, demographic, sales office, product version.
2. **Marketing Manager:** Product, product category, time (day/week/month), sale district, distribution channel.
3. **Financial Controller:** Budget line, time (month/quarter/year), district, division.
4. **Insurance Claim Analysis:** Time, agent, claim, insured party, policy, status.
5. **Airline Flyer Analysis:** Time, customer, flight, fare class, airport, status.

What are Concept Hierarchies?

1. A concept hierarchy defines a sequence of mappings from a set of lower-level concepts to higher-level, more general concepts.
2. They allow data to be viewed at different levels of abstraction.
3. Used for roll-up and drill-down operations in OLAP.
4. The same data can have different levels of granularity (detail) depending on the hierarchy level.
5. Concept hierarchies are built into dimension tables and are essential for meaningful data analysis.

Examples of Concept Hierarchies

Location Hierarchy: office → city → country → region → all

- Example: L. Chan's office → Vancouver → Canada → North America → all locations

Time Hierarchy: day → week → month → quarter → year → all

- Example: A specific date → the week it belongs to → the month → Q1/Q2/Q3/Q4 → the full year

Product Hierarchy: product → category → industry → all

- Example: iPhone 15 → Smartphones → Electronics → all products

Hierarchical Summarization Paths

1. **Product path:** Industry → Category → Product — a manager can start from a broad industry view and drill down to specific individual products.
2. **Location path:** Region → Country → City → Office — analysis can start at a global region level and drill down all the way to a specific office location.
3. **Time path:** Year → Quarter → Month → Week → Day — reports can be generated at any time granularity from annual summaries down to daily records.
4. These paths allow managers to summarize and drill through data at any desired level of detail depending on what question they are trying to answer.
5. The ability to navigate these paths fluidly is what makes OLAP powerful — users can zoom in or zoom out on the data as needed.

---

7. Measures — Categorization and Computation

What are Measures?

1. Measures are the numerical values stored in the fact table of a data warehouse.
2. They represent what is being tracked or analyzed (e.g., sales amount, count of orders).
3. Measures are categorized based on how they can be computed across aggregate levels.
4. Knowing the category of a measure helps decide which ones can be efficiently pre-computed.
5. There are three categories: Distributive, Algebraic, and Holistic.

Distributive Measures

1. A function is distributive if the result derived by applying it to n aggregate values is the same as applying it on all data without partitioning.
2. In other words, they can be computed by dividing data into partitions and combining partial results.
3. Examples: count(), sum(), min(), max().
4. These are the simplest and most efficient to compute in a data cube.
5. Distributive measures can always be pre-computed and materialized for all cuboids efficiently.

Algebraic Measures

1. Algebraic measures can be computed by applying an algebraic function with M arguments, each obtained by applying a distributive aggregate function.

2. They require some additional information beyond just the partial result.
3. Examples: avg(), min_N(), standard_deviation().
4. avg() = sum() / count() — both sum and count are distributive, so avg is algebraic.
5. Algebraic measures can also be pre-computed efficiently when the supporting distributive functions are stored.

Holistic Measures

1. There is no constant bound on the storage size needed to describe a subaggregate for holistic measures.
2. They cannot be computed efficiently by combining partial results.
3. Examples: median(), mode(), rank().
4. These are the most expensive to compute in a data cube.
5. Holistic measures are typically computed only when needed rather than pre-computed for all cuboids.

---

8. OLAP Operations on a Data Cube

1. OLAP (On-Line Analytical Processing) is used for complex analysis and decision-making on data warehouse data.
2. OLAP operations allow users to view the data cube from different perspectives and at different levels of detail.
3. The main OLAP operations are: Roll-Up, Drill-Down, Slice, Dice, and Pivot.
4. These operations can be combined to answer complex business questions.
5. OLAP queries typically access millions of records and return summarized results.

