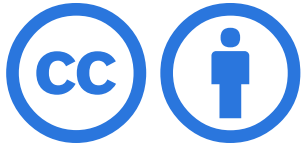
université
PARIS-SACLAY

DATA WAREHOUSE I

WEEK 1



INTRODUCTION

COURSE OVERVIEW

Course Objectives

- Understand the fundamentals of data warehousing
- Learn about data warehouse architecture and design
- Explore ETL processes and data modeling
- Gain hands-on experience with open-source DW tools

COURSE OVERVIEW

Course Structure

- 7 weeks of lectures and labs
- Weekly quizzes and in-class exercises
- Final exam in Week 8

IMPORTANCE OF DATA WAREHOUSING

- Enabling informed decision-making

Speaker notes

Data Warehousing allows businesses to consolidate their data from various sources into a single, consistent store. This not only ensures higher data quality and consistency but also enables historical analysis and supports complex queries that operational databases are not optimized for. This is critical for businesses that rely on deep analytics to drive decision-making, offering a consolidated view of performance over time. For instance, sales trends over multiple years or customer behavior analysis are best handled through a data warehouse, which can efficiently manage and retrieve such insights.

IMPORTANCE OF DATA WAREHOUSING

- Enabling informed decision-making
- Consolidating data from multiple sources

IMPORTANCE OF DATA WAREHOUSING

- Enabling informed decision-making
- Consolidating data from multiple sources
- Providing historical perspective on business performance

IMPORTANCE OF DATA WAREHOUSING

- Enabling informed decision-making
- Consolidating data from multiple sources
- Providing historical perspective on business performance
- Supporting complex queries and analytics

IMPORTANCE OF DATA WAREHOUSING

- Enabling informed decision-making
- Consolidating data from multiple sources
- Providing historical perspective on business performance
- Supporting complex queries and analytics
- Improving data quality and consistency

CHALLENGES FOR DATABASES

THE DATA EXPLOSION

- Volume: Exponential growth in data generation
  - Facebook processes 500+ terabytes of data daily

Speaker notes

The world is generating data at an unprecedented rate. It’s crucial to understand the "3 Vs" of Big Data: Volume, Variety, and Velocity. Volume refers to the massive amounts of data created every second—think of platforms like Facebook generating 500+ terabytes daily. Variety represents the different formats of data—structured like SQL tables, semi-structured like JSON, or unstructured data like videos. Lastly, Velocity is about the speed at which data is generated and the need for real-time processing, as seen with stock exchanges like NYSE that produce 1 TB of trade data every day. The challenge is how to store, manage, and analyze such data efficiently, which is where Data Warehousing comes in.

THE DATA EXPLOSION

- Volume: Exponential growth in data generation
  - Facebook processes 500+ terabytes of data daily
- Variety: Structured, semi-structured, and unstructured data
  - Relational databases, JSON, text, images, videos

THE DATA EXPLOSION

- Volume: Exponential growth in data generation
  - Facebook processes 500+ terabytes of data daily
- Variety: Structured, semi-structured, and unstructured data
  - Relational databases, JSON, text, images, videos
- Velocity: Speed of data generation and processing
  - NYSE generates 1 TB of trade data per day

NEED FOR REAL-TIME ANALYTICS

- Businesses require up-to-the-minute insights

> ℹ Fraud detection in financial transactions

> ℹ Real-time inventory management in e-commerce

> ℹ Personalized recommendations in streaming services

Speaker notes

Real-time analytics is becoming increasingly important in various industries. Fraud detection in financial services is a perfect example of why businesses need instant access to fresh data. Delays in identifying fraudulent transactions could cost companies millions. Similarly, e-commerce platforms rely on real-time data to manage inventory, ensuring stock levels are updated as sales happen. In streaming services, real-time recommendations based on user behavior enhance customer experience and keep viewers engaged. Real-time analytics requires data warehousing solutions that can process and provide up-to-the-minute insights from large volumes of streaming data.

DATA INTEGRATION CHALLENGES

- Multiple data sources with different formats
- Data silos in organizations
- Ensuring data consistency across sources

ℹ A multinational company integrating sales data from different countries

Speaker notes

1. Multiple data sources with different formats:
    - Challenge: Organizations often have data spread across various systems (e.g., CRM, ERP, HR systems) in different formats (databases, spreadsheets, text files).
    - Impact: Makes it difficult to get a complete view of the business or perform cross-functional analysis.
    - Solution approach: ETL (Extract, Transform, Load) processes are used to consolidate and standardize data.
2. Data silos in organizations:
    - Definition: Isolated pockets of data that are not easily accessible by other parts of the organization.
    - Causes: Often result from departmental systems, acquisitions, or legacy technology.
    - Problem: Leads to incomplete information for decision-making and potential data inconsistencies.
    - Solution: Data warehouses aim to break down these silos by integrating data from across the organization.
3. Ensuring data consistency across sources:
    - Challenge: Different systems may represent the same data differently (e.g., date formats, customer IDs).
    - Importance: Inconsistent data can lead to incorrect analysis and poor decision-making.
    - Approach: Data governance policies and data quality processes are crucial in maintaining consistency.
4. Real-world example: A multinational company integrating sales data from different countries faces challenges like:
    - Different currencies and exchange rate fluctuations
    - Varying fiscal years and reporting standards
    - Multiple languages and cultural differences in data entry
    - Diverse local systems and data formats
5. Why this matters:
    - Effective data integration is foundational for accurate analytics and reporting.
    - It enables a holistic view of the organization, supporting better strategic decisions.
    - Addressing these challenges is a key part of building a successful data warehouse.

Think about a company you're familiar with (perhaps from a previous job or internship). What different types of data

systems might they use, and what challenges might they face in integrating all their data?

DATA QUALITY AND CONSISTENCY

- Common data quality issues:
  - Duplicates, missing values, inconsistent formats

Speaker notes

Poor data quality leads to incorrect analysis and flawed decision-making. Common issues include duplicate records, missing values, and inconsistent formats across datasets. When multiple systems contribute data to a warehouse, there’s a high risk of inconsistency if not properly managed. Data cleansing and validation are critical steps in any ETL (Extract, Transform, Load) process. Before data is loaded into the warehouse, it needs to be checked and standardized to ensure the accuracy and consistency of the results. Poor data quality not only affects decision-making but can also lead to a loss of trust from customers if, for example, personalized recommendations are inaccurate.

DATA QUALITY AND CONSISTENCY

- Common data quality issues:
  - Duplicates, missing values, inconsistent formats
- Impact of poor data quality:
  - Incorrect analysis and decision-making
  - Loss of customer trust

DATA QUALITY AND CONSISTENCY

- Common data quality issues:
  - Duplicates, missing values, inconsistent formats
- Impact of poor data quality:
  - Incorrect analysis and decision-making
  - Loss of customer trust
- Importance of data cleansing and validation

SCALABILITY AND PERFORMANCE

- Handling growing data volumes
- Maintaining query performance as data size increases
- Balancing read and write operations

ℹ Scaling challenges faced by social media platforms

Speaker notes

As data grows, data warehouses must be designed to scale efficiently. Social media platforms, for instance, face scalability challenges as they need to manage petabytes of user-generated content. A well-designed DW should maintain query performance even as the data scales. One approach is partitioning, where large datasets are split into smaller, more manageable segments, improving the performance of queries. Indexing and using columnar storage are other techniques to enhance performance. Ensuring the warehouse scales to handle increasing read and write operations is critical to maintaining a responsive system.

SECURITY AND PRIVACY CONCERNS

- Protecting sensitive data (e.g., personal information, financial data)

Speaker notes

Data security is one of the most important considerations for any data warehouse. Protecting sensitive data such as personally identifiable information (PII) and financial records is crucial, particularly with stringent regulations like GDPR in the EU or CCPA in California. These regulations enforce strict requirements for data protection, user consent, and the right to be forgotten. Techniques like data masking, encryption, and access control ensure that only authorized personnel can view or modify sensitive data. Furthermore, anonymization and pseudonymization techniques help safeguard individual privacy while allowing businesses to perform analytics on the data.

