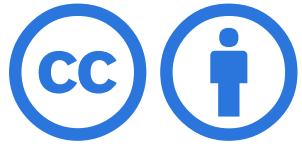
université
PARIS-SACLAY

DATA WAREHOUSE I

WEEK 2



MULTI-DIMENSIONAL MODEL

INTRODUCTION TO MULTI-DIMENSIONAL MODEL

📖 A data structure optimized for data analysis

🎯 **Purpose**: Enable complex analytical and ad-hoc queries with rapid execution time

🔑 **Key components**: dimensions, measures, facts

DIMENSIONS

🕮 Descriptive attributes by which facts are analyzed

- Examples: time, product, customer, location

Dimensions are tables containing:

- **Attributes**: Descriptive properties of a dimension
  - product name, color, size
- **Hierarchies**: Logical structures within a dimension that support different levels of granularity
  - Example: Year > Quarter > Month > Day

Speaker notes

DIMENSIONS IN DATA WAREHOUSING

DEFINITION

Dimensions are descriptive attributes used to analyze facts in a data warehouse. They provide the context for numerical measures (facts) and enable various types of analysis.

KEY CHARACTERISTICS

1. Serve as the foundation for querying and filtering data
2. Organize data into hierarchies, allowing for drill-down and roll-up operations
3. Contain textual descriptions that give meaning to the numerical facts

COMMON EXAMPLES

- **Time:** Allows analysis of data over different time periods (e.g., day, month, quarter, year)
- **Product:** Enables analysis of data by product categories, individual products, etc.
- **Customer:** Facilitates analysis based on customer attributes (e.g., demographics, behavior)
- **Location:** Supports geographical analysis (e.g., country, region, city)

ATTRIBUTES AND HIERARCHIES

- **Attributes:** Descriptive properties of a dimension (e.g., product name, color, size)
- **Hierarchies:** Logical structures within a dimension that support different levels of granularity
    - Example: Time hierarchy - Year > Quarter > Month > Day

TYPES OF DIMENSIONS

1. CONFORMED DIMENSIONS

- **Definition:** Dimensions that are shared across multiple fact tables or data marts
- **Purpose:** Ensure consistency and enable integrated analysis across the entire data warehouse
- **Example:** A customer dimension used in both sales and support fact tables

2. ROLE-PLAYING DIMENSIONS

- **Definition:** A single dimension used multiple times in a fact table, each time with a different context
- **Purpose:** Reduce redundancy and save storage while maintaining logical distinctions
- **Example:** A date dimension playing roles like order date, ship date, and delivery date in an order fact table

3. JUNK DIMENSIONS

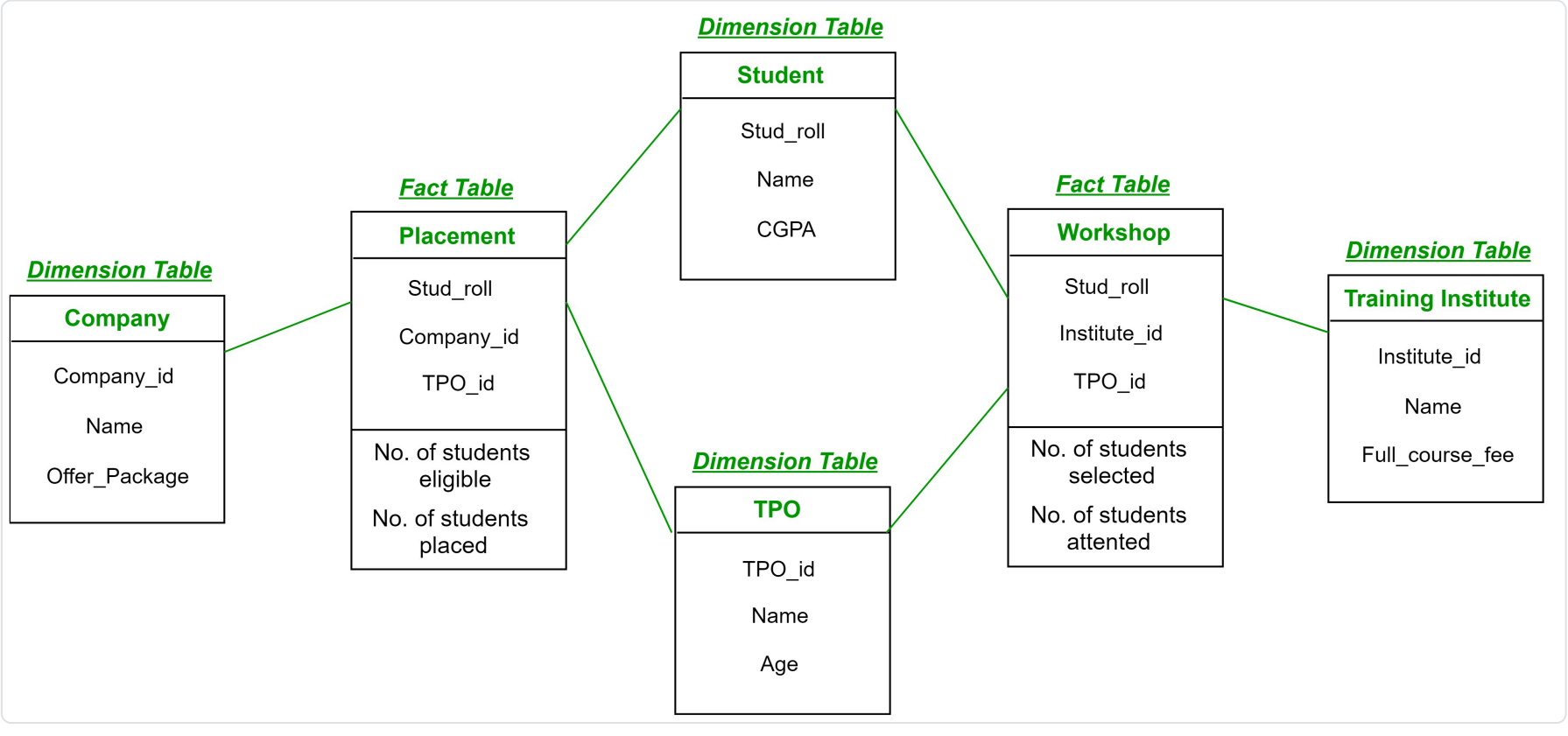
- **Definition:** A dimension that combines several low-cardinality attributes (flags or indicators) into a single dimension
- **Purpose:** Simplify the dimensional model by reducing the number of small dimensions
- **Example:** Combining order status, shipping method, and payment type into a single dimension

4. DEGENERATE DIMENSIONS

- **Definition:** Dimensional attributes stored in the fact table rather than in a separate dimension table
- **Purpose:** Improve query performance for attributes that are used primarily for grouping facts
- **Example:** Order number or transaction ID stored directly in the fact table

BEST PRACTICES

1. Design dimensions with the end-users' analytical needs in mind
2. Ensure dimension tables contain rich, descriptive attributes to support various types of analysis
3. Use meaningful, business-friendly names for dimension attributes
4. Regularly update slowly changing dimensions to maintain data accuracy
5. Document the structure and meaning of each dimension thoroughly



Example of dimensions

MEASURES

🕮 Numerical facts to be analyzed

- Types of measures:
  - Additive: Can be summed across all dimensions
  - Semi-additive: Can be summed across some dimensions
  - Non-additive: Cannot be summed meaningfully
- Derived measures and calculated members

Speaker notes

MEASURES IN DATA WAREHOUSING

DEFINITION

Measures are numerical facts to be analyzed in a data warehouse. They represent the quantitative data that users want to examine and analyze across various dimensions.

TYPES OF MEASURES

1. ADDITIVE MEASURES

- **Definition:** Measures that can be summed across all dimensions
- **Characteristics:**
    - Can be aggregated meaningfully along any dimension
    - Most common type of measure in data warehouses
- **Examples:** Sales amount, quantity sold, revenue

2. SEMI-ADDITIVE MEASURES

- **Definition:** Measures that can be summed across some dimensions, but not all
- **Characteristics:**
    - Often meaningful when aggregated over some dimensions (e.g., products) but not others (e.g., time)
    - Require careful consideration when aggregating
- **Examples:** Account balance, inventory levels