Roll Up (Drill Up)

1. Roll-up summarizes data by climbing up the hierarchy or by dimension reduction.
2. It moves from more detailed data to more summarized data.
3. Example: Rolling up from city → country → region in the location dimension.
4. Example: Rolling up from monthly sales to quarterly sales to annual sales.
5. Also possible by dimension reduction — dropping one dimension entirely.

Drill Down (Roll Down)

1. Drill-down is the reverse of roll-up — it moves from higher-level summary to lower-level detail.
2. It introduces more detail or adds new dimensions.

3. Example: Drilling down from annual sales to quarterly to monthly sales.
4. Example: Drilling down from country → province → city.
5. Drill-down is used when a manager wants to investigate the cause behind a high-level trend.

Slice

1. Slice performs a selection on one dimension of the data cube, resulting in a sub-cube.
2. It fixes one dimension to a specific value and shows the remaining dimensions.
3. Example: Slice the cube where time = "Q1 2024" — shows sales for all products and locations in Q1.
4. The result is a smaller cube with one fewer dimension.
5. Useful for focusing analysis on a specific time period, region, or product.

Dice

1. Dice defines a sub-cube by specifying a range or set of values across two or more dimensions.
2. It is like multiple slices at once — multiple dimensions are filtered simultaneously.
3. Example: Dice where (time = "Q1" or "Q2") AND (location = "Canada" or "USA") AND (item = "TV" or "PC").
4. The result is a smaller sub-cube containing only the selected data.
5. Useful for analyzing data within a specific combination of conditions.

Pivot (Rotate)

1. Pivot reorients the cube for visualization.
2. It rotates the data axes to provide an alternative presentation of the data.
3. Example: Converting a 3D cube into a series of 2D planes for easier viewing.
4. Pivot does not change the data — it only changes the way it is displayed.
5. Useful for presenting data in a different layout to spot patterns more easily.

Other Operations

1. **Drill Across** — Involves executing queries that span across more than one fact table in the data warehouse to compare related measures.
2. **Drill Through** — Goes through the bottom level of the cube all the way to its back-end relational tables using SQL, retrieving the raw, most detailed records from the warehouse.
3. Both drill across and drill through are used when the summarized cube data is not enough and you need to go deeper into the underlying raw data.

4. Drill across is useful when analyzing related business processes stored in separate fact tables, such as comparing sales and returns together.
5. Drill through is useful when an analyst spots an unusual number in a report and wants to see the exact individual transactions that caused it.

---

9. Data Warehouse Design — Star, Snowflake, and Fact Constellation Schemas

Why Not Use ER Model for Data Warehouses?

1. End users cannot understand or remember a complex ER model.
2. Many data warehouses have failed because of overly complex ER designs.
3. ER models are not optimized for complex, ad-hoc queries.
4. Data retrieval becomes difficult due to normalization.
5. Browsing and analysis become difficult with a normalized ER model.

Star Schema

1. Has a single fact table in the middle connected to a set of dimension tables — resembling a star shape.
2. Every fact record points to one tuple in each of the dimension tables and contains measure attributes.
3. Does not capture hierarchies directly — dimension tables are flat (denormalized) and contain all levels in one table.
4. System-generated (surrogate) keys are used instead of natural keys for better performance and easier maintenance.
5. It is the most widely used schema in data warehousing because of its simplicity and fast query performance.

Advantages:

- Simple design — the single flat structure makes it very easy for users to understand and navigate.
- Fast read performance because fewer joins are needed to answer analytical queries.
- Easy data aggregation — rollups and summaries can be computed quickly across the denormalized dimensions.
- Easy integration with OLAP systems and data cubes since the structure directly maps to multidimensional views.

Disadvantages:

- Redundant data across dimension tables makes for larger storage requirements on disk.
- Potential for data anomalies, errors, and inconsistencies since dimension tables are not normalized.
- Limited flexibility for non-dimensional data or complex hierarchical relationships that need further normalization.