SECURITY AND PRIVACY CONCERNS

- Protecting sensitive data (e.g., personal information, financial data)
- Compliance with regulations (e.g., GDPR, CCPA)

SECURITY AND PRIVACY CONCERNS

- Protecting sensitive data (e.g., personal information, financial data)
- Compliance with regulations (e.g., GDPR, CCPA)
- Balancing data accessibility with security

SECURITY AND PRIVACY CONCERNS

- Protecting sensitive data (e.g., personal information, financial data)
- Compliance with regulations (e.g., GDPR, CCPA)
- Balancing data accessibility with security
- Challenges of data anonymization

BRIEF HISTORY OF DATABASES

1960S - EARLY DATABASE SYSTEMS

- Hierarchical Databases
  - IBM's Information Management System (IMS)
  - Tree-like structure
  - Limitations: Inflexibility, data redundancy

Speaker notes

1. Context:
    - 1960s: Computers were just beginning to be used for business data processing.
    - Challenge: How to efficiently store and retrieve large amounts of data.
2. Hierarchical Databases:
    - Structure: Tree-like, parent-child relationships.
    - Example: IBM's Information Management System (IMS)
    - How it works: Data is organized in a hierarchy, like an organizational chart.
    - Strengths: Efficient for certain types of relationships (e.g., parts in assemblies).
    - Limitations:
        - Difficulty representing many-to-many relationships.
        - Inflexible for changing business needs.
        - Complex querying for data not following the hierarchy.
3. Network Databases:
    - Structure: Based on the CODASYL model, allowing more complex relationships.
    - Example: Integrated Data Store (IDS)
    - Improvement over hierarchical: Could represent more complex relationships.
    - How it works: Uses records and sets to create network-like connections between data.
    - Limitations:
        - Still complex to manage and query.
        - Required detailed knowledge of the database structure to navigate data.
4. Legacy Impact:
    - Some of these systems (especially IMS) are still used today in certain industries (banking, insurance) due to their reliability and the cost of migration.
5. Historical Significance:
    - These early systems laid the groundwork for future database development.
    - Their limitations directly influenced the development of the relational model in the 1970s.

Study Tip: Try to visualize how you would structure a simple dataset (e.g., employees in a company) using a hierarchical

model. Then, think about what kinds of questions would be easy or difficult to answer with this structure. This exercise can help you understand why these models were eventually superseded.

1960S - EARLY DATABASE SYSTEMS

- Hierarchical Databases
  - ℹ IBM's Information Management System (IMS)
  - Tree-like structure
  - Limitations: Inflexibility, data redundancy
- Network Databases
  - ℹ Integrated Data Store (IDS)
  - Based on the CODASYL model
  - Improvement over hierarchical, but still complex

1970S - RELATIONAL DATABASES AND SQL

- Introduction of the relational model by E.F. Codd (1970)

Speaker notes

1. The Relational Model:
    - Introduced by E.F. Codd in 1970.
    - Core idea: Represent data in tables with rows and columns, using relationships between these tables.
    - Revolutionary because: It provided a more flexible and intuitive way to structure and query
2. Key Concepts:
    - Tables (Relations): Each table represents an entity or concept.
    - Rows (Tuples): Each row is a specific instance or record.
    - Columns (Attributes): Represent properties or characteristics of the entity.
    - Keys: Used to uniquely identify rows and establish relationships between tables.
3. Advantages over earlier models:
    - Flexibility: Easier to modify the database structure as business needs change.
    - Ad-hoc querying: Users can ask complex questions without needing to understand the physical data storage.
    - Data independence: Changes to the physical storage don't affect the logical view of the data.
4. SQL (Structured Query Language):
    - Developed to interact with relational databases.
    - Standardized language for querying and managing relational databases.
    - Made databases accessible to a wider range of users, not just specialized programmers.
5. First Commercial RDBMS:
    - Oracle, founded in 1977, released its first commercial SQL-based RDBMS in 1979.
    - Other early players: IBM's System R (research project that influenced SQL development).
6. Impact:
    - Relational databases quickly became the dominant model for data management.
    - Concepts from the relational model still underpin much of modern data management, including in data warehouses.
7. Modern relevance:
    - Most business applications today use relational databases.
    - SQL remains the standard language for database interaction, with various dialects for different systems.

Practice designing a simple relational database. For example, try to model a library system with books, authors, and borrowers. Think about how you would structure the tables and relationships. This exercise will help you grasp the fundamental concepts of relational database design.

1970S - RELATIONAL DATABASES AND SQL

- Introduction of the relational model by E.F. Codd (1970)
- Key concepts: Tables, rows, columns, keys

1970S - RELATIONAL DATABASES AND SQL

- Introduction of the relational model by E.F. Codd (1970)
- Key concepts: Tables, rows, columns, keys
- Development of SQL (Structured Query Language)

1970S - RELATIONAL DATABASES AND SQL

- Introduction of the relational model by E.F. Codd (1970)
- Key concepts: Tables, rows, columns, keys
- Development of SQL (Structured Query Language)
- First commercial RDBMS: Oracle (1979)

1980S - OBJECT-ORIENTED DATABASES

- Designed to handle complex data structures

Speaker notes

1. Object-Oriented Databases (OODBs) - 1980s:
    1. Purpose:
        - Designed to handle complex data structures that were difficult to represent in relational databases.
        - Aimed to bridge the gap between object-oriented programming and database management.
    2. Key features:
        - Data stored as objects, mirroring object-oriented programming concepts.
        - Support for complex data types and relationships.
        - Ability to store and retrieve complete object structures.
    3. Integration with object-oriented programming languages:
        - Allowed seamless interaction between databases and OOP languages like C++ and Java.
        - Reduced the "impedance mismatch" problem (the difficulty of translating between programming objects and relational database structures).
        - Example: A Java object could be directly stored in and retrieved from the database without needing to be decomposed into tables.
    4. Examples of OODBs:
        - Versant: One of the early commercial OODBs, known for its performance.
        - ObjectStore: Another pioneering OODB, focused on scalability and integration with C++.
        - Other notable examples: Objectivity/DB, Db4o (database for objects)
    5. Limited adoption compared to relational databases:
        - Despite initial enthusiasm, OODBs didn't become as widespread as expected.
        - Reasons for limited adoption:
            - Maturity and widespread use of relational databases
            - Complexity of object-oriented data modeling
            - Lack of standardization (each OODB had its own way of doing things)
            - Performance issues with complex queries
            - Limited support for ad-hoc querying compared to SQL
    6. Legacy and influence:
        - While pure OODBs are not widely used, their concepts influenced:

        - Object-relational databases (e.g., PostgreSQL with its object-relational features)
        - NoSQL databases, particularly document databases
    - Some niche applications in fields like computer-aided design (CAD) and scientific databases still use OODBs
2. Why this is important to understand:
    - Shows the evolution of database technology in response to programming paradigms.
    - Illustrates challenges in adopting new database models, even when they have theoretical advantages.
    - Helps in understanding the strengths and limitations of different database types.

Think about an application you use regularly (e.g., a social media app). Try to imagine how its data might be structured as objects (User objects, Post objects, etc.). Then consider why a relational model might be chosen instead. This exercise will help you grasp the trade-offs between object-oriented and relational data models.

1980S - OBJECT-ORIENTED DATABASES

- Designed to handle complex data structures
- Integration with object-oriented programming languages

1980S - OBJECT-ORIENTED DATABASES

- Designed to handle complex data structures
- Integration with object-oriented programming languages
- Examples: Versant, ObjectStore

1980S - OBJECT-ORIENTED DATABASES

- Designed to handle complex data structures
- Integration with object-oriented programming languages
- Examples: Versant, ObjectStore
- Limited adoption compared to relational databases

1990S - RISE OF DATA WAREHOUSING AND OLAP

- Inmon and Kimball's data warehouse methodologies

1990S - RISE OF DATA WAREHOUSING AND OLAP

- Inmon and Kimball's data warehouse methodologies
- Introduction of star schema and snowflake schema

1990S - RISE OF DATA WAREHOUSING AND OLAP

- Inmon and Kimball's data warehouse methodologies
- Introduction of star schema and snowflake schema
- Development of OLAP (Online Analytical Processing) tools

1990S - RISE OF DATA WAREHOUSING AND OLAP

- Inmon and Kimball's data warehouse methodologies
- Introduction of star schema and snowflake schema
- Development of OLAP (Online Analytical Processing) tools
- Separation of transactional and analytical systems

2000S - NOSQL AND BIG DATA

- NoSQL databases emerge to handle web-scale data
  - Types: Document, Key-value, Column-family, Graph

2000S - NOSQL AND BIG DATA

- NoSQL databases emerge to handle web-scale data
  - Types: Document, Key-value, Column-family, Graph
- Examples: MongoDB, Cassandra, Neo4j