3. NON-ADDITIVE MEASURES

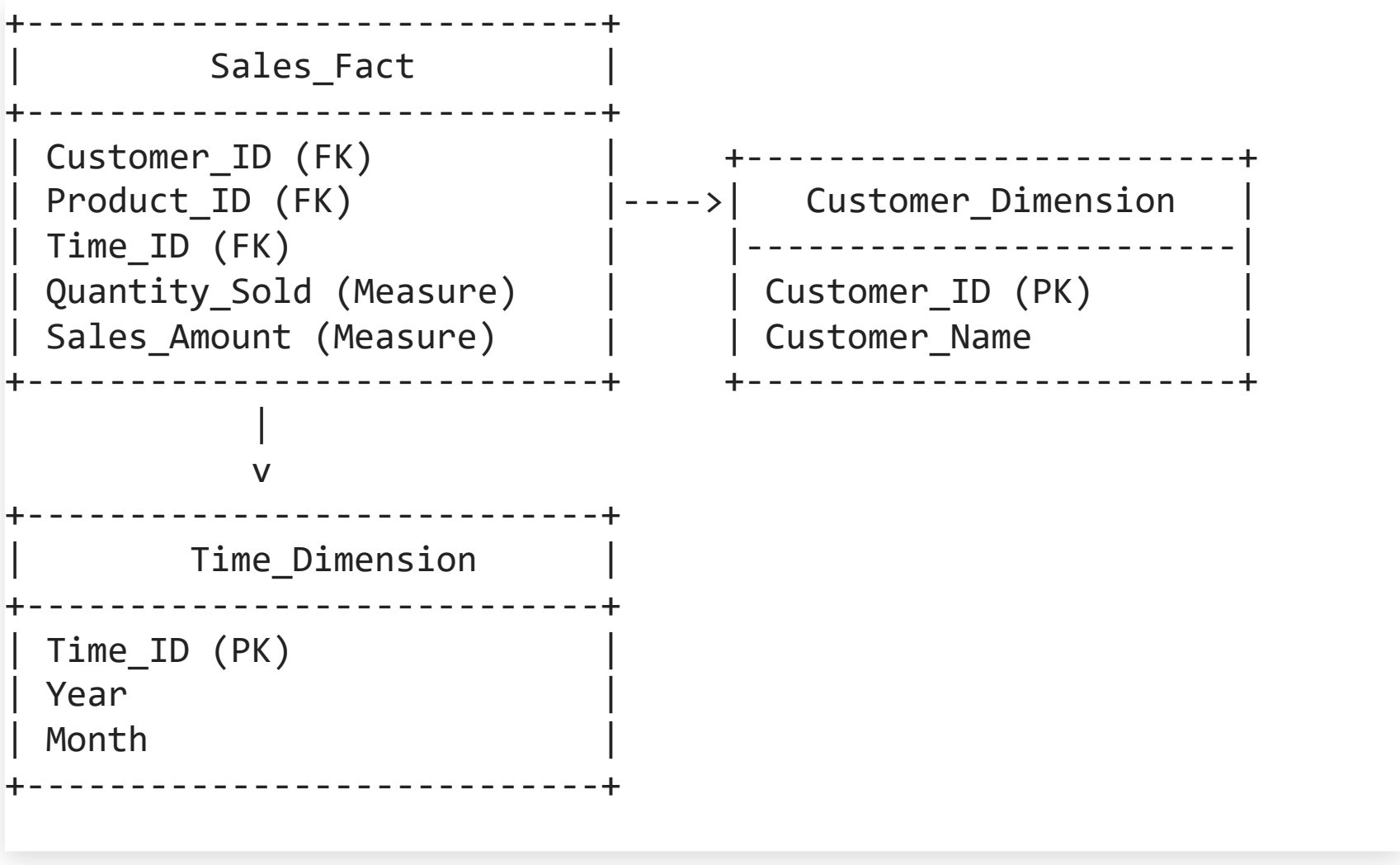
- **Definition:** Measures that cannot be summed meaningfully across any dimension
- **Characteristics:**
    - Often ratios, percentages, or averages
    - Require special handling in analysis and reporting
- **Examples:** Profit margin percentages, average prices, ratios

DERIVED MEASURES AND CALCULATED MEMBERS

- **Derived Measures:**
    - Measures calculated from other measures or dimensional attributes
    - Computed at query time rather than stored in the fact table
    - Example: Profit (derived from Revenue - Cost)
- **Calculated Members:**
    - Similar to derived measures but defined within the dimensional structure
    - Can involve complex calculations and business logic
    - Example: Year-to-date totals, moving averages

BEST PRACTICES

1. Clearly identify and document the type of each measure (additive, semi-additive, non-additive)
2. Design fact tables with primarily additive measures for optimal performance
3. Use appropriate aggregation methods for semi-additive and non-additive measures
4. Consider pre-calculating complex derived measures for performance reasons
5. Ensure calculated members are well-documented and understood by end-users
6. Regularly validate the accuracy of derived measures and calculated members


Sales_Fact
Customer_ID (FK)
Product_ID (FK)
Time_ID (FK)
Quantity_Sold (Measure)
Sales_Amount (Measure)
Customer_Dimension
Customer_ID (PK)
Customer_Name
Time_Dimension
Time_ID (PK)
Year
Month

Speaker notes

Explanation: In this diagram, the fact table is at the center with measures like Quantity_Sold and Sales_Amount which are the metrics being analyzed. The dimension tables (e.g., Customer_Dimension, Time_Dimension) provide context for those measures.

FACTS

📖 Collection of related data items, consisting of measures and context

- Types of fact tables:
  - Transaction fact tables
  - Periodic snapshot fact tables
  - Accumulating snapshot fact tables
- Granularity of facts
- Relationship between facts and dimensions

Speaker notes

FACTS IN DATA WAREHOUSING

DEFINITION

Facts are collections of related data items, consisting of measures and context. They represent the core data to be analyzed in a data warehouse, typically stored in fact tables.

TYPES OF FACT TABLES

1. TRANSACTION FACT TABLES

- **Description:** Represent individual transactions or events
- **Characteristics:**
    - Finest grain of detail
    - One row per transaction
    - Usually the most voluminous
- **Example:** Individual sales transactions, ATM withdrawals

2. PERIODIC SNAPSHOT FACT TABLES

- **Description:** Capture the state of things at regular, predetermined time intervals
- **Characteristics:**
    - Regular time intervals (e.g., daily, weekly, monthly)
    - Consistent level of aggregation over time
    - Good for analyzing trends over time
- **Example:** Monthly account balances, daily inventory levels

3. ACCUMULATING SNAPSHOT FACT TABLES

- **Description:** Track the progress of a process with a definite beginning and end
- **Characteristics:**
    - One row per process instance
    - Updated as the process progresses
    - Contains multiple date columns for different milestones
- **Example:** Order processing (order date, shipment date, delivery date)

GRANULARITY OF FACTS

- **Definition:** The level of detail represented by each row in a fact table
- **Importance:**
    - Determines the types of analyses that can be performed
    - Affects the size and performance of the data warehouse
- **Best Practice:** Choose the lowest level of granularity that is practical and meaningful for the business
- **Example:** Individual product sales vs. daily total sales by store

RELATIONSHIP BETWEEN FACTS AND DIMENSIONS

- **Structure:** Facts are typically surrounded by dimensions in a star or snowflake schema
- **Connections:**
    - Facts contain foreign keys that link to dimension tables
    - These links allow for rich, multi-dimensional analysis
- **Dimensionality:** The number of dimensions associated with a fact table determines its dimensionality
- **Analysis:** Dimensions provide the context for analyzing the measures in the fact table

BEST PRACTICES

1. Choose the appropriate fact table type based on business requirements and analysis needs
2. Determine the optimal granularity that balances detail with performance
3. Ensure consistency in the level of granularity across related fact tables
4. Design fact tables to be as narrow as possible, including only necessary columns
5. Use surrogate keys for dimension references to improve performance and handle changing dimension data
6. Document the meaning and context of each fact table thoroughly

THE CUBE CONCEPT

📖 Multi-dimensional representation of data

- Visualizing multi-dimensional data
- Basic operations:
  - Slicing
  - Dicing
  - Pivoting

Speaker notes

THE CUBE CONCEPT IN DATA WAREHOUSING

DEFINITION

The cube concept refers to a multi-dimensional representation of data in a data warehouse. It allows for the visualization and analysis of data across multiple dimensions simultaneously.

VISUALIZING MULTI-DIMENSIONAL DATA

- **Three-dimensional cube:** Often used to represent data with three dimensions (e.g., Product, Time, Location)
- **Hypercube:** Represents data with more than three dimensions
- **Cells:** Intersection points in the cube, containing measure values
- **Edges:** Represent dimensions (e.g., time, product, location)

BASIC OPERATIONS

1. SLICING

- **Definition:** Extracting a specific slice of the data cube by fixing one dimension
- **Example:** Analyzing sales for a specific month across all products and locations
- **Benefit:** Allows for focused analysis on a particular aspect of the data

2. DICING

- **Definition:** Extracting a sub-cube by fixing two or more dimensions
- **Example:** Analyzing sales for a specific product category in a particular region for the last quarter
- **Benefit:** Enables more granular analysis by focusing on multiple specific aspects simultaneously

3. PIVOTING

- **Definition:** Rotating the cube to view data from different perspectives
- **Also known as:** Rotation
- **Example:** Changing the view from "Product by Region" to "Region by Product"
- **Benefit:** Provides different analytical perspectives on the same dataset

ADDITIONAL IMPORTANT CONCEPTS

DRILL-DOWN AND ROLL-UP

- **Drill-down:** Moving from a higher level of aggregation to a more detailed level
- **Roll-up:** Aggregating data to a higher level in a dimension hierarchy
- **Example:** Drilling down from yearly sales to monthly sales, or rolling up from city-level data to country-level data