Snowflake Schema

1. Similar to the star schema but each dimension table may have foreign keys relating to other dimension tables — dimension tables are normalized.
2. A refinement of the star schema where some dimensional hierarchy is normalized into smaller dimension tables — forming a snowflake shape.
3. More normalized than the star schema — reduces data redundancy by splitting large dimension tables into multiple related smaller tables.
4. Used to manage the size of dimension tables and ensure each piece of information is stored in only one place.
5. The name comes from the shape it creates — the central fact table branches out into dimension tables, which further branch into sub-dimension tables, like a snowflake.

Advantages:

- Saves memory because normalized dimension tables eliminate repeated data across records.
- Normalized structures are easier to update and maintain since each piece of information is stored only once.

Disadvantages:

- Increases the complexity of the schema since dimension tables are split into multiple related tables.
- Browsing becomes more difficult for users who need to understand multiple joined tables to answer queries.
- Degrades query performance because additional joins are required to connect the normalized dimension tables.

Fact Constellation Schema (Galaxy Schema)

1. Multiple fact tables share dimension tables — unlike the star and snowflake schemas which each have only one fact table.
2. Used when applications are complex and require multiple fact tables to model different business processes.

3. Viewed as a collection of stars sharing common dimension tables — hence also called Galaxy Schema.
4. Example: A Sales Fact Table and a Shipping Fact Table both share the Time, Item, and Location dimension tables.
5. This schema is used in complex data warehouses where different departments need to analyze related but distinct processes together.

Advantages:

- Offers more flexibility since multiple business processes can be modeled together in a single schema.
- Multiple fact tables are explicitly assigned to shared dimension tables, enabling cross-process analysis.

Disadvantages:

- Structure is more complex and sophisticated, making it harder for non-technical users to understand.
- Difficult to maintain because changes to a shared dimension table can affect multiple fact tables at once.
- High number of aggregations may be required when querying across multiple fact tables simultaneously.

Summary Comparison

Schema	Fact Tables	Dimension Tables	Normalization
Star	One	Denormalized (flat)	Low
Snowflake	One	Normalized (multiple levels)	High
Fact Constellation	Multiple	Shared across fact tables	Medium

---

10. Data Warehouse Design — Top-Down and Bottom-Up Approaches

Four Views for Data Warehouse Design

1. **Top-down view** — Allows selection of the relevant information necessary for the data warehouse based on business needs.
2. **Data source view** — Exposes the information being captured, stored, and managed by operational systems that will feed the warehouse.

3. **Data warehouse view** — Consists of fact tables and dimension tables that represent the core structure of the warehouse.
4. **Business query view** — Sees the perspectives of data from the end-user's point of view and defines the types of questions they need to answer.
5. All four views must be considered together during design to ensure the warehouse is technically sound, user-friendly, and aligned with business goals.

Top-Down Approach

1. Starts with the overall design and planning of the entire enterprise warehouse before any data mart is built.
2. A mature, comprehensive approach — the whole enterprise warehouse is designed first, then data marts are built from it as needed.
3. Follows the Waterfall method: structured and systematic analysis at each step before proceeding to the next phase.
4. All data marts are derived from the central enterprise warehouse, ensuring consistency across the entire organization.
5. Best suited for organizations that have the time, budget, and expertise to invest in a large, long-term data infrastructure project.

Advantages:

- Easier to maintain since all data marts are built from a single, consistent enterprise warehouse.
- Provides consistent dimensional views of data across all data marts, so every team works with the same definitions and numbers.
- Robust against business changes — when a new data mart is needed, it can simply be built from the existing warehouse without redesigning from scratch.
- Initial cost is high but subsequent data mart development becomes cheaper since the foundation is already in place.

Disadvantages:

- Represents a very large project — the cost of initial implementation is significant and requires heavy investment upfront.
- Very time consuming — the entire enterprise warehouse must be designed and built before any data mart can go live.
- Requires highly skilled people with deep expertise in data architecture, business processes, and data modeling.