2000S - NOSQL AND BIG DATA

- NoSQL databases emerge to handle web-scale data
  - Types: Document, Key-value, Column-family, Graph
- Examples: MongoDB, Cassandra, Neo4j
- Big Data technologies: Hadoop, MapReduce

2000S - NOSQL AND BIG DATA

- NoSQL databases emerge to handle web-scale data
  - Types: Document, Key-value, Column-family, Graph
- Examples: MongoDB, Cassandra, Neo4j
- Big Data technologies: Hadoop, MapReduce
- Emphasis on scalability and flexibility

2010S - CLOUD DATABASES AND NEWSQL

- Cloud-based database services (DBaaS)
  - Examples: Amazon RDS, Google Cloud SQL, Azure SQL Database

2010S - CLOUD DATABASES AND NEWSQL

- Cloud-based database services (DBaaS)
  - Examples: Amazon RDS, Google Cloud SQL, Azure SQL Database
- NewSQL: Combining ACID properties with NoSQL scalability
  - Examples: Google Spanner, CockroachDB

2010S - CLOUD DATABASES AND NEWSQL

- Cloud-based database services (DBaaS)
  - Examples: Amazon RDS, Google Cloud SQL, Azure SQL Database
- NewSQL: Combining ACID properties with NoSQL scalability
  - Examples: Google Spanner, CockroachDB
- Increased focus on distributed systems and global scale

CURRENT TRENDS

- AI/ML integration in databases
  - Automated tuning, predictive analytics

Speaker notes

1. AI/ML integration in databases:
    - Automated tuning: Databases can optimize themselves based on usage patterns.
    - Predictive analytics: Embedding machine learning models directly into the database for real-time predictions.
    - Why it matters: Reduces need for manual database administration and enables more sophisticated,
2. Graph databases:
    - Purpose: Optimized for managing and querying highly interconnected data.
    - Use cases: Social network analysis, fraud detection, recommendation engines.
    - Examples: Neo4j, Amazon Neptune
    - Why it's important: Enables efficient analysis of relationships in data, which is challenging in traditional relational databases.
3. Multi-model databases:
    - Definition: Databases that can store and process multiple data models (e.g., relational, document, graph) in a single system.
    - Advantage: Provides flexibility to handle diverse data types and queries within one database system.
    - Examples: ArangoDB, OrientDB
    - Why it matters: Simplifies data architecture and reduces the need for multiple specialized databases.
4. Blockchain in database management:
    - Application: Using blockchain technology to create tamper-proof audit trails and ensure data integrity.
    - Potential use cases: Financial transactions, supply chain management, healthcare records.
    - Current status: Still an emerging area, with more potential than widespread adoption.
    - Why it's significant: Could revolutionize how we ensure data integrity and trust in distributed systems.
5. Overall impact of these trends:
    - Databases are becoming more intelligent, flexible, and capable of handling diverse and complex data.
    - The lines between traditional databases, data warehouses, and analytics platforms are blurring.
    - These advancements are enabling new types of applications and business models.
6. Challenges:
    - Keeping up with rapidly evolving technology
    - Ensuring security and privacy with more complex systems

- Managing the increased complexity these advanced features bring

For each trend, try to think of a specific application or company that might benefit from it. For example, how might a social media company use graph databases? How could a bank leverage blockchain in its database systems? This exercise will help you connect these trends to real-world scenarios.

CURRENT TRENDS

- AI/ML integration in databases
  - Automated tuning, predictive analytics
- Graph databases for complex relationship analysis

CURRENT TRENDS

- AI/ML integration in databases
  - Automated tuning, predictive analytics
- Graph databases for complex relationship analysis
- Multi-model databases

CURRENT TRENDS

- AI/ML integration in databases
  - Automated tuning, predictive analytics
- Graph databases for complex relationship analysis
- Multi-model databases
- Blockchain in database management

FOUNDING PRINCIPLES OF DBMS

DATA INDEPENDENCE

- Physical Data Independence
  - Changes in storage structures don't affect application programs

Speaker notes

1. Data Independence:
    - Definition: The ability to change the database structure without affecting the programs or users accessing it.
2. Types of Data Independence:
    1. Physical Data Independence:
        - Definition: The ability to modify the physical storage structure without affecting application programs.
        - What can be changed:
            - Storage devices (e.g., switching from HDD to SSD)
            - File organizations (e.g., changing from heap files to indexed files)
            - Access methods (e.g., implementing new indexing techniques)
        - Example: Moving from a single server to a distributed database system without changing application code.
        - Why it matters: Allows for performance optimizations and hardware upgrades without disrupting applications.
    2. Logical Data Independence:
        - Definition: The ability to change the logical schema without affecting application programs.
        - What can be changed:
            - Adding or removing tables
            - Adding or removing columns in existing tables
            - Changing relationships between tables
        - Example: Adding a new column "loyalty_points" to a Customer table without requiring changes to existing applications.
        - Why it matters: Enables database evolution to meet new business requirements without major application rewrites.
3. Benefits of Data Independence:
    1. Flexibility:
        - Allows the database to evolve as needs change.
        - Supports adaptation to new technologies without massive overhauls.

2. Maintainability:
    - Reduces the impact of database changes on the overall system.
    - Simplifies updates and modifications to either the database or applications.
3. Scalability:
    - Enables growth in data volume or complexity without necessitating application changes.
    - Supports transitions to more powerful hardware or distributed systems.

4. How Data Independence is Achieved:
    - Through the use of abstraction layers in the DBMS architecture.
    - View mechanisms that present a stable interface to applications.
    - Separation of concerns between data storage, data model, and application logic.
5. Challenges in Maintaining Data Independence:
    - Ensuring performance doesn't degrade with abstractions.
    - Balancing flexibility with optimization opportunities.
    - Managing complex view updates in cases of logical data independence.
6. Real-world Importance:
    - Crucial for large, long-lived systems where both the database and applications evolve over time.
    - Essential in enterprise environments where multiple applications share the same database.
    - Key factor in reducing the total cost of ownership for database systems.

Think about a familiar application (e.g., a school management system). Consider how you might need to change the database over time (adding new types of courses, changing storage systems). How would data independence principles help manage these changes without disrupting the entire system? This exercise will help you appreciate the practical value of data independence in real-world scenarios.

DATA INDEPENDENCE

- Physical Data Independence
  - Changes in storage structures don't affect application programs
- Logical Data Independence
  - Changes in logical schema don't affect application programs

DATA INDEPENDENCE

- Physical Data Independence
  - Changes in storage structures don't affect application programs
- Logical Data Independence
  - Changes in logical schema don't affect application programs
- Benefits: Flexibility, maintainability, scalability

ACID PROPERTIES

- **Atomicity**: All-or-nothing transaction execution

ACID PROPERTIES

- **Atomicity**: All-or-nothing transaction execution
- **Consistency**: Database remains in a valid state after transaction