AGGREGATION

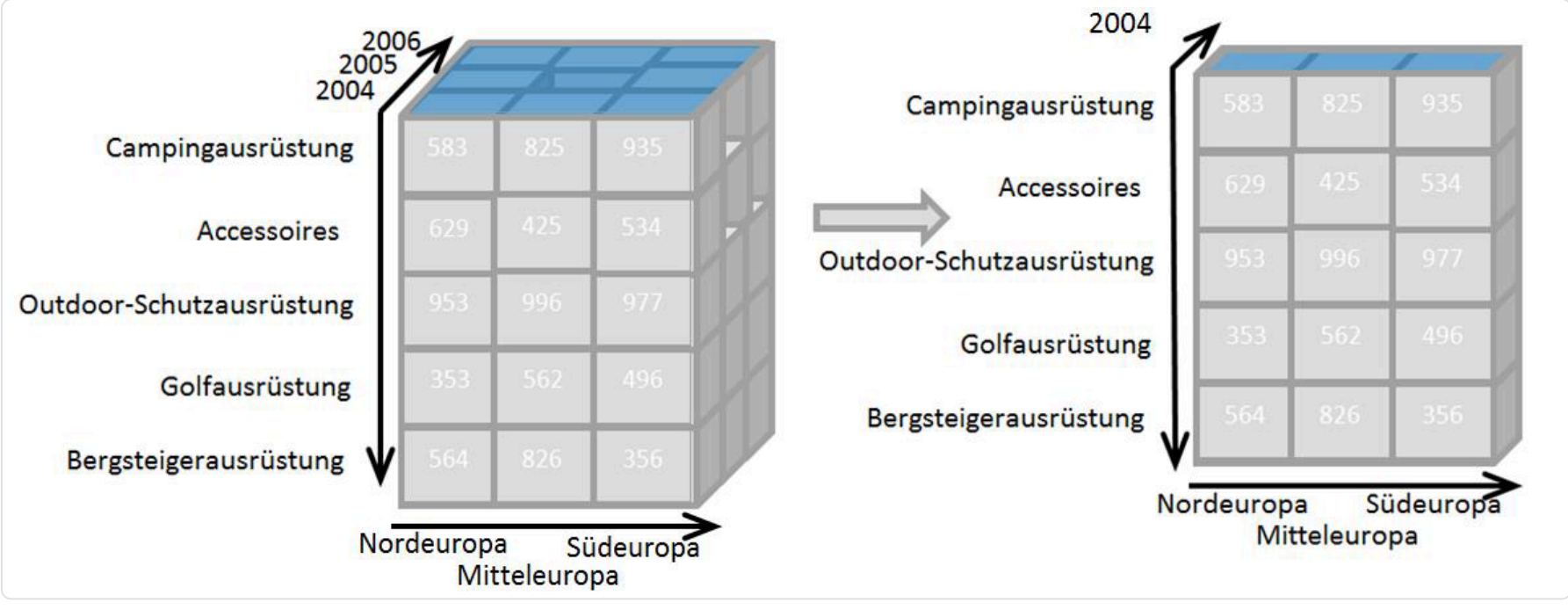
- Process of calculating summary values across dimensions
- Types include sum, average, count, min, max, etc.
- Essential for providing different levels of data granularity

BENEFITS OF THE CUBE CONCEPT

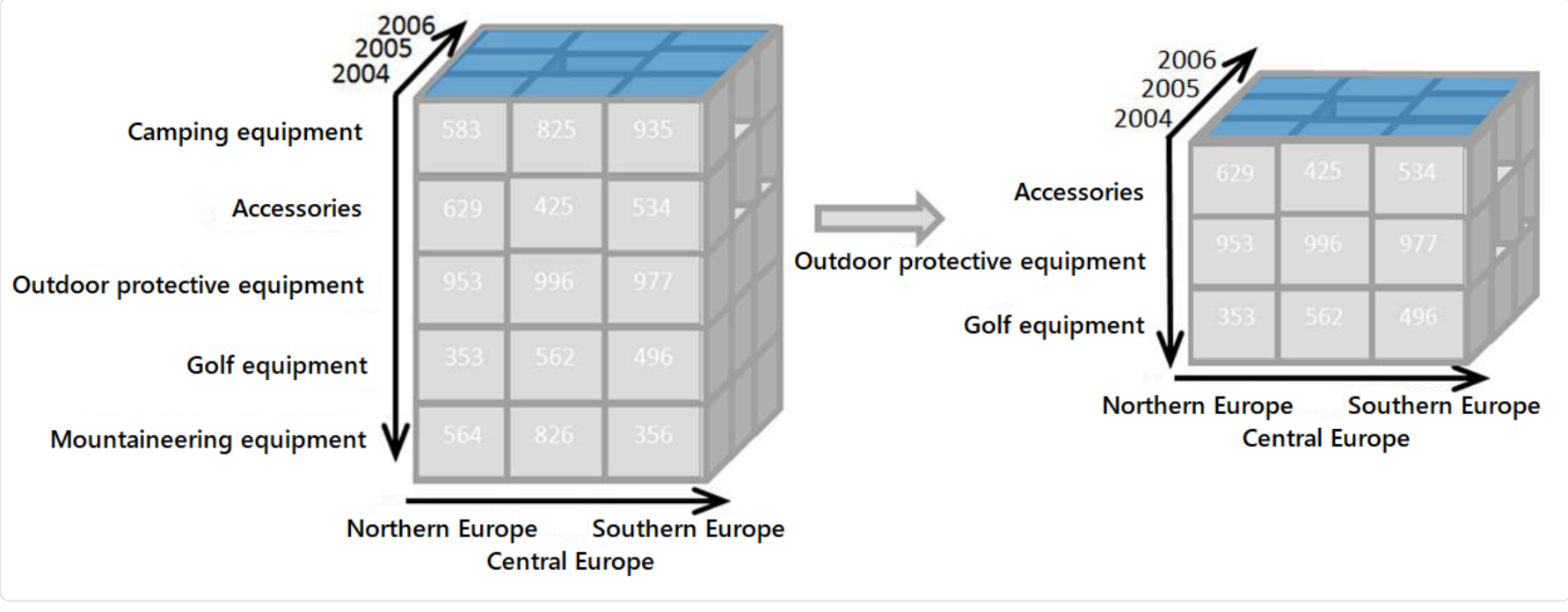
1. Enables intuitive representation of complex, multi-dimensional data
2. Facilitates quick and flexible data analysis
3. Supports various levels of data aggregation and detail
4. Allows for easy identification of trends, patterns, and anomalies
5. Enhances decision-making by providing multi-faceted views of business data

CHALLENGES AND CONSIDERATIONS

- Data sparsity: Many cells in a cube may be empty, leading to storage inefficiencies
- Performance: Large cubes with many dimensions can be computationally intensive
- Design complexity: Deciding on the right dimensions and hierarchies requires careful planning
- Data updates: Updating pre-aggregated data in cubes can be complex and time-consuming



Cube slicing


2006
2005
2004
Camping equipment
Accessories
Outdoor protective equipment
Golf equipment
Mountaineering equipment
583
825
935
629
425
534
953
996
977
353
562
496
564
826
356
Northern Europe
Southern Europe
Central Europe
Outdoor protective equipment
2006
2005
2004
Accessories
Golf equipment
629
425
534
953
996
977
353
562
496
Northern Europe
Southern Europe
Central Europe

Cube dicing

BENEFITS OF MULTI-DIMENSIONAL MODEL

- Intuitive data representation
- Efficient query performance
- Flexibility in analysis

MULTI-DIMENSIONAL VS. RELATIONAL MODEL

Aspect	Relational Model	Multi-dimensional Model
Primary Purpose	Operational processing (OLTP)	Analytical processing (OLAP)
Data Structure	Normalized tables	Denormalized, star or snowflake schema
Optimization	For data insertion and updates	For complex queries and aggregations
Query Complexity	Simple, predefined queries	Complex, ad-hoc queries
Data Redundancy	Minimized	Accepted for performance
Time Dimension	Usually represents current state	Historical data is a key aspect
Data Volume	Typically smaller	Usually much larger

IN-CLASS EXERCISE

A retail store chain tracks its sales data to analyze business performance. The store collects the following information for each sale:

- **Product Information**: Product name, Product category (e.g., electronics, clothing), Price
- **Customer Information**: Customer ID, Customer age group (e.g., 18-25, 26-35), Gender
- **Store Information**: Store location (city), Store region (e.g., North, South)
- **Sales Information**: Sale date, Quantity sold, Total sales amount (Quantity sold * Price)

Task:

DATA WAREHOUSE ARCHITECTURE AND COMPONENTS

SOURCE SYSTEMS

- Types of source systems:
  - OLTP databases
  - Flat files
  - External data sources
- Challenges in data extraction
  - Data quality issues
  - Heterogeneous data formats
  - Data volume and extraction frequency

Speaker notes

Source systems are the origin of data that flows into the data warehouse. Understanding these systems is crucial for effective data extraction and integration.

TYPES OF SOURCE SYSTEMS:

- **OLTP databases:** Operational databases that handle day-to-day transactions. These are typically normalized and optimized for quick updates and insertions.
- **Flat files:** Text files containing structured data, often used for data exchange between systems or for legacy data storage.
- **External data sources:** Third-party data providers, web services, APIs, or other external systems that provide valuable data for analysis.

Example: A retail company might have OLTP databases for point-of-sale transactions, flat files for historical sales data, and external data sources for market trends and weather information.

CHALLENGES IN DATA EXTRACTION:

- **Data quality issues:** Inconsistent formats, missing values, or incorrect data in source systems.
- **Heterogeneous data formats:** Different systems may use various data types, encodings, or structures.
- **Data volume and extraction frequency:** Balancing the need for up-to-date data with system performance and network constraints.

ETL PROCESS

Extract, Transform, Load

- **Extract:** Collect data from source systems (databases, flat files, etc.)
- **Transform:** Cleanse, filter, aggregate, and standardize the data
- **Load:** Move transformed data into the data warehouse or data marts
- Challenges in ETL: Handling large volumes, ensuring data consistency

Speaker notes

ETL Definition: ETL stands for Extract, Transform, Load. It is a crucial process in data warehousing that involves extracting data from various source systems, transforming it to fit the data warehouse's schema, and loading it into the target data warehouse.

1. EXTRACT

The first step in the ETL process is to extract data from various source systems.

KEY POINTS:

- Collect data from diverse source systems such as databases, flat files, APIs, and legacy systems.
- Determine the appropriate extraction method based on the source system and requirements.
- Handle different data formats and structures during extraction.