Bottom-Up Approach

1. Starts with experiments and prototypes — builds individual data marts first, then gradually integrates them into an enterprise warehouse.
2. Follows the Spiral method: rapid generation of increasingly functional systems with quick turnaround times at each stage.
3. Each data mart is built to serve a specific department or business function, such as sales, finance, or marketing.
4. The enterprise warehouse emerges over time as more and more data marts are built and connected together.
5. Best suited for organizations that need quick results, have limited initial budgets, or are new to data warehousing and want to start small.

Advantages:

- Data marts can be delivered quickly, giving business teams access to useful reports much sooner than the top-down approach.
- Data marts are created first to provide immediate reporting and analysis capability to specific departments.
- Less time required — the initial setup is very quick since you only build one small data mart at a time.
- Easier to extend — when a new business area needs coverage, a new data mart is built and connected to the growing system.

Disadvantages:

- Initial cost is low but each subsequent phase of building a new data mart costs roughly the same amount, so total cost adds up.
- Difficult to maintain over time because data marts built independently may use different definitions, formats, or business rules.
- Often leads to redundant data and inconsistencies across data marts, since they were not designed from a single unified blueprint.

Typical Data Warehouse Design Steps

1. Choose a business process to model — for example, orders, invoices, claims, or student grades.
2. Choose the grain (atomic level of data) of the business process — this is the lowest level of detail the warehouse will store, such as one row per individual transaction.
3. Choose the dimensions that will apply to each fact table record — these are the perspectives for analysis, such as time, product, and location.

4. Choose the measures that will populate each fact table record — these are the numeric values being tracked, such as sales amount or quantity sold.
5. Once the grain, dimensions, and measures are defined, the schema can be built and data loading can begin.

---

11. Role of Metadata

What is Metadata?

1. Metadata in a data warehouse is data that describes and contextualizes other data.
2. It provides information about the content, format, structure, and other characteristics of data.
3. It is considered the backbone of effective data management.
4. Often called "data about data."
5. Metadata makes the data warehouse understandable and governable.

What Metadata Stores

1. **Structure of the data warehouse** — schema, view, dimensions, hierarchies, derived data definitions, data mart locations and contents.
2. **Operational metadata** — data lineage (history of migrated data and transformation path), currency of data (active, archived, or purged), monitoring information (usage statistics, error reports, audit trails).
3. **Algorithms used for summarization** — records what methods were used to compute summaries and aggregations stored in the warehouse.
4. **Mapping from operational environment to the data warehouse** — describes how each field in the warehouse relates back to its original source field in the operational system.
5. **Business data** — business terms and definitions, ownership of data, charging policies so that end users understand what each piece of data means in a business context.

Roles of Metadata

Data Understanding:

1. Helps users understand the data stored within the system so they know what each field means and how it was created.
2. Structural metadata describes how different elements of a compound data object are assembled and related to each other.
3. Example: page layout of a document, chapter structure of an ebook, or column definitions in a database table.

4. Without proper metadata, users may misinterpret data or use fields incorrectly in their analysis.
5. Good metadata documentation reduces the time analysts spend figuring out what data means and where it came from.

Data Governance:

1. Helps ensure data accuracy and compliance with organizational policies and legal regulations.
2. Enforces the definition of business terms so that every team uses the same definitions and standards.
3. Ensures that business rules and policies are applied consistently across the entire data warehouse.
4. Tracks ownership of data — metadata records which team or person is responsible for each dataset.
5. Supports auditing and regulatory compliance by keeping a clear record of how data is managed and used.

Data Integration:

1. Helps validate data transformation steps and ensures the accuracy of calculations during the ETL process.
2. Tracks exactly how data moved from operational systems to the warehouse, including every transformation applied.
3. Makes it possible to trace back any value in the warehouse to its original source — this is called data lineage.
4. If an error is found in the warehouse, metadata helps identify at which integration step the error was introduced.
5. Ensures that data from different sources has been correctly combined and no critical information was lost.