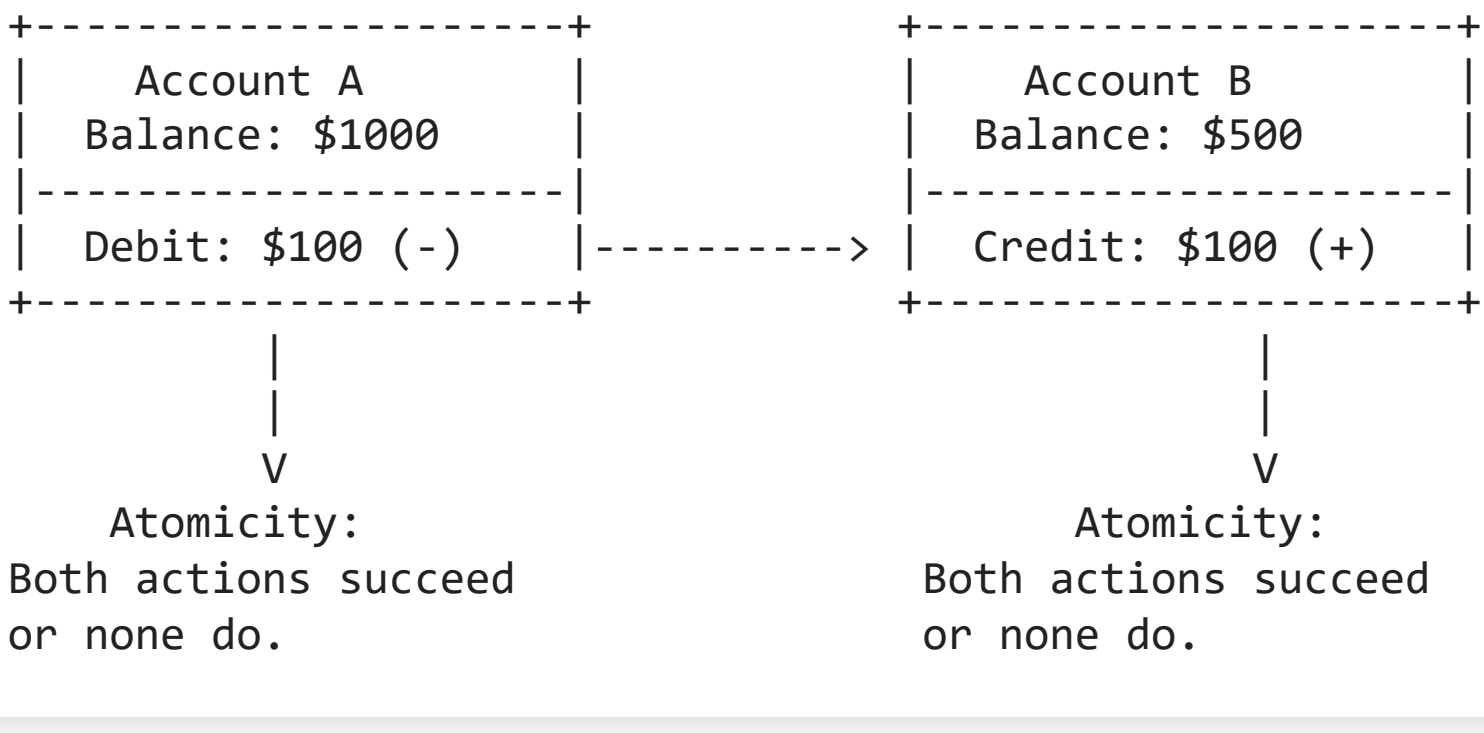
ACID PROPERTIES

- **Atomicity**: All-or-nothing transaction execution
- **Consistency**: Database remains in a valid state after transaction
- **Isolation**: Concurrent transactions don't interfere with each other

ACID PROPERTIES

- **Atomicity**: All-or-nothing transaction execution
- **Consistency**: Database remains in a valid state after transaction
- **Isolation**: Concurrent transactions don't interfere with each other
- **Durability**: Committed transactions are permanent

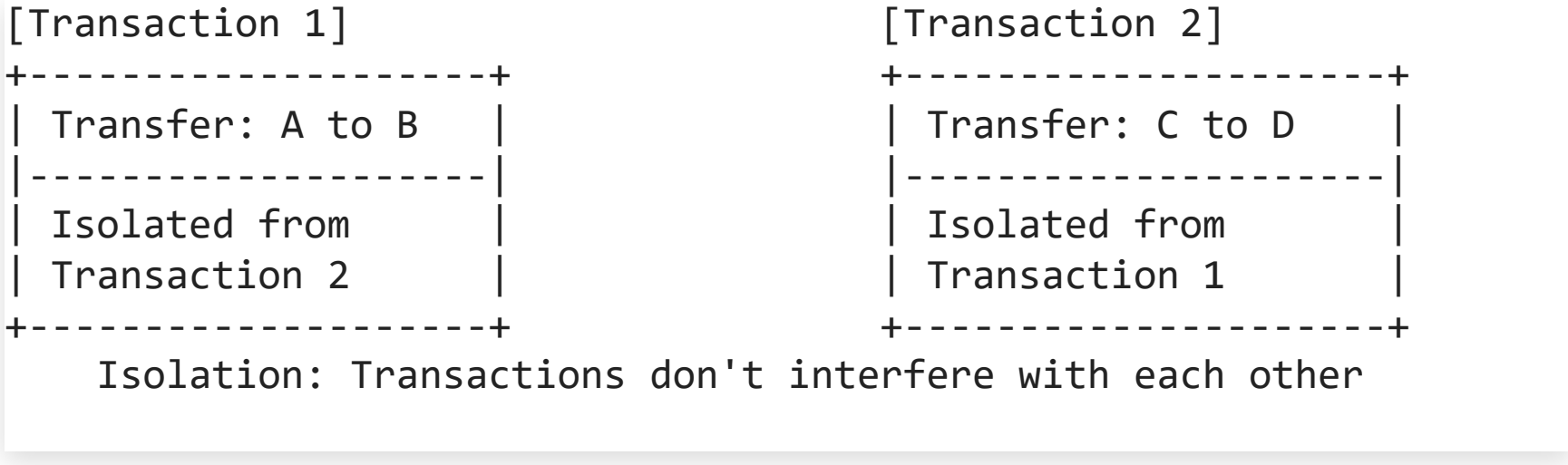
ATOMICITY


Account A
Balance: $1000
Debit: $100 (-)
Account B
Balance: $500
Credit: $100 (+)
Atomicity:
Both actions succeed
or none do.
Atomicity:
Both actions succeed
or none do.

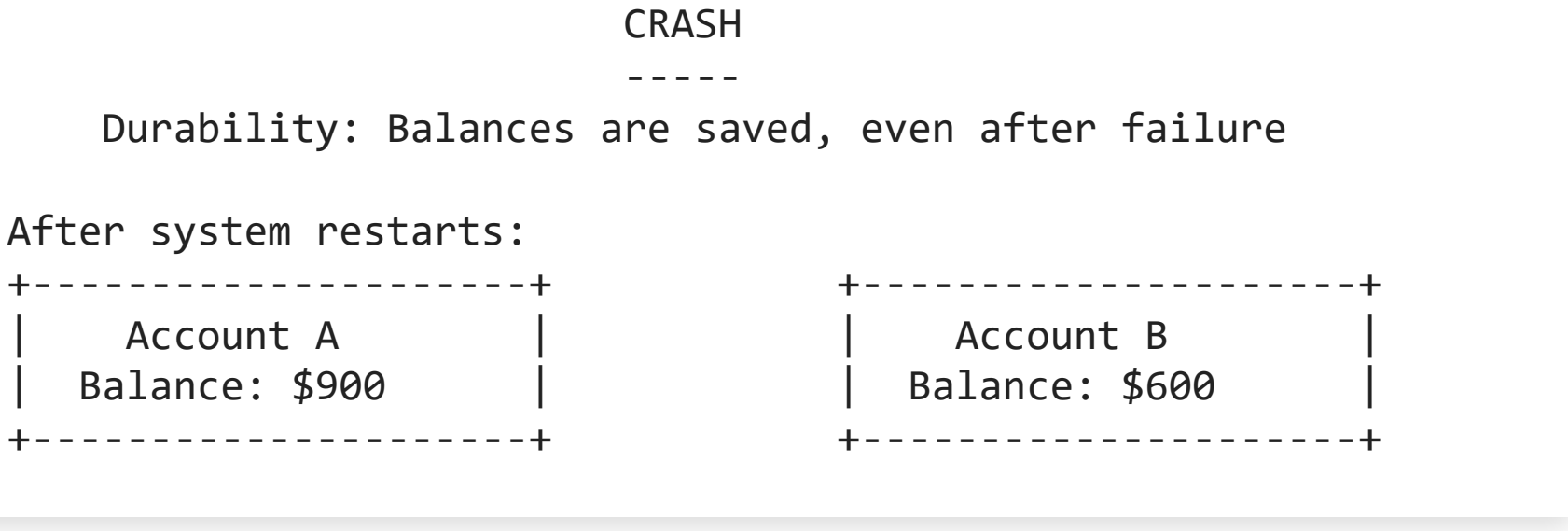
CONSISTENCY

```
Total balance remains the same: $1000 + $500 = $1500
Consistency: Total amount conserved.
```

ISOLATION


[Transaction 1]
Transfer: A to B
Isolated from
Transaction 2
[Transaction 2]
Transfer: C to D
Isolated from
Transaction 1
Isolation: Transactions don't interfere with each other

DURABILITY


CRASH
Durability: Balances are saved, even after failure
After system restarts:
Account A
Balance: $900
Account B
Balance: $600

CONCURRENCY CONTROL

- Managing simultaneous access to data

CONCURRENCY CONTROL

- Managing simultaneous access to data
- Techniques:
  - Locking (Shared locks, Exclusive locks)
  - Multiversion Concurrency Control (MVCC)

CONCURRENCY CONTROL

- Managing simultaneous access to data
- Techniques:
  - Locking (Shared locks, Exclusive locks)
  - Multiversion Concurrency Control (MVCC)
- Dealing with deadlocks