EXTRACTION METHODS:

- **Full Extraction:** Extracting all data from the source system each time.
- **Incremental Extraction:** Extracting only new or modified data since the last extraction.
- **Change Data Capture (CDC):** Identifying and capturing changes in the source data in real-time or near real-time.

Example: A retail company extracts daily sales data from its point-of-sale systems, customer information from its CRM database, and inventory data from its supply chain management system.

2. TRANSFORM

The transformation phase involves converting the extracted data into a format suitable for the data warehouse.

KEY TRANSFORMATION TASKS:

- **Cleansing:** Correcting or removing incorrect, incomplete, or improperly formatted data.
- **Filtering:** Selecting only the relevant data for the data warehouse.
- **Aggregating:** Summarizing data to reduce volume and improve query performance.
- **Standardizing:** Ensuring consistency in data formats, units, and representations across different sources.

COMMON TRANSFORMATIONS:

- Converting data types (e.g., string to date)
- Splitting or combining fields
- Encoding categorical data
- Calculating derived values
- Merging data from multiple sources

Example: Customer addresses from different systems are standardized to a common format, sales figures are converted to a single currency, and daily sales data is aggregated to weekly and monthly summaries.

3. LOAD

The final step involves loading the transformed data into the target data warehouse or data marts.

LOADING STRATEGIES:

- **Initial Load:** Populating the data warehouse for the first time with historical data.
- **Incremental Load:** Regularly updating the data warehouse with new or changed data.
- **Full Refresh:** Completely replacing existing data in the warehouse with a new dataset.

LOADING CONSIDERATIONS:

- Ensuring data integrity and consistency during the load process
- Managing the impact on query performance during loading
- Handling slowly changing dimensions
- Maintaining referential integrity in the data warehouse

Example: Transformed sales data is loaded into fact tables, while updated customer information is merged into the customer dimension table, handling any changes in customer attributes over time.

CHALLENGES IN ETL

ETL processes face several challenges, particularly when dealing with large-scale data warehouses.

1. HANDLING LARGE VOLUMES OF DATA:

- Processing and moving massive amounts of data efficiently
- Optimizing ETL jobs for performance
- Managing network bandwidth and storage requirements

2. ENSURING DATA CONSISTENCY:

- Maintaining data quality across diverse source systems
- Reconciling conflicting data from different sources
- Handling data type mismatches and inconsistencies

3. OTHER COMMON CHALLENGES:

- Meeting tight ETL windows and service level agreements (SLAs)
- Dealing with changing source systems and data structures
- Scaling ETL processes as data volumes grow
- Monitoring and error handling in complex ETL workflows

BEST PRACTICES IN ETL

- Design for scalability and performance from the start
- Implement robust error handling and logging mechanisms
- Use staging areas to minimize impact on source and target systems
- Automate ETL processes and schedule them appropriately
- Continuously monitor and optimize ETL jobs
- Maintain clear documentation of ETL processes and data lineage

DATA STAGING AREA

- Purpose and functions
  - Temporary storage for extracted data
  - Area for data transformation and cleansing
  - Improves overall data warehouse performance
- ETL processes in staging area:
  - Data extraction
  - Data cleansing
  - Data transformation

Speaker notes

- **Purpose**: The **data staging area** is a temporary workspace where raw data from various source systems is extracted, transformed, and prepared for loading into the data warehouse.
- **Key Functions**:
    - **Data extraction** from different sources (e.g., OLTP systems, flat files, external systems).
    - **Data cleansing** to remove inconsistencies, duplicates, and errors.
    - **Data transformation**, including filtering, aggregating, and standardizing the data to make it suitable for analysis.
    - This is where the **ETL (Extract, Transform, Load)** process begins: raw data is brought in, cleaned, and transformed before being loaded into the next layer.
- **Temporary Nature**: The data staging area usually holds data temporarily and may discard it once the data has been successfully transformed and loaded.

INTEGRATION LAYER

- Purpose: Consolidate and standardize data from different sources
- Data harmonization and application of business rules
- Prepares data for loading into the core data warehouse

Speaker notes

- **Purpose**: The **integration layer** consolidates and integrates data from different sources to create a unified, consistent dataset that is ready for analysis.
- **Key Functions**:
    - **Data consolidation**: Data from various sources is combined into a single, consistent view, eliminating redundancy and ensuring consistency across different data sources.
    - **Data harmonization**: Different formats, units of measurement, or terminology are standardized to create a cohesive dataset. For example, customer data from multiple sources might be merged to ensure each customer has a unique identifier across all systems.
    - **Business logic application**: Often, business rules are applied in this layer to ensure the data is structured according to the organization's requirements.
- **Permanent Nature**: The integration layer typically stores the **transformed and cleansed data** that is then used for querying and analysis. Data in this layer is integrated and consistent, but not yet fully optimized for end-user querying (that would happen in the presentation layer).

DATA STAGING AREA VS. INTEGRATION LAYER

Aspect	Data Staging Area	Integration Layer
Purpose	Temporary workspace for extracting and transforming raw data	Consolidates and integrates data for unified, consistent datasets
Functions	Extraction, cleansing, transformation, loading	Data harmonization, applying business logic, consolidation
Nature	Temporary data storage	Permanent, integrated data storage
Processing Stage	Early stage of ETL process (before transformation is complete)	Post-transformation, ready for querying or further loading
Persistence	Short-term; data is usually discarded after loading	Long-term; integrated data is stored for querying

CORE DATA WAREHOUSE

- Central repository characteristics
  - Integrated data from multiple sources
  - Historical and current data
  - Optimized for querying and analysis
- Data organization strategies:
  - Normalized approach: Reduces data redundancy, suitable for large-scale enterprise data warehouses
  - Dimensional approach: Optimized for query performance, suitable for specific business areas

Speaker notes

The core data warehouse is the central repository where integrated, historical data is stored for analysis and reporting.

CENTRAL REPOSITORY CHARACTERISTICS:

- **Integrated data from multiple sources:** Provides a single version of truth for the entire organization.
- **Historical and current data:** Enables trend analysis and comparisons over time.
- **Optimized for querying and analysis:** Designed for complex queries and large data volumes, often using columnar storage or other performance-enhancing techniques.

DATA ORGANIZATION STRATEGIES:

- **Normalized approach:** Reduces data redundancy, suitable for large-scale enterprise data warehouses. Uses normal forms to organize data, which can be beneficial for data integrity but may require more complex queries.
- **Dimensional approach:** Optimized for query performance, suitable for specific business areas. Uses fact and dimension tables in a star or snowflake schema, which simplifies queries and improves query performance.

Example: A healthcare data warehouse might use a normalized approach for patient records to ensure data integrity, while using a dimensional approach for analyzing treatment outcomes across different demographics and time periods.

DATA MARTS

🕮 Subset of data warehouse focused on specific business area

- Types:
  - Dependent data marts: Derived from the central data warehouse
  - Independent data marts: Built directly from source systems
- Relationship with the core data warehouse
  - Can serve as a layer between the core warehouse and end-users
  - Provides tailored data for specific departments or functions

Speaker notes

A data mart is a subset of the data warehouse focused on a specific business area or department.

TYPES:

- **Dependent data marts:** Derived from the central data warehouse, ensuring consistency with other data marts and the core warehouse.
- **Independent data marts:** Built directly from source systems, often used when a full data warehouse is not feasible or when quick results are needed for a specific business area.

RELATIONSHIP WITH THE CORE DATA WAREHOUSE:

- **Layer between core warehouse and end-users:** Provides tailored, often aggregated data for specific departments or functions, improving query performance and usability.
- **Tailored data for specific needs:** Allows for customized data structures and aggregations that suit particular business requirements.

Example: A large retailer might have separate data marts for inventory management, customer relationship management, and financial analysis, each derived from the central data warehouse but optimized for its specific use case.

METADATA REPOSITORY

- Types of metadata:
  - Business metadata: Definitions, ownership, and usage of data
  - Technical metadata: Data structures, ETL mappings, and database schemas
  - Operational metadata: ETL job logs, data lineage, and usage statistics
- Importance of metadata management
  - Facilitates data governance and compliance
  - Improves data understanding and usability
  - Aids in impact analysis and change management

Speaker notes

The metadata repository stores information about the data warehouse itself, crucial for understanding, managing, and using the data effectively.

TYPES OF METADATA:

- **Business metadata:** Definitions, ownership, and usage of data. Includes business terms, KPI definitions, and data lineage from a business perspective.
- **Technical metadata:** Data structures, ETL mappings, and database schemas. Includes information about data types, table relationships, and transformation rules.
- **Operational metadata:** ETL job logs, data lineage, and usage statistics. Provides information about data freshness, processing times, and user access patterns.

IMPORTANCE OF METADATA MANAGEMENT:

- **Facilitates data governance and compliance:** Helps in tracking data origins, transformations, and usage, crucial for regulatory compliance and data privacy.
- **Improves data understanding and usability:** Enables users to understand the meaning and context of data, improving analysis accuracy.
- **Aids in impact analysis and change management:** Allows administrators to understand dependencies and potential impacts of changes to the data warehouse structure.

Example: In a financial data warehouse, the metadata repository might contain definitions of financial terms, the formulas used to calculate various ratios, the sources of different financial data points, and logs of when and how often specific reports are accessed.

FRONT-END APPLICATIONS

- Reporting tools
- OLAP tools
- Data mining applications
- Dashboards and scorecards

Speaker notes

Front-end applications are the tools and interfaces that users interact with to access and analyze data from the warehouse.