Data Accessibility:

1. Helps improve the accessibility of data by making it easy for users to find what they need.
2. A data catalog can combine metadata with data management and search tools to help analysts quickly find the right dataset.
3. Metadata acts like an index — it tells users what data exists, where it is stored, and how to access it.
4. Without accessible metadata, analysts waste significant time searching for data or working with the wrong datasets.

5. Good data accessibility through metadata increases the overall productivity of analytics and data science teams.

---

12. Materialized Views

What is Materialization?

1. A data cube can be viewed as a lattice of cuboids — from the base cuboid (most detailed) to the apex cuboid (most summarized).
2. Materialization refers to pre-computing and storing some or all of the cuboids for faster query processing.
3. When a query is received, the system checks if a materialized view already contains the result.
4. This significantly speeds up repeated or common analytical queries.
5. Materialized views must be refreshed periodically to reflect changes in source data.

Types of Materialization

1. **Full Materialization** — Every possible cuboid in the lattice is pre-computed and stored; this gives the fastest query response time but requires the most storage space.
2. **No Materialization** — No cuboids are pre-computed at all; storage is minimized but every query must be computed from scratch, making responses the slowest.
3. **Partial Materialization** — Only selected cuboids that are frequently accessed are pre-computed and stored; this is the most practical approach as it balances storage cost with query performance.

Selection of Which Cuboids to Materialize

1. Based on the **size** of the cuboid — smaller cuboids are cheaper to store and faster to compute, so they are better candidates for materialization.
2. Based on **sharing** — cuboids that can be reused across multiple different queries are more valuable to materialize since they serve more use cases at once.
3. Based on **access frequency** — cuboids that are queried most often by users should be materialized first to give the greatest performance benefit.
4. Always use the most specific materialized cuboid that satisfies the query to minimize unnecessary computation and retrieval time.
5. Distributive and algebraic measures can be easily pre-computed for all cuboids; holistic measures like median and rank are expensive and are typically computed on demand only.

Formula for Number of Cuboids

Formula: Total cuboids T = ∏(Li + 1) for i = 1 to n
Where n = number of dimensions, Li = number of levels in dimension i.

Example: 4 dimensions each with 4 levels → T = (4+1)^4 = 625 cuboids.

---

13. ETL Process

What is ETL?

1. ETL stands for Extract, Transform, Load — the process of moving data from source systems into the data warehouse.
2. It is the backbone of how data gets into the warehouse from operational systems.
3. ETL ensures that data is clean, consistent, and properly formatted before it is stored.
4. It is executed on a regular schedule (daily, weekly, etc.) to keep the warehouse up to date.
5. The quality of ETL directly affects the quality of data in the warehouse.

Extract

1. Data is extracted from multiple, heterogeneous, and external sources.
2. Sources may include relational databases, flat files, APIs, and external data.
3. Only relevant data is extracted based on the warehouse requirements.
4. This step may involve reading large volumes of data from different systems.
5. The extracted data is stored in a staging area before transformation.

Transform

1. Data is cleaned — errors are detected and corrected when possible.
2. Data is converted from legacy or source formats to the warehouse format.
3. Transformations include sorting, aggregating, joining, and calculating new fields.
4. Data is normalized and standardized to ensure consistency.
5. Business rules are applied during transformation to calculate derived values.

Load

1. The transformed data is loaded into the data warehouse.
2. Loading may involve sorting, summarizing, consolidating, computing views, and checking integrity.
3. Indices and partitions are built during or after loading for query performance.
4. Initial loading populates the warehouse for the first time; subsequent loads are incremental (refresh).

5. Refresh propagates updates from data sources to the warehouse to keep it current.

---

14. OLAP Server Architectures — ROLAP, MOLAP, HOLAP

ROLAP — Relational OLAP

1. Uses relational or extended-relational DBMS to store and manage warehouse data.
2. OLAP middleware is added on top of the relational database to provide multidimensional analysis capabilities.
3. Supports RDBMS products through a metadata layer — avoids the need to create a static multidimensional data structure.
4. Facilitates the creation of multiple multidimensional views of 2D relations without physically restructuring the data.
5. Best suited for very large datasets where scalability is the top priority and query speed is a secondary concern.