CONCURRENCY CONTROL

- Managing simultaneous access to data
- Techniques:
  - Locking (Shared locks, Exclusive locks)
  - Multiversion Concurrency Control (MVCC)
- Dealing with deadlocks
- Ensuring data consistency in multi-user environments

DATA INTEGRITY AND CONSTRAINTS

- **Entity Integrity**: Primary Key constraints

DATA INTEGRITY AND CONSTRAINTS

- **Entity Integrity**: Primary Key constraints
- **Entity Integrity**: Foreign Key constraints

DATA INTEGRITY AND CONSTRAINTS

- **Entity Integrity**: Primary Key constraints
- **Entity Integrity**: Foreign Key constraints
- **Domain Integrity**: Data type, format constraints

DATA INTEGRITY AND CONSTRAINTS

- **Entity Integrity**: Primary Key constraints
- **Entity Integrity**: Foreign Key constraints
- **Domain Integrity**: Data type, format constraints
- **User-Defined Integrity**: Custom business rules

DATA INTEGRITY AND CONSTRAINTS

- **Entity Integrity**: Primary Key constraints
- **Entity Integrity**: Foreign Key constraints
- **Domain Integrity**: Data type, format constraints
- **User-Defined Integrity**: Custom business rules
- Importance in maintaining data quality

QUERY OPTIMIZATION

- Process of selecting the most efficient query execution plan

Speaker notes

1. Definition: Query optimization is the process of selecting the most efficient way to execute a database query.
2. Importance:
    - Can dramatically improve query performance, turning hours-long queries into seconds.
    - Critical for maintaining responsiveness in large databases.
3. Components of query optimization:
    - Query rewriting:
        - Definition: Restructuring a query to a more efficient form without changing its result.
        - Example: Pushing down filters before joins to reduce the amount of data processed.
        - Why it matters: Can significantly reduce the amount of data the database needs to process.
    - Statistics and cost estimation:
        - Definition: Using database statistics to estimate the cost of different execution plans.
        - Examples of statistics: Table sizes, data distribution, index information.
        - Why it matters: Allows the optimizer to make informed decisions about the best execution plan.
    - Join order selection:
        - Definition: Determining the most efficient order to join tables in a query.
        - Example: In a query joining tables A, B, and C, is it more efficient to join A and B first, or B and C?
        - Why it matters: Join order can have a massive impact on query performance, especially for queries involving many tables.
4. How query optimization works:
    - The database analyzes multiple possible execution plans.
    - It estimates the cost of each plan based on statistics and heuristics.
    - It selects the plan with the lowest estimated cost.
5. Challenges in query optimization:
    - Statistics can become outdated, leading to suboptimal plans.
    - Complex queries can have an enormous number of possible execution plans.
    - Optimizing for all possible queries is computationally infeasible.
6. Practical implications:
    - Understanding query optimization can help in writing more efficient SQL queries.

- It's crucial for database administrators in tuning database performance.
- In data warehouses, query optimization is especially important due to the complex, analytical nature of queries.

Try writing a complex SQL query (e.g., joining several tables with multiple conditions) and then use your database's EXPLAIN feature to see the execution plan. Try rewriting the query in different ways and observe how the execution plan changes. This hands-on experience will help you understand query optimization in practice.

QUERY OPTIMIZATION

- Process of selecting the most efficient query execution plan
- Components of query optimization:
  - Query rewriting
  - Statistics and cost estimation
  - Join order selection

QUERY OPTIMIZATION

- Process of selecting the most efficient query execution plan
- Components of query optimization:
  - Query rewriting
  - Statistics and cost estimation
  - Join order selection
- Impact on database performance

TRANSACTION MANAGEMENT

- Definition of database transactions

Speaker notes

Transaction management in databases ensures the ACID properties—Atomicity, Consistency, Isolation, and Durability—are maintained. When multiple users or processes access a database concurrently, the system must ensure that these transactions do not interfere with one another, ensuring data integrity. One way to manage this is through locking mechanisms that prevent conflicting changes to the same data. Another is Multiversion Concurrency Control (MVCC), where each user sees a consistent snapshot of the database. It's crucial to handle transaction failures effectively by rolling back incomplete transactions and ensuring no data corruption occurs.

TRANSACTION MANAGEMENT

- Definition of database transactions
- Transaction states: Active, Partially Committed, Failed, Aborted, Committed

TRANSACTION MANAGEMENT

- Definition of database transactions
- Transaction states: Active, Partially Committed, Failed, Aborted, Committed
- Transaction scheduling

TRANSACTION MANAGEMENT

- Definition of database transactions
- Transaction states: Active, Partially Committed, Failed, Aborted, Committed
- Transaction scheduling
- Handling transaction failures and system crashes

RECOVERY MECHANISMS

- Ensuring data persistence and consistency after failures

Speaker notes

Recovery mechanisms in databases ensure that data remains consistent and durable, even in the event of a failure. Techniques like Write-Ahead Logging (WAL) ensure that all changes are logged before they are applied to the database. If a crash occurs, the database can roll back incomplete transactions or roll forward completed transactions from the log. Checkpointing creates a stable point to which the database can restore in case of failure, minimizing recovery time. Recovery mechanisms are especially important in large, distributed databases to prevent data loss and corruption across nodes.

RECOVERY MECHANISMS

- Ensuring data persistence and consistency after failures
- Recovery techniques:
  - Write-Ahead Logging (WAL)
  - Checkpointing
  - Rollback and Rollforward operations

RECOVERY MECHANISMS

- Ensuring data persistence and consistency after failures
- Recovery techniques:
  - Write-Ahead Logging (WAL)
  - Checkpointing
  - Rollback and Rollforward operations
- Balancing performance and reliability

IN-CLASS EXERCISE: DBMS PRINCIPLES

EXERCISE INSTRUCTIONS

- Form groups of 3-4 students
- Each group will be assigned a real-world scenario
- Identify which DBMS principles are most relevant to the scenario
- Discuss how these principles address the challenges in the scenario
- Prepare a brief presentation of your findings

SCENARIO EXAMPLES

1. Online banking system handling thousands of transactions per second
2. E-commerce platform updating inventory across multiple warehouses
3. Healthcare system managing patient records with strict privacy requirements
4. Social media platform supporting millions of concurrent users

link

DATA WAREHOUSE: MOTIVATIONS

LIMITATIONS OF OPERATIONAL DATABASES FOR ANALYTICS

- Designed for day-to-day transactions, not complex queries

Speaker notes

1. Operational databases are designed for day-to-day transactions, not complex analytics:
    - Optimized for quick inserts and updates, not large-scale data retrieval.
    - Schema designed for operational efficiency, not analytical queries.
2. Performance impact of analytical queries on operational systems:
    - Complex queries can slow down critical business operations.
    - Example: Running a year-end sales analysis could impact the system's ability to process new orders.
3. Lack of historical data retention:
    - Operational systems often only keep current or recent data.
    - Example: If you want to analyze sales trends over the past 5 years, but your system only keeps the last 6 months of data, you can't perform the analysis.
4. Data scattered across multiple systems:
    - In many organizations, relevant data is spread across various operational systems.
    - Example: Customer information might be in a CRM system, their purchase history in an ERP system, and their support tickets in a helpdesk system.
5. Why these limitations led to data warehouses:
    - Data warehouses are designed to address these specific challenges.
    - They provide a centralized repository optimized for analytical queries.
    - Allow integration of data from multiple sources into a consistent format.
    - Designed to store and manage historical data effectively.
6. Impact on business:
    - Without addressing these limitations, businesses struggle to gain comprehensive insights from their data.
    - Data-driven decision making becomes challenging and time-consuming.
    - Competitive advantage can be lost to more data-savvy competitors.

Think about a business you're familiar with. What kinds of analytical questions might they want to ask that would be

difficult with just operational databases? How might a data warehouse help them answer these questions more effectively?