- **Reporting tools:** For creating standard and ad-hoc reports. These range from simple tabular reports to complex, interactive dashboards.
- **OLAP tools:** For multi-dimensional analysis and data exploration. Allow users to slice and dice data, drill down into details, or roll up to higher levels of aggregation.
- **Data mining applications:** For discovering patterns and insights in data. Use advanced statistical and machine learning techniques to uncover hidden trends and relationships.
- **Dashboards and scorecards:** For visualizing key performance indicators. Provide at-a-glance views of important metrics and trends, often with interactive elements.

Example: A marketing team might use a dashboard to track campaign performance, an OLAP tool to analyze customer segments, and a data mining application to predict customer churn based on historical data.

ARCHITECTURAL APPROACHES

- Inmon's approach (top-down):
  - Enterprise-wide data warehouse: Centralized repository for all organizational data
  - Normalized data model: Reduces data redundancy and ensures data integrity
- Kimball's approach (bottom-up):
  - Series of integrated data marts: Built incrementally based on business priorities
  - Dimensional model: Optimized for query performance and ease of use
- Hybrid approaches

Speaker notes

Different architectural approaches to data warehousing have evolved to meet various business needs and constraints.

INMON'S APPROACH (TOP-DOWN):

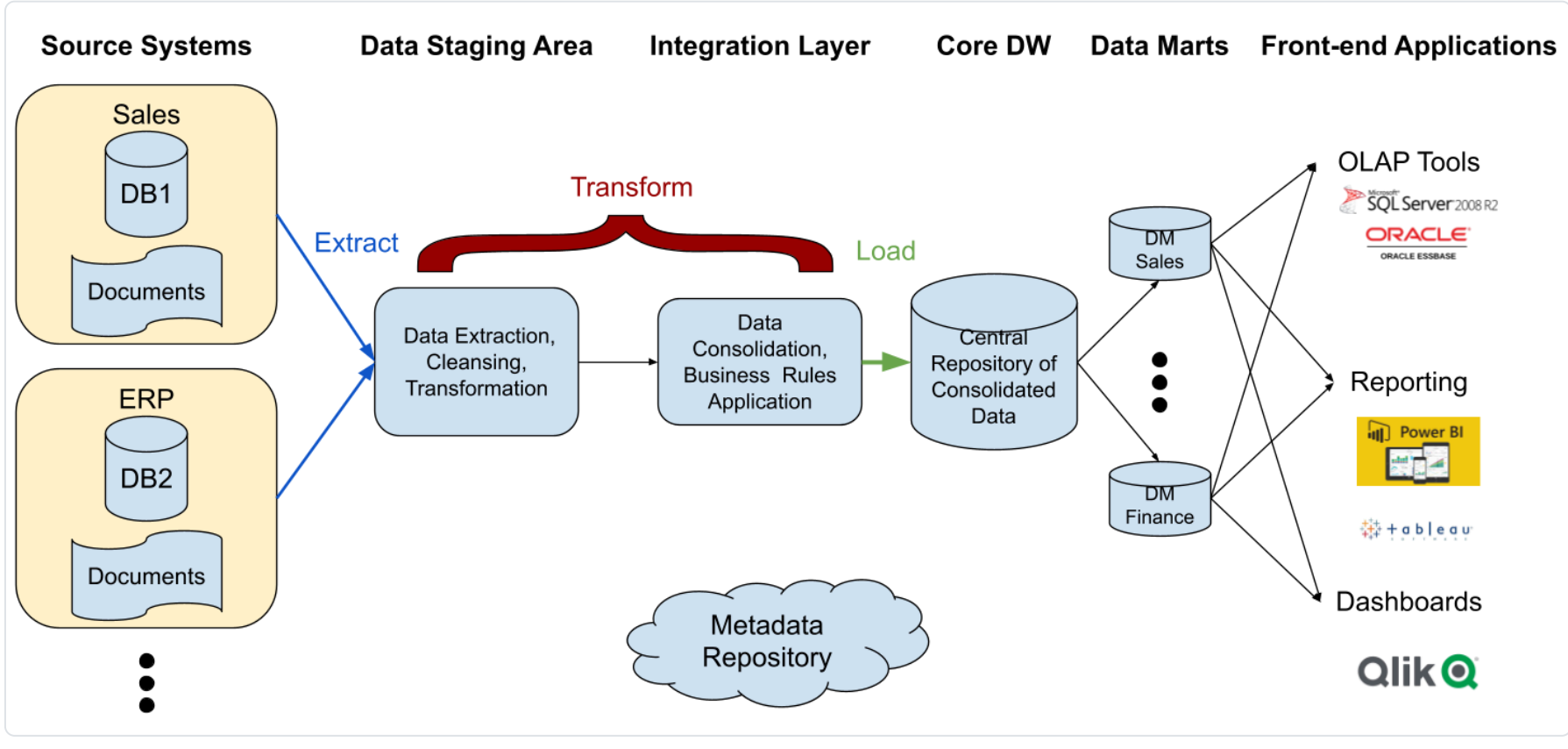
- **Enterprise-wide data warehouse:** Centralized repository for all organizational data, providing a single source of truth.
- **Normalized data model:** Reduces data redundancy and ensures data integrity, but may require more complex queries for analysis.

KIMBALL'S APPROACH (BOTTOM-UP):

- **Series of integrated data marts:** Built incrementally based on business priorities, allowing for faster time to value.
- **Dimensional model:** Optimized for query performance and ease of use, using star or snowflake schemas.

HYBRID APPROACHES:

- **Combines elements of both Inmon and Kimball approaches:** May use a normalized core with dimensional data marts, or start with data marts and gradually build towards an enterprise warehouse.
- **Adapts to specific organizational needs and constraints:** Allows for flexibility in implementation while striving for long-term integration and consistency.



Data Warehouse Architecture

MULTI-DIMENSIONAL QUERIES: OLAP OPERATIONS

INTRODUCTION TO OLAP

📖 Online Analytical Processing

🎯 Purpose: Enable complex analytical queries and data exploration

- OLAP vs. OLTP:
  - Query complexity
  - Data volume
  - User types

Speaker notes

OLAP (Online Analytical Processing) enables users to perform complex, multi-dimensional queries for analytical purposes. While OLTP systems focus on processing day-to-day transactions, OLAP systems are designed for querying and reporting on large datasets. Key differences include:

- **Query complexity:** OLAP queries often involve aggregations and multidimensional analysis, whereas OLTP queries are simpler and transactional.
- **Data volume:** OLAP systems handle large datasets, often historical, while OLTP works with smaller, real-time transactional data.
- **User types:** OLAP is typically used by analysts for decision-making, while OLTP is used by clerks or operational staff.

Example: A financial analyst might use OLAP to analyze sales data by product, region, and time, identifying trends and patterns across multiple dimensions.

BASIC OLAP OPERATIONS

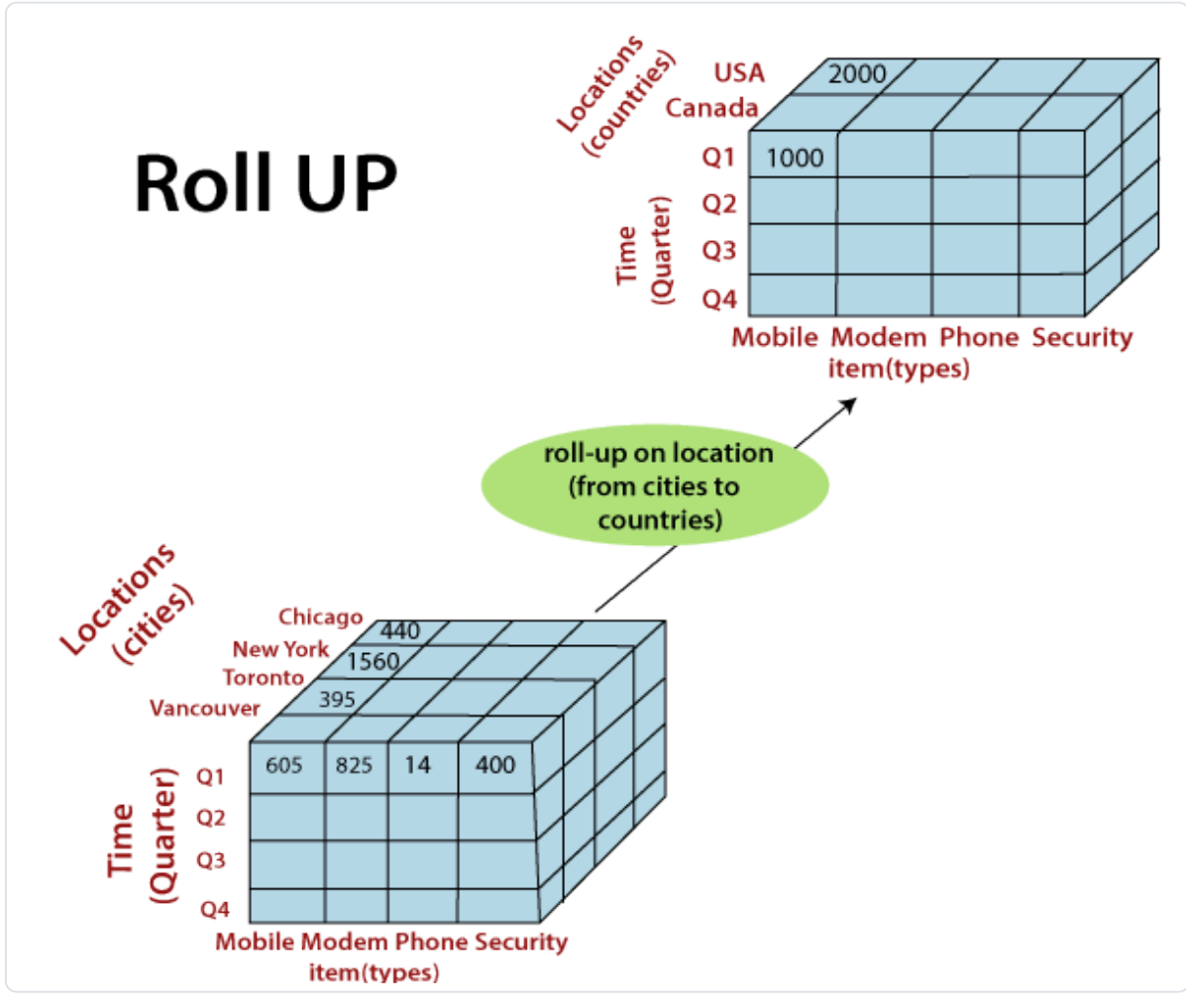
- Roll-up (drill-up): Aggregating data to a higher level
- Drill-down: Navigating to more detailed data
- Slice: Selecting a specific dimension value
- Dice: Selecting values from multiple dimensions
- Pivot (rotate): Changing the dimensional orientation