Advantage: Greater scalability — ROLAP can handle very large data volumes because it leverages the full power of relational databases that are designed to scale.

Disadvantages:

- Performance problems can arise with complex queries that require multiple passes through relational data, making it slower than MOLAP.
- Requires development of middleware to support multidimensional applications on top of the relational database.
- Slower for complex analytical queries compared to MOLAP because data is not pre-computed in multidimensional arrays.

MOLAP — Multidimensional OLAP

1. Uses specialized data structures and multidimensional database management systems (MDDBMS) to organize, navigate, and analyze data.
2. Uses a sparse array-based multidimensional storage engine that minimizes disk space through sparse data management.
3. Fast indexing to pre-computed summarized data allows analytical queries to return results almost instantly.
4. All aggregations are pre-computed and stored in the multidimensional structure before queries are even asked.
5. Best suited for smaller datasets where query speed is the highest priority and the analysis requirements are well-defined in advance.

Advantage: Very fast query performance because all aggregations are pre-computed and stored in multidimensional arrays, so responses are near-instant.

Disadvantages:

- Only a limited amount of data can be efficiently stored and analyzed, making MOLAP unsuitable for very large datasets.
- Navigation and analysis are limited because the data structure is designed according to pre-determined requirements that cannot easily change.
- Requires a different set of specialized skills and tools to build and maintain the multidimensional database, unlike standard SQL-based tools.

HOLAP — Hybrid OLAP

1. Combines features of both ROLAP and MOLAP to get the benefits of both architectures.
2. Provides limited analysis capability either directly against RDBMS products or via an intermediate MOLAP server.
3. Low-level, detailed data is stored relationally; high-level aggregations are stored as multidimensional arrays.
4. Example: Microsoft SQL Server uses a HOLAP architecture for its Analysis Services.
5. Best suited when an organization needs to support both large-volume detailed queries and fast high-level summary queries on the same platform.

Advantage: Flexibility — HOLAP combines the scalability of ROLAP for detailed low-level data with the speed of MOLAP for high-level aggregations, giving the best of both worlds.

Disadvantages:

- Significant data redundancy can occur since some data is stored both relationally and as arrays, which may cause problems for large multi-user networks.
- Each user building a custom data cube may cause a lack of data consistency, as different users may see slightly different versions of the same aggregated data.
- Only a limited amount of data can be efficiently maintained in the MOLAP component, so very large datasets may still face performance limitations.

Summary Comparison

Feature	ROLAP	MOLAP	HOLAP
Storage	Relational DB	Multidimensional arrays	Both
Performance	Moderate	Fast	Moderate-Fast
Scalability	High	Limited	Moderate

Feature	ROLAP	MOLAP	HOLAP
Flexibility	High	Low	High

---

15. Data Lakes

What is a Data Lake?

1. A Data Lake is a centralized repository that stores raw data in its original format at any scale.
2. Think of it like a big lake where all kinds of data flow in — you decide later how to process and analyze it.
3. It stores all three types of data: structured (tables, CSV), semi-structured (JSON, XML, logs), and unstructured (images, videos, audio, PDFs, text).
4. Data in a data lake is stored as-is, without any upfront processing or transformation.
5. It is designed for flexibility — analysts and data scientists can access raw data and process it as needed.

Problems with Data Lakes

1. No ACID transactions — data consistency cannot be guaranteed, so multiple users writing at the same time can cause corrupt or incomplete data.
2. Data quality issues — since raw, unprocessed data is stored without validation, the lake may contain unreliable, duplicated, or incorrect records.
3. Can become a "data swamp" — without proper governance, the lake fills up with too much unorganized, undocumented data that nobody can find or trust.
4. Limited support for traditional BI tools that expect clean, structured data — most reporting tools cannot work directly with raw unstructured data in a lake.
5. Schema is only defined when data is read (schema-on-read), which can lead to misinterpretation errors if different analysts apply different schemas to the same data.