LIMITATIONS OF OPERATIONAL DATABASES FOR ANALYTICS

- Designed for day-to-day transactions, not complex queries
- Performance impact of analytical queries on operational systems

LIMITATIONS OF OPERATIONAL DATABASES FOR ANALYTICS

- Designed for day-to-day transactions, not complex queries
- Performance impact of analytical queries on operational systems
- Lack of historical data retention

LIMITATIONS OF OPERATIONAL DATABASES FOR ANALYTICS

- Designed for day-to-day transactions, not complex queries
- Performance impact of analytical queries on operational systems
- Lack of historical data retention
- Data scattered across multiple systems

Analyst: "*How many sales completed in dec. before Christmas per group of product and discount?*"

```
SELECT Y.year, PG.name, DI.disc, count(*)
FROM year Y, month M, day D, session S,
  line_item I, order O, product P, productgroup PG,
  discount DI, order_status OS
WHERE M.year_id = Y.id and
  D.month_id = M.id and
  S.day_id = D.id and
  O.session_id = S.id and
  I.order_id = O.id and
  I.product_id = P.id and
  P.productgroup_id = PG.id and
  DI.productgroup_id = PG.id and
  O.id = OS.order_id and
  D.day < 24 and
  M.month = 12
  and OS.status='FINISHED'
GROUP BY Y.year, PG.name, DI.discount
ORDER BY Y.year, DI.discount
```

Source: Ulf Leser, Data Warehouses course

Large relationships (millions of orders, sessions), numerous joins

⇒ potentially difficult query.

NEED FOR HISTORICAL AND AGGREGATED DATA

- Business requirements for trend analysis

NEED FOR HISTORICAL AND AGGREGATED DATA

- Business requirements for trend analysis
- Comparing current performance with historical data

NEED FOR HISTORICAL AND AGGREGATED DATA

- Business requirements for trend analysis
- Comparing current performance with historical data
- Aggregations for different time periods (daily, monthly, yearly)

NEED FOR HISTORICAL AND AGGREGATED DATA

- Business requirements for trend analysis
- Comparing current performance with historical data
- Aggregations for different time periods (daily, monthly, yearly)

ℹ Analyzing sales trends over the past 5 years

SUPPORT FOR COMPLEX QUERIES AND REPORTING

- Ad-hoc querying capabilities

Speaker notes

1. Ad-hoc querying capabilities:
    - Definition: Ability for users to create custom, on-the-fly queries without predefined templates.
    - Why it's important: Allows business users to explore data freely, answering new questions as they arise.
    - Example: A marketing manager wanting to quickly analyze the effectiveness of a campaign across different customer segments and regions.
2. Handling multi-dimensional analysis:
    - Definition: Analyzing data across multiple dimensions simultaneously (e.g., time, geography, product).
    - How it works: Data is structured in a way that allows quick "slicing and dicing" across dimensions.
    - Example: Analyzing sales by product category, region, time period, and customer demographic all at once.
    - Why it matters: Provides a comprehensive view of business performance and allows for deep, nuanced analysis.
3. Rapid response times for large datasets:
    - How it's achieved: Through specific design choices like denormalization, pre-aggregation, and specialized indexing.
    - Impact: Queries that might take hours on an operational system could return results in seconds in a well-designed data warehouse.
    - Why it's crucial: Enables interactive analysis and rapid decision-making based on large volumes of data.
4. Supporting various reporting tools and dashboards:
    - Types of tools: Business Intelligence (BI) software, data visualization tools, custom reporting applications.
    - Examples: Tableau, Power BI, Looker, QlikView.
    - Benefits:
        - Provides user-friendly interfaces for non-technical users to access and analyze data.
        - Enables creation of dynamic, interactive dashboards for monitoring key business metrics.
        - Allows for scheduled report generation and distribution.
5. Real-world applications:
    - Financial analysis: Quickly assessing profitability across multiple product lines and regions.
    - Customer segmentation: Identifying high-value customer groups based on various attributes and

behaviors.
    - Supply chain optimization: Analyzing inventory levels, supplier performance, and demand patterns across the entire supply network.
6. Skills needed:
    - SQL for complex querying
    - Understanding of dimensional modeling concepts
    - Familiarity with BI and data visualization tools

If possible, get hands-on experience with a BI tool like Tableau Public (free version available). Try connecting to a sample dataset and creating some multi-dimensional visualizations. This practical experience will help you understand the power of these analytical capabilities.

SUPPORT FOR COMPLEX QUERIES AND REPORTING

- Ad-hoc querying capabilities
- Handling multi-dimensional analysis

SUPPORT FOR COMPLEX QUERIES AND REPORTING

- Ad-hoc querying capabilities
- Handling multi-dimensional analysis
- Rapid response times for large datasets

SUPPORT FOR COMPLEX QUERIES AND REPORTING

- Ad-hoc querying capabilities
- Handling multi-dimensional analysis
- Rapid response times for large datasets
- Supporting various reporting tools and dashboards

IMPROVED DECISION-MAKING AND BUSINESS INTELLIGENCE

- Providing a single, consistent view of business data

IMPROVED DECISION-MAKING AND BUSINESS INTELLIGENCE

- Providing a single, consistent view of business data
- Enabling data-driven decision making

IMPROVED DECISION-MAKING AND BUSINESS INTELLIGENCE

- Providing a single, consistent view of business data
- Enabling data-driven decision making
- Supporting predictive analytics and forecasting

IMPROVED DECISION-MAKING AND BUSINESS INTELLIGENCE

- Providing a single, consistent view of business data
- Enabling data-driven decision making
- Supporting predictive analytics and forecasting

ℹ Using historical sales data to predict future demand

DATA CONSOLIDATION AND SINGLE VERSION OF TRUTH

- Integrating data from multiple sources

DATA CONSOLIDATION AND SINGLE VERSION OF TRUTH

- Integrating data from multiple sources
- Resolving data inconsistencies and conflicts

DATA CONSOLIDATION AND SINGLE VERSION OF TRUTH

- Integrating data from multiple sources
- Resolving data inconsistencies and conflicts
- Providing a unified view of the organization

DATA CONSOLIDATION AND SINGLE VERSION OF TRUTH

- Integrating data from multiple sources
- Resolving data inconsistencies and conflicts
- Providing a unified view of the organization
- Ensuring data quality and consistency across the enterprise

DATA WAREHOUSE: DEFINITIONS

BILL INMON'S DEFINITION

> *"A subject-oriented, integrated, time-variant, and non-volatile collection of data in support of management's decision-making process."*

- **Subject-oriented**: Organized around major subjects (e.g., customer, product)

Speaker notes

"father of data warehousing"

1. Subject-oriented:
    - Meaning: Data is organized around major subjects of the enterprise (e.g., customers, products, sales).
    - Contrast with operational systems: These are often organized around specific applications or processes.
    - Example: Instead of having separate data for the order system, inventory system, and customer service system, a data warehouse would organize all relevant data around the concept of "sales."
    - Why it matters: Provides a business-centric view of data, making it easier for analysts to work with.
2. Integrated:
    - Meaning: Data from different sources is merged into a consistent format.
    - Challenges addressed: Resolves differences in naming conventions, encoding structures, attribute measures, etc.
    - Example: Combining data where one system uses "Gender" (M/F) and another uses "Sex" (0/1) into a standardized format.
    - Why it matters: Ensures consistency and reliability in reporting and analysis across the entire organization.
3. Time-variant:
    - Meaning: The data warehouse keeps historical data, not just current data.
    - How it's implemented: Often includes a time dimension in its structure.
    - Example: Storing multiple versions of a product price over time, not just the current price.
    - Why it matters: Enables trend analysis, year-over-year comparisons, and other time-based analytics.
4. Non-volatile:
    - Meaning: Once data enters the warehouse, it doesn't change.
    - How it works: Data is typically loaded in regular batches and is not continuously updated like in operational systems.
    - Example: Yesterday's sales figures, once loaded into the warehouse, remain constant.
    - Why it matters: Ensures consistent reporting results and provides a stable environment for complex queries.
5. In support of management's decision-making process:
    - Overall purpose: To provide reliable, comprehensive data for strategic decision-making.
    - Types of decisions supported: Long-term strategic planning, performance evaluation, trend analysis.

6. Inmon's approach (also known as Corporate Information Factory):
    - Advocates for a top-down design approach.
    - Emphasizes a centralized data warehouse that feeds departmental data marts.

Try to think of examples for each characteristic from a business you're familiar with. How might their data be subject-oriented? What kinds of historical data might they need to keep? This exercise will help you understand how these concepts apply in real-world scenarios.