Speaker notes

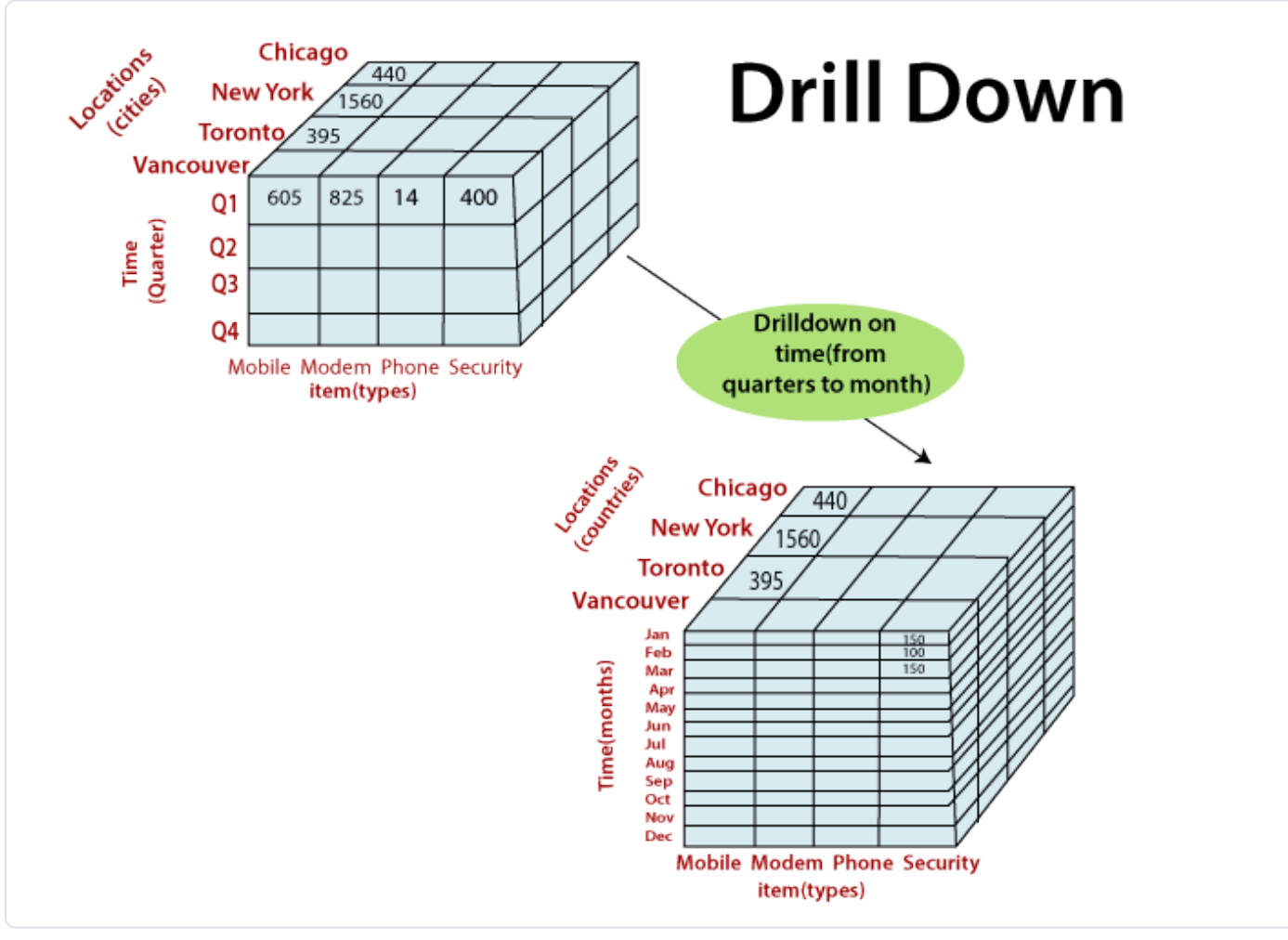
OLAP operations allow users to explore and analyze data from different perspectives:

- **Roll-up (drill-up):** Aggregates data to a higher level (e.g., viewing total sales by region instead of by city).
- **Drill-down:** Enables users to view more detailed data (e.g., drilling down from sales by region to sales by individual stores).
- **Slice:** Selects a single dimension value, such as filtering data to only show sales for 2023.
- **Dice:** Selects multiple dimensions (e.g., viewing sales for 2023 and Region A).
- **Pivot (rotate):** Changes the orientation of the data to look at it from a different perspective (e.g., swapping rows and columns).

Example: A retail company might drill down from total sales to see product-level sales, then slice the data by year or region.



Roll-up (from javatpoint.com)



Drill-down (from javatpoint.com)

ADVANCED OLAP CONCEPTS

- Drill-across: Combining data from different fact tables
- Drill-through: Accessing detailed source data
- Ranking and windowing functions

Speaker notes

Advanced OLAP concepts provide even greater flexibility for data analysis:

- **Drill-across:** Combines data from different fact tables (e.g., sales and inventory data).
- **Drill-through:** Allows access to detailed source data, often at the transaction level (e.g., viewing the original sales transaction records).
- **Ranking and windowing functions:** These functions help with ordering data, calculating running totals, and other advanced analyses.

Example: An analyst might drill across to compare sales data with inventory levels or use ranking to find the top-selling products.

OLAP SCHEMAS

- ROLAP (Relational OLAP): Using relational databases
- MOLAP (Multidimensional OLAP): Using specialized multidimensional databases
- HOLAP (Hybrid OLAP): Combining ROLAP and MOLAP

Speaker notes

There are three main OLAP architectures, each with its strengths and weaknesses:

- **ROLAP (Relational OLAP):** Uses a relational database to store and query data. It can handle large volumes but may suffer from slower performance due to reliance on SQL.
- **MOLAP (Multidimensional OLAP):** Stores data in a multidimensional cube, enabling faster query performance but requiring more storage space.
- **HOLAP (Hybrid OLAP):** Combines ROLAP and MOLAP, using relational databases for detailed data and cubes for aggregated data.

Example: A company might use ROLAP for large historical data storage and MOLAP for fast analysis of key metrics.

QUERYING MULTI-DIMENSIONAL DATA

- MDX (Multidimensional Expressions)
-

```
SELECT
  { [Measures].[Store Sales] } ON COLUMNS,
  { [Date].[2002], [Date].[2003] } ON ROWS
FROM Sales
WHERE ( [Store].[USA].[CA] )
```

Speaker notes

MDX (Multidimensional Expressions) is a language designed specifically for querying OLAP cubes, offering greater flexibility for multidimensional data:

- **MDX:** Used for querying and manipulating OLAP data, particularly in multidimensional databases.
- **Comparison with SQL:** While SQL is designed for two-dimensional relational databases, MDX is optimized for querying multi-dimensional data.

Example: An MDX query might return sales data sliced by year and product category, offering more flexibility than an equivalent SQL query.

In this example, the query defines the following result set information

- The SELECT clause sets the query axes as the Store Sales member of the Measures dimension, and the 2002 and 2003 members of the Date dimension.
- The FROM clause indicates that the data source is the Sales cube.
- The WHERE clause defines the "slicer axis" as the California member of the Store dimension.

RELATIONAL SCHEMAS FOR DATA WAREHOUSES

STAR SCHEMA

🕮 Fact table surrounded by dimension tables

- Structure: Centralized fact table with foreign keys to dimension tables
- Benefits:
  - Simplified queries
  - Improved query performance
  - Easier to understand and navigate

Speaker notes

In data warehousing, the choice of relational schema is crucial as it impacts query performance, data integrity, and overall maintainability. The three main types of schemas used in data warehouses are Star Schema, Snowflake Schema, and Fact Constellation Schema.

1. STAR SCHEMA

A star schema consists of a central fact table surrounded by dimension tables, resembling a star-like structure.