---

16. Data Warehouse vs. Data Lake

Feature	Data Lake	Data Warehouse
Data Type	All (structured, semi, unstructured)	Structured only
Schema	On-read (schema defined at query time)	On-write (schema defined before loading)

Feature	Data Lake	Data Warehouse
ACID Transactions	❌ No	✅ Yes
Cost	Low	High
BI Support	Limited	Excellent
ML Support	Excellent	Limited

Key Differences:

1. A data warehouse stores only structured, processed data; a data lake stores raw data in any format.
2. Data warehouses define the schema before data is loaded (schema-on-write); data lakes define schema when data is read (schema-on-read).
3. Data warehouses are optimized for BI and reporting; data lakes are optimized for big data processing and machine learning.
4. Data warehouses have strict data quality controls; data lakes accept all data as-is.
5. Data warehouses are more expensive to maintain; data lakes are cheaper to store data in.

---

17. Data Lakehouses

What is a Data Lakehouse?

1. A Data Lakehouse is a hybrid data architecture that combines the flexibility of a Data Lake with the performance, reliability, and structure of a Data Warehouse.
2. Formula: Data Lake + Data Warehouse = Data Lakehouse.
3. Solves the limitations of both Data Lakes (poor data quality) and Data Warehouses (expensive, rigid schema).
4. It allows both BI (Business Intelligence) and ML (Machine Learning) workloads on the same platform.
5. Examples of Data Lakehouse formats: Delta Lake, Apache Iceberg, Apache Hudi.

Key Features of Data Lakehouse

1. **ACID transactions** — ensures data consistency and prevents data corruption even when multiple users read and write simultaneously, just like a traditional data warehouse.
2. **Schema enforcement and evolution** — data must conform to a defined schema like a warehouse, but the schema can also evolve over time without breaking existing data, giving

more flexibility than a rigid warehouse.

3. **Time travel (data versioning)** — users can query data as it existed at any past point in time, making it easy to audit changes, debug pipelines, or reproduce historical results.
4. **High query performance** — query performance is comparable to a traditional data warehouse because of optimized formats, indexing, and pre-computed metadata.
5. **Single source of truth** — both BI reporting and machine learning workflows can run on the same platform without needing to duplicate or move data between separate systems.

Data Lakehouse Architecture (Layered)

1. **Storage Layer** — Raw data is stored in cloud object storage (e.g., S3, ADLS, GCS) using open columnar formats like Parquet and ORC that are efficient for large-scale analytical reads.
2. **Metadata & Table Format Layer** — Tools like Delta Lake, Apache Iceberg, and Apache Hudi sit on top of the storage layer and provide ACID transactions, schema management, and time travel capabilities.
3. **Processing Layer** — Data is processed using distributed engines like Apache Spark, Flink, Trino, or Presto, which can handle large-scale transformations, aggregations, and queries efficiently.
4. **Analytics & BI Layer** — Business intelligence tools like Power BI, Tableau, and Looker connect to the lakehouse and allow non-technical users to create reports, dashboards, and visualizations.
5. **ML & AI Layer** — Machine learning frameworks like MLflow, TensorFlow, and PyTorch access the same data for model training and experimentation; a Governance & Security layer manages access control, versioning, and auditing across all layers.

Full Comparison — Data Lake vs. Data Warehouse vs. Data Lakehouse

Feature	Data Lake	Data Warehouse	Data Lakehouse
Data Type	All (structured, semi, unstructured)	Structured only	All
Schema	On-read	On-write	Flexible (both)
ACID Transactions	❌ No	✅ Yes	✅ Yes
Cost	Low	High	Medium
BI Support	Limited	Excellent	Excellent
ML Support	Excellent	Limited	Excellent