BILL INMON'S DEFINITION

> *"A subject-oriented, integrated, time-variant, and non-volatile collection of data in support of management's decision-making process."*

- **Subject-oriented**: Organized around major subjects (e.g., customer, product)
- **Integrated**: Consistent naming conventions, formats, encoding structures

BILL INMON'S DEFINITION

> *"A subject-oriented, integrated, time-variant, and non-volatile collection of data in support of management's decision-making process."*

- **Subject-oriented**: Organized around major subjects (e.g., customer, product)
- **Integrated**: Consistent naming conventions, formats, encoding structures
- **Time-variant**: Explicitly contains time dimension

BILL INMON'S DEFINITION

> *"A subject-oriented, integrated, time-variant, and non-volatile collection of data in support of management's decision-making process."*

- **Subject-oriented**: Organized around major subjects (e.g., customer, product)
- **Integrated**: Consistent naming conventions, formats, encoding structures
- **Time-variant**: Explicitly contains time dimension
- **Non-volatile**: Data is stable and doesn't change once it's in the warehouse

RALPH KIMBALL'S DEFINITION

> *"A copy of transaction data specifically structured for query and analysis."*

- Key aspects of Kimball's approach:

Speaker notes

1. Interpretation of the definition:
    - "Copy of transaction data": Implies that the data warehouse doesn't replace operational systems but replicates their data.
    - "Specifically structured": The data is reorganized and optimized for analytical purposes.
    - "For query and analysis": The primary goal is to support business intelligence and decision-making processes.
2. Key aspects of Kimball's approach:
    1. Dimensional modeling:
        - Definition: A technique for structuring data in a way that's intuitive for business users and optimized for query performance.
        - Key components:
            - Fact tables: Contain quantitative metrics of business processes (e.g., sales amounts, quantities).
            - Dimension tables: Contain descriptive attributes (e.g., product details, customer information, time).
        - Benefits:
            - Simplifies complex queries
            - Improves query performance
            - Makes data more understandable to business users
        - Example: A sales fact table might have foreign keys to dimensions like Date, Product, Customer, and Store.
    2. Bus architecture:
        - Definition: A design approach that uses standardized dimensions across different business processes.
        - How it works:
            - Identifies key business processes (e.g., sales, orders, inventory)
            - Defines conformed dimensions that can be used across these processes
        - Benefits:
            - Enables integration of data marts across the enterprise

        - Ensures consistency in reporting across different business areas
    - Example: A "Customer" dimension used consistently across sales, support, and marketing data marts.
3. Focus on business processes:
    - Approach: Organizes the data warehouse around core business processes rather than departments.
    - Why it matters:
        - Aligns the data warehouse with how the business actually operates
        - Facilitates end-to-end analysis of business processes
        - Makes the data warehouse more adaptable to organizational changes
    - Example: Focusing on an "Order to Cash" process rather than separate "Sales" and "Finance" data marts.

3. Kimball vs. Inmon approach:
    - Kimball advocates a bottom-up approach, starting with individual data marts.
    - Inmon prefers a top-down approach with a centralized data warehouse.
    - Kimball's approach often allows for faster implementation and more flexibility.
4. Impact on data warehouse design:
    - Emphasis on creating a user-friendly, business-oriented data structure.
    - Use of star schemas or snowflake schemas in database design.
    - Development of conformed dimensions for enterprise-wide consistency.
5. Skills needed to implement Kimball's approach:
    - Understanding of business processes and metrics
    - Proficiency in dimensional modeling techniques
    - Ability to design and implement star schemas
    - Knowledge of ETL processes to populate dimensional models

Try to design a simple star schema for a business process you're familiar with (e.g., sales, library book checkouts). Identify what would be in the fact table and what dimensions you'd need. This exercise will help you grasp the practical application of Kimball's dimensional modeling concept.

RALPH KIMBALL'S DEFINITION

"A copy of transaction data specifically structured for query and analysis."

- Key aspects of Kimball's approach:
  - Dimensional modeling

RALPH KIMBALL'S DEFINITION

"A copy of transaction data specifically structured for query and analysis."

- Key aspects of Kimball's approach:
  - Dimensional modeling
  - Bus architecture

RALPH KIMBALL'S DEFINITION

> *"A copy of transaction data specifically structured for query and analysis."*

- Key aspects of Kimball's approach:
  - Dimensional modeling
  - Bus architecture
  - Focus on business processes

KEY CHARACTERISTICS OF A DATA WAREHOUSE

- Centralized repository

KEY CHARACTERISTICS OF A DATA WAREHOUSE

- Centralized repository
- Optimized for reading and analysis

KEY CHARACTERISTICS OF A DATA WAREHOUSE

- Centralized repository
- Optimized for reading and analysis
- Contains both detailed and summarized data

KEY CHARACTERISTICS OF A DATA WAREHOUSE

- Centralized repository
- Optimized for reading and analysis
- Contains both detailed and summarized data
- Supports time series and trend analysis

KEY CHARACTERISTICS OF A DATA WAREHOUSE

- Centralized repository
- Optimized for reading and analysis
- Contains both detailed and summarized data
- Supports time series and trend analysis
- Metadata-driven

COMPARISON WITH OPERATIONAL DATABASES

Aspect	Data Warehouse	Operational Database
Purpose	Analytics	Transactions
Data model	Dimensional	Normalized
Data freshness	Periodic updates	Real-time
Query complexity	Complex, unpredictable	Simple, predictable
User base	Analysts, executives	Clerks, customers

OLTP VS OLAP

OLTP (ONLINE TRANSACTION PROCESSING)

- Characteristics:
  - Handles day-to-day transactions
  - Short, simple transactions
  - High concurrency
  - Predictable, repetitive queries

OLTP (ONLINE TRANSACTION PROCESSING)

- Characteristics:
  - Handles day-to-day transactions
  - Short, simple transactions
  - High concurrency
  - Predictable, repetitive queries
- Use cases:
  - Banking transactions
  - Airline reservations
  - Order processing

DATABASE DESIGN FOR OLTP

- Normalized data model (3NF)

DATABASE DESIGN FOR OLTP

- Normalized data model (3NF)
- Optimized for write operations

DATABASE DESIGN FOR OLTP

- Normalized data model (3NF)
- Optimized for write operations
- Index design for quick lookups

DATABASE DESIGN FOR OLTP

- Normalized data model (3NF)
- Optimized for write operations
- Index design for quick lookups
- Focus on data integrity and consistency

OLAP (ONLINE ANALYTICAL PROCESSING)

- Characteristics:
  - Supports complex analytical queries
  - Aggregations and summarizations
  - Lower concurrency, longer-running queries
  - Historical and predictive analysis

OLAP (ONLINE ANALYTICAL PROCESSING)

- Characteristics:
  - Supports complex analytical queries
  - Aggregations and summarizations
  - Lower concurrency, longer-running queries
  - Historical and predictive analysis
- Use cases:
  - Sales analysis
  - Financial reporting
  - Customer segmentation

MULTIDIMENSIONAL DATA MODEL

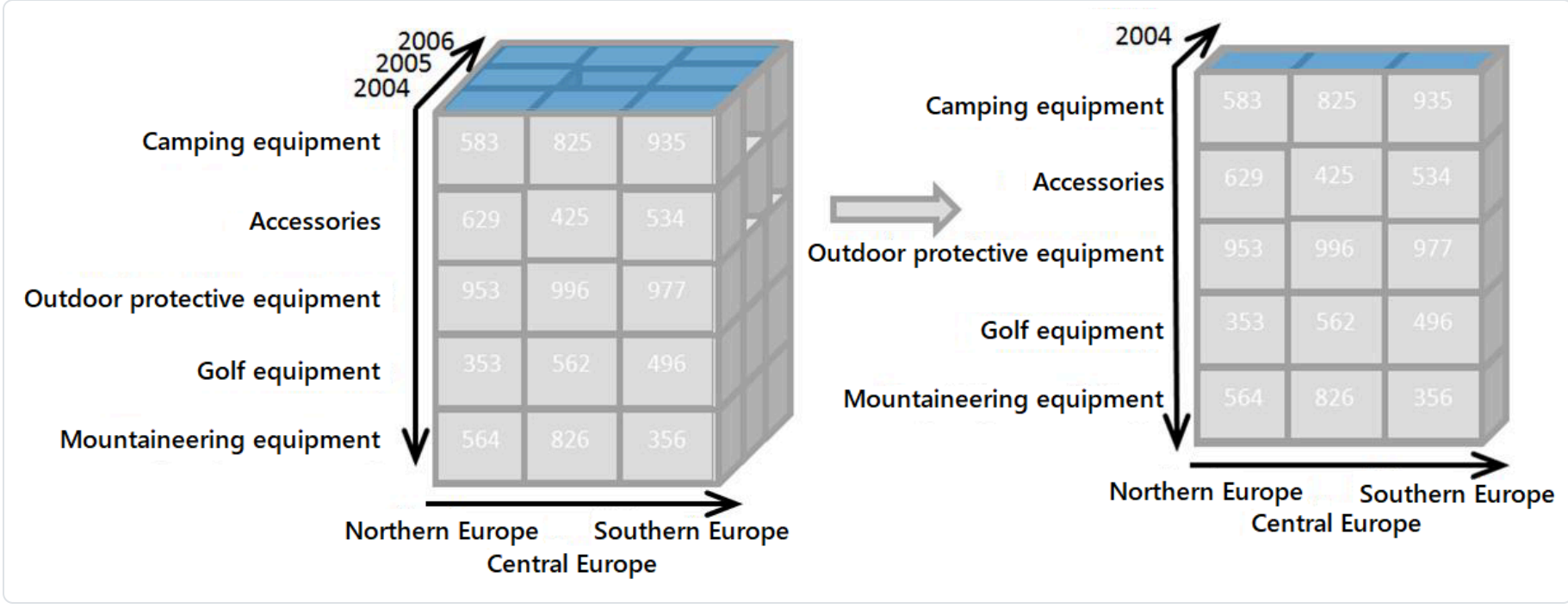
- **Dimensions**: Descriptive attributes (e.g., time, product, location)