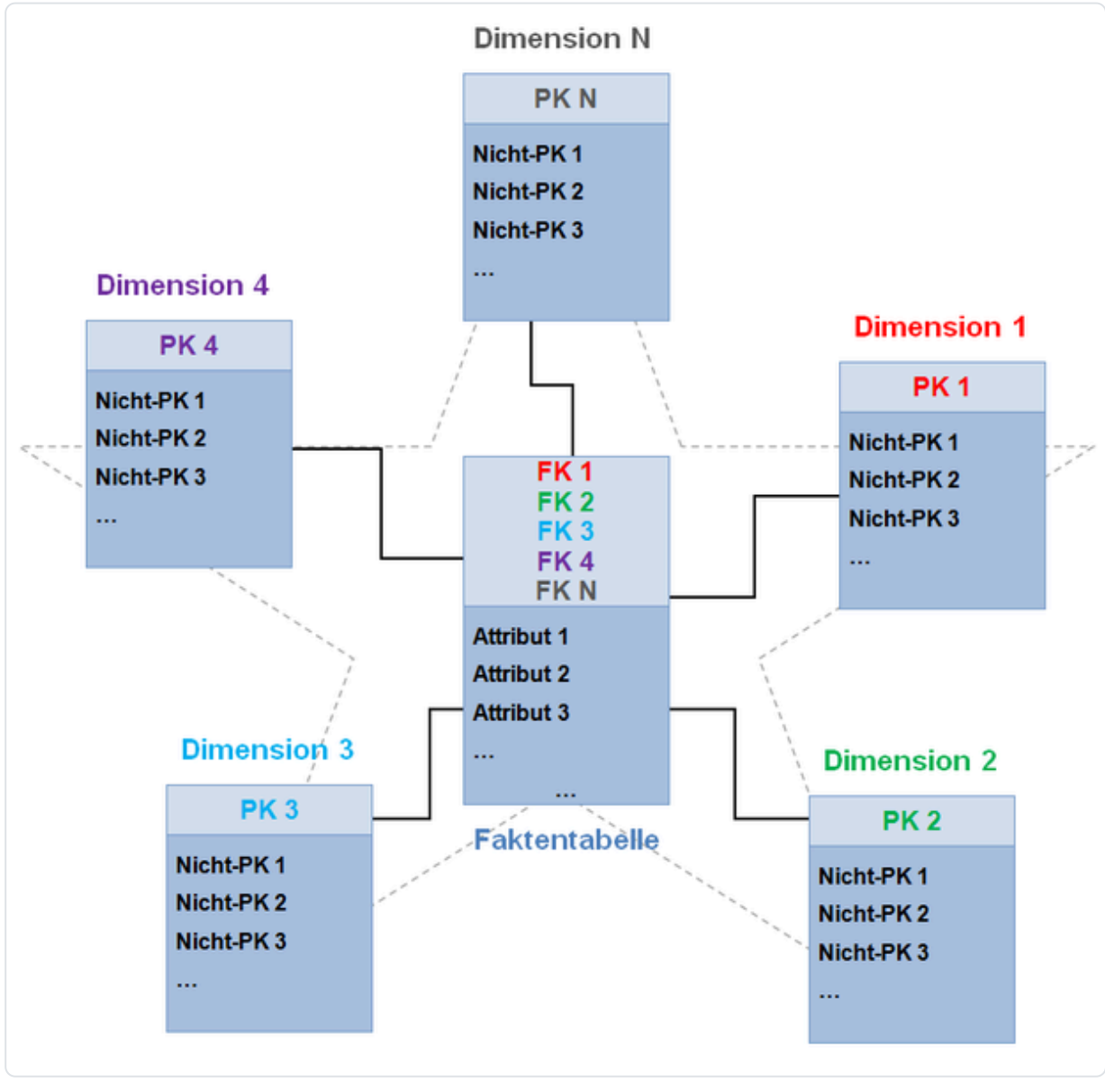
STRUCTURE:

- Centralized fact table containing business metrics (facts)
- Dimension tables connected to the fact table via foreign keys
- Dimension tables are denormalized (not normalized)

BENEFITS:

- **Simplified queries:** Fewer joins required due to denormalized dimension tables
- **Improved query performance:** Faster data retrieval, especially for aggregations
- **Easier to understand and navigate:** Intuitive structure for business users

Example: In a retail data warehouse, a Sales fact table might be surrounded by dimension tables like Date, Product, Store, and Customer.


Dimension N
PK N
Nicht-PK 1
Nicht-PK 2
Nicht-PK 3
...
Dimension 4
PK 4
Nicht-PK 1
Nicht-PK 2
Nicht-PK 3
...
Dimension 1
PK 1
Nicht-PK 1
Nicht-PK 2
Nicht-PK 3
...
FK 1
FK 2
FK 3
FK 4
FK N
Attribut 1
Attribut 2
Attribut 3
...
...
Faktentabelle
Dimension 3
PK 3
Nicht-PK 1
Nicht-PK 2
Nicht-PK 3
...
Dimension 2
PK 2
Nicht-PK 1
Nicht-PK 2
Nicht-PK 3
...

Star schema

SNOWFLAKE SCHEMA

🕮 Extension of star schema with normalized dimensions

- Structure: Dimension tables are normalized into multiple related tables
- Comparison with star schema:
  - Reduced data redundancy
  - More complex queries
  - Potentially slower query performance
- When to use snowflake schema
  - When data integrity is a top priority
  - When storage space is a concern

Speaker notes

2. SNOWFLAKE SCHEMA

A snowflake schema is an extension of the star schema where dimension tables are normalized into multiple related tables.

STRUCTURE:

- Central fact table, similar to star schema
- Dimension tables are normalized, forming a branching structure

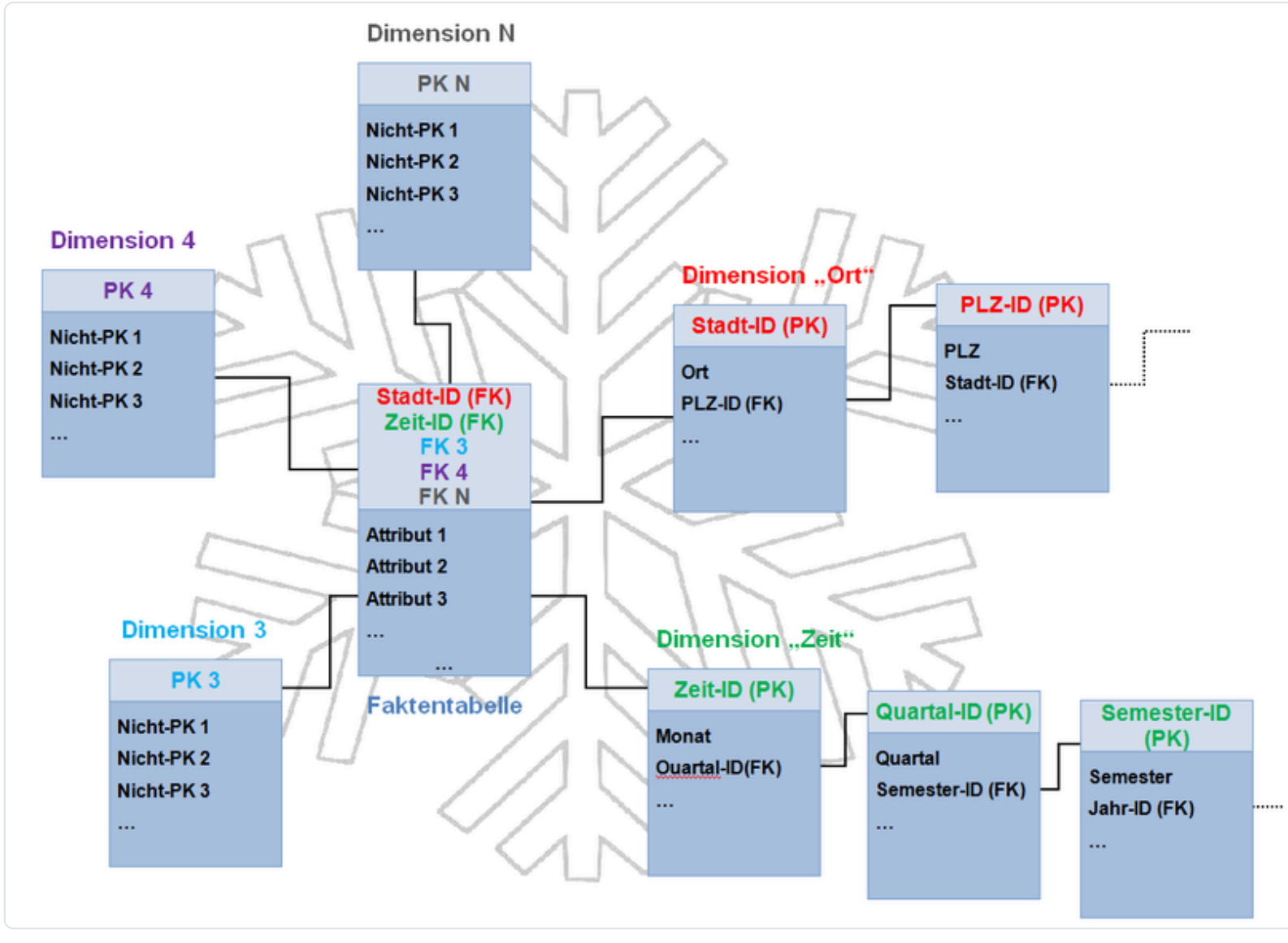
COMPARISON WITH STAR SCHEMA:

- **Reduced data redundancy:** Normalization eliminates duplicate data
- **More complex queries:** Requires additional joins due to normalized dimensions
- **Potentially slower query performance:** More joins can impact query speed

WHEN TO USE SNOWFLAKE SCHEMA:

- **Data integrity is a top priority:** Normalization helps maintain data consistency
- **Storage space is a concern:** Reduces storage requirements through normalization

Example: In a product dimension, attributes like category and subcategory might be split into separate tables, branching out from the main product table.



Snowflake schema

FACT CONSTELLATION SCHEMA

🕮 Multiple fact tables sharing dimension tables

- Structure: Multiple star schemas with shared dimensions
- Use cases and challenges
  - Complex business environments
  - Maintaining consistency across shared dimensions
  - Increased complexity in query design

Speaker notes

3. FACT CONSTELLATION SCHEMA

A fact constellation schema consists of multiple fact tables sharing dimension tables, forming a more complex structure.

STRUCTURE:

- Multiple fact tables, each representing different business processes
- Shared dimension tables between fact tables

USE CASES AND CHALLENGES:

- **Use cases:** Complex business environments with multiple related business processes
- **Challenges:**
    - Maintaining consistency across shared dimensions
    - Increased complexity in query design and optimization
    - Potentially more difficult for end-users to navigate

Example: A retail data warehouse might have separate fact tables for Sales and Inventory, both sharing dimensions like Product and Store.

- Star Schema: Centralized fact table with denormalized dimensions
- Snowflake Schema: Normalized dimensions branching out from the fact table
- Fact Constellation Schema: Multiple fact tables sharing dimensions

Key Differences Normalization: Snowflake schema is more normalized than galaxy schema. Complexity: Snowflake schema is generally more complex to design and implement. Performance: The performance of both schemas depends on the specific query patterns and data volumes. In some cases, snowflake schema can offer better performance for queries involving multiple levels of

dimension hierarchies. When to Use Which Schema

- Galaxy Schema:
    - Simple data models with limited dimension hierarchies.
    - Performance is critical for queries involving a single fact table and multiple dimensions.
- Snowflake Schema:
    - Complex data models with multiple levels of dimension hierarchies.
    - Data normalization is important.
    - Performance is critical for queries involving multiple levels of dimension hierarchies.

CHOOSING THE RIGHT SCHEMA

- Factors to consider:
  - Query performance requirements
  - Data volume
  - Maintenance complexity
- Performance implications
- Maintainability and flexibility

Speaker notes

CHOOSING THE RIGHT SCHEMA

Selecting the appropriate schema depends on various factors and requires careful consideration of trade-offs.

FACTORS TO CONSIDER:

- **Query performance requirements:** How critical is query speed for your use case?
- **Data volume:** How much data will the warehouse store and process?
- **Maintenance complexity:** How often will the schema need updates, and who will maintain it?

PERFORMANCE IMPLICATIONS:

- Star schemas generally offer the best query performance due to fewer joins
- Snowflake schemas may have slower queries but can be more space-efficient
- Fact constellation schemas can be optimized for specific queries but may be complex to manage

MAINTAINABILITY AND FLEXIBILITY:

- Star schemas are easier to maintain and modify
- Snowflake schemas offer better data integrity but are more complex to update
- Fact constellation schemas provide flexibility for complex business scenarios but require careful management

DENORMALIZATION IN DW SCHEMAS

🎯 Purpose: Improve query performance

- Benefits:
  - Reduced number of joins
  - Faster query execution
- Potential drawbacks:
  - Data redundancy
  - Increased storage requirements
  - More complex data updates

Speaker notes

DENORMALIZATION IN DW SCHEMAS

Denormalization is a technique often used in data warehouse design to improve query performance at the cost of some data redundancy.

PURPOSE:

- Improve query performance by reducing the need for joins

BENEFITS:

- **Reduced number of joins:** Fewer tables to connect in queries
- **Faster query execution:** Simpler queries often translate to quicker results

POTENTIAL DRAWBACKS:

- **Data redundancy:** Same information may be stored in multiple places
- **Increased storage requirements:** Redundant data consumes more space
- **More complex data updates:** Changes must be propagated to all instances of redundant data

Example: In a star schema, a Product dimension might include category information directly, even though this creates some redundancy, to avoid an additional join to a separate Category table.

DIMENSIONAL MODELING

INTRODUCTION TO DIMENSIONAL MODELING

📖 Technique for designing logical data models for data warehouses

🎯 Purpose: Optimize database structures for analytical queries

- Kimball's approach to dimensional modeling

Speaker notes

Dimensional Modeling is a technique for designing logical data models for data warehouses, optimizing database structures for analytical queries.

- **Purpose:** To create a database structure that is intuitive for business users and optimized for query performance in data warehouses and business intelligence systems.
- **Kimball's approach:** Focuses on creating a user-centric, bottom-up design that starts with the most critical business processes and expands over time.

STEPS IN DIMENSIONAL MODELING

1. Choose the business process
2. Declare the grain (level of detail)
3. Identify the dimensions
4. Identify the facts

Speaker notes

1. **Choose the business process:** Identify the specific business activity to model (e.g., sales transactions, customer support tickets).
2. **Declare the grain:** Determine the level of detail for the fact table (e.g., individual sales transactions, daily sales summaries).
3. **Identify the dimensions:** Determine the contextual attributes that describe the facts (e.g., date, product, customer, store).
4. **Identify the facts:** Define the numerical measures that will be analyzed (e.g., quantity sold, sales amount, profit).

FACT TABLES

- Types of fact tables:
  - Transaction fact tables
  - Periodic snapshot fact tables
  - Accumulating snapshot fact tables
- Selecting appropriate measures
- Handling multiple grains in a single fact table

Speaker notes

TYPES OF FACT TABLES:

- **Transaction fact tables:** Represent individual business events (e.g., individual sales transactions).
- **Periodic snapshot fact tables:** Capture the state of things at regular time intervals (e.g., daily inventory levels).
- **Accumulating snapshot fact tables:** Track the progress of a process with a definite beginning and end (e.g., order fulfillment process).

SELECTING APPROPRIATE MEASURES:

- Choose measures that are relevant to the business process and align with analytical needs.
- Ensure measures are consistent with the declared grain.
- Consider both additive and non-additive measures.

HANDLING MULTIPLE GRAINS IN A SINGLE FACT TABLE:

- Use the most granular level that makes sense for the business process.
- Consider creating separate fact tables for different grains if necessary.
- Use aggregation tables or materialized views for commonly requested summary levels.

DIMENSION TABLES

- Role of dimension tables: Provide context to facts
- Slowly Changing Dimensions (SCD):
  - Type 1: Overwrite
  - Type 2: Add new row
  - Type 3: Add new attribute
- Handling hierarchies in dimensions

Speaker notes

Role of dimension tables: Provide context to facts, serving as the primary source of query constraints and report labels.

SLOWLY CHANGING DIMENSIONS (SCD):

- **Type 1 (Overwrite):** Update the dimension table, overwriting the old value with the new one. No history is kept.
- **Type 2 (Add new row):** Add a new row with the changed data, keeping historical records. Requires additional columns like effective date and current flag.
- **Type 3 (Add new attribute):** Add a new column to track changes, typically for a specific attribute where limited history is needed.

HANDLING HIERARCHIES IN DIMENSIONS:

- Represent natural hierarchies within a single dimension table (e.g., Product Category > Product Subcategory > Product).
- Use separate dimension tables for complex or variable hierarchies.
- Consider using bridge tables for ragged or unbalanced hierarchies.

HANDLING COMPLEX SCENARIOS

- Many-to-many relationships:
  - Bridge tables
  - Factless fact tables
- Handling ragged hierarchies:
  - Bridge tables
  - Nested sets model
- Dealing with sparse facts:
  - Separate fact tables
  - Bitmap indexing

Speaker notes

MANY-TO-MANY RELATIONSHIPS:

- **Bridge tables:** Use to connect facts to multiple dimension instances (e.g., products in a sales order).
- **Factless fact tables:** Represent relationships or events without measures (e.g., student course enrollments).

HANDLING RAGGED HIERARCHIES:

- **Bridge tables:** Use to represent variable-depth hierarchies.
- **Nested sets model:** An alternative approach for efficient querying of hierarchical data.

DEALING WITH SPARSE FACTS:

- **Separate fact tables:** Create different fact tables for dense and sparse facts.
- **Bitmap indexing:** Use bitmap indexes to efficiently handle sparse data in large fact tables.

BEST PRACTICES IN DIMENSIONAL MODELING

- Choosing the right grain:
  - Balance between detail and performance
  - Consider business requirements
- Handling multi-valued dimensions:
  - Bridge tables
  - Flattening the structure

Speaker notes

CHOOSING THE RIGHT GRAIN:

- Balance between detail and performance: Choose a grain that provides necessary detail without overwhelming the system.
- Consider business requirements: Ensure the chosen grain supports all required types of analysis.

HANDLING MULTI-VALUED DIMENSIONS:

- **Bridge tables:** Use for many-to-many relationships between facts and dimensions.
- **Flattening the structure:** Denormalize multi-valued attributes into the fact table when appropriate.

ENSURING CONSISTENCY ACROSS THE ENTERPRISE:

- **Conformed dimensions:** Use standardized dimensions across different fact tables and data marts.
- **Standard naming conventions:** Implement consistent naming for tables, columns, and attributes.

PERFORMANCE CONSIDERATIONS:

- **Appropriate indexing:** Use indexes on commonly queried columns in both fact and dimension tables.
- **Partitioning strategies:** Implement table partitioning for large fact tables to improve query performance.

CONCLUSION AND PREVIEW

RECAP OF KEY POINTS

- Multi-dimensional model concepts
- DW architecture and components
- OLAP operations
- Relational schemas for DWHs
- Dimensional modeling principles

Q&A