MULTIDIMENSIONAL DATA MODEL

- **Dimensions**: Descriptive attributes (e.g., time, product, location)
- **Measures**: Numerical values for analysis

MULTIDIMENSIONAL DATA MODEL

- **Dimensions**: Descriptive attributes (e.g., time, product, location)
- **Measures**: Numerical values for analysis
- Cube structure: Allows quick slicing and dicing of data



OLAP cube slicing

COMPARISON OF OLTP AND OLAP

Aspect	OLTP	OLAP
Workload	Many short, atomic transactions	Few complex queries
Data model	Highly normalized	Typically denormalized (star or snowflake schema)
User types	Clerks, customers, automated processes	Knowledge workers, business analysts, executives
Records accessed	Tens	Millions

Speaker notes

This slide compares Online Transaction Processing (OLTP) with Online Analytical Processing (OLAP), two core database processing paradigms. OLTP is designed for high concurrency and supports day-to-day transactional tasks like customer order processing or ATM withdrawals. Its database design is highly normalized to avoid redundancy and ensure data integrity. OLAP, on the other hand, is optimized for complex queries and analytical reporting. It often uses a denormalized schema like a star or snowflake model to support rapid aggregation and multidimensional analysis, such as sales trends over time. OLTP databases prioritize quick writes and updates, while OLAP systems are optimized for read-heavy operations.

IN-CLASS EXERCISE: OLTP VS OLAP

EXERCISE INSTRUCTIONS

- Students will be given a list of business scenarios
- For each scenario, identify whether it's better suited for OLTP or OLAP
- Justify your choice based on the characteristics we've discussed
- We'll discuss the answers as a class

SCENARIO EXAMPLES

1. Processing customer orders on an e-commerce website
2. Analyzing customer buying patterns over the last 5 years
3. Updating inventory levels after each sale
4. Generating a report on the top-selling products by region
5. Recording patient visits in a hospital
6. Predicting future sales based on historical data and market trends

link

DW INDUSTRIAL LANDSCAPE

MAJOR PLAYERS IN THE DW MARKET

- Traditional vendors:
  - Oracle
  - IBM (Db2)
  - Microsoft (SQL Server)
  - Teradata

MAJOR PLAYERS IN THE DW MARKET

- Traditional vendors:
  - Oracle
  - IBM (Db2)
  - Microsoft (SQL Server)
  - Teradata

MAJOR PLAYERS IN THE DW MARKET

- Traditional vendors:
  - Oracle
  - IBM (Db2)
  - Microsoft (SQL Server)
  - Teradata

CLOUD DATA WAREHOUSE SOLUTIONS

- Benefits of cloud data warehouses:
  - Scalability
  - Cost-effectiveness
  - Managed services
  - Integration with cloud ecosystems

CLOUD DATA WAREHOUSE SOLUTIONS

- Benefits of cloud data warehouses:
  - Scalability
  - Cost-effectiveness
  - Managed services
  - Integration with cloud ecosystems
- Comparison of leading cloud DW solutions

CLOUD DATA WAREHOUSE SOLUTIONS

- Benefits of cloud data warehouses:
  - Scalability
  - Cost-effectiveness
  - Managed services
  - Integration with cloud ecosystems
- Comparison of leading cloud DW solutions
- Hybrid and multi-cloud strategies

OPEN-SOURCE DATA WAREHOUSE TOOLS

- Apache Hadoop ecosystem:
  - Hive, HBase, Impala

OPEN-SOURCE DATA WAREHOUSE TOOLS

- Apache Hadoop ecosystem:
  - Hive, HBase, Impala
- Analytical databases:
  - ClickHouse, Apache Druid

OPEN-SOURCE DATA WAREHOUSE TOOLS

- Apache Hadoop ecosystem:
  - Hive, HBase, Impala
- Analytical databases:
  - ClickHouse, Apache Druid
- ETL and data integration:
  - Apache NiFi, Talend Open Studio

OPEN-SOURCE DATA WAREHOUSE TOOLS

- Apache Hadoop ecosystem:
  - Hive, HBase, Impala
- Analytical databases:
  - ClickHouse, Apache Druid
- ETL and data integration:
  - Apache NiFi, Talend Open Studio
- Visualization and BI:
  - Apache Superset, Metabase

EMERGING TRENDS IN DATA WAREHOUSING

- Real-time data warehousing:
  - Streaming data integration
  - Real-time analytics

Speaker notes

Data lakes are repositories that store vast amounts of raw, unstructured, and semi-structured data. Unlike traditional data warehouses, which require data to be transformed before storage, data lakes store everything as-is. They are ideal for capturing streaming data from IoT devices or social media platforms. Increasingly, companies are integrating data lakes with data warehouses to form a "data lakehouse" architecture, where raw data is stored in the lake but can be processed and analyzed through a warehouse. This hybrid approach gives businesses the flexibility to handle both real-time and historical data in a unified ecosystem.

EMERGING TRENDS IN DATA WAREHOUSING

- Real-time data warehousing:
  - Streaming data integration
  - Real-time analytics
- Data lake integration:
  - Data lakehouse concept
  - Unified analytics on structured and unstructured data

EMERGING TRENDS IN DATA WAREHOUSING

- Real-time data warehousing:
  - Streaming data integration
  - Real-time analytics
- Data lake integration:
  - Data lakehouse concept
  - Unified analytics on structured and unstructured data

CONCLUSION AND PREVIEW

RECAP OF KEY POINTS

- Challenges driving the need for data warehouses
- Evolution of database technologies
- Fundamental principles of DBMS
- Distinctions between OLTP and OLAP systems
- Current landscape and trends in data warehousing

PREVIEW OF NEXT WEEK'S TOPICS

- Data warehouse architectures
- Dimensional modeling
- Introduction to ETL processes

Speaker notes

- A typical data warehouse architecture follows a multi-layered approach. The source layer contains the raw data from various transactional systems. The ETL (Extract, Transform, Load) layer processes this data by extracting it from multiple sources, transforming it into a consistent format, and loading it into the warehouse. The data storage layer is where the processed data resides—usually in a dimensional format to support analytical queries. Finally, the presentation layer is what users interact with through reporting tools, dashboards, or direct queries. This architecture supports the separation of concerns: operational systems handle transactions, while the warehouse manages analytics.
- Dimensional modeling is a design technique used in data warehouses that focuses on simplifying complex queries. It organizes data into "facts" and "dimensions." Facts are quantitative data points (like sales amounts or quantities) while dimensions are descriptive data that provide context (like time, location, or product categories). A common representation of this model is the star schema, where facts are at the center of the "star," and dimensions surround it. The snowflake schema is a more normalized version, where dimensions themselves are further broken down into related sub-dimensions. This model helps optimize queries for fast retrieval and is especially useful for reporting and analytical tasks.
- ETL (Extract, Transform, Load) processes are critical to data warehousing. Data is first extracted from multiple, often heterogeneous sources, such as databases, flat files, or APIs. It’s then transformed, which can involve data cleaning, normalization, and aggregating records. For example, sales data from different countries might need to be converted to a standard currency or date format. The transformed data is then loaded into the data warehouse for analysis. ETL processes often happen in batch jobs, especially during off-hours to reduce the load on operational systems. For real-time data warehouses, ELT (Extract, Load, Transform) processes are more common, where data is loaded first and then processed within the warehouse.

Q&A

If you have any questions or comments, please do not hesitate to contact me:

me[at]phparis[dot]net