CLASS 1 - 07-06-25

DATA WAREHOUSE COMPREHENSIVE STUDY GUIDE

TABLE OF CONTENTS

---

1. DATA WAREHOUSE: WHAT, WHY AND HOW?

Definition

A Data Warehouse is a concept and process first coined by **Bill Inmon in 1990**. According to Inmon, it is a collection of data that is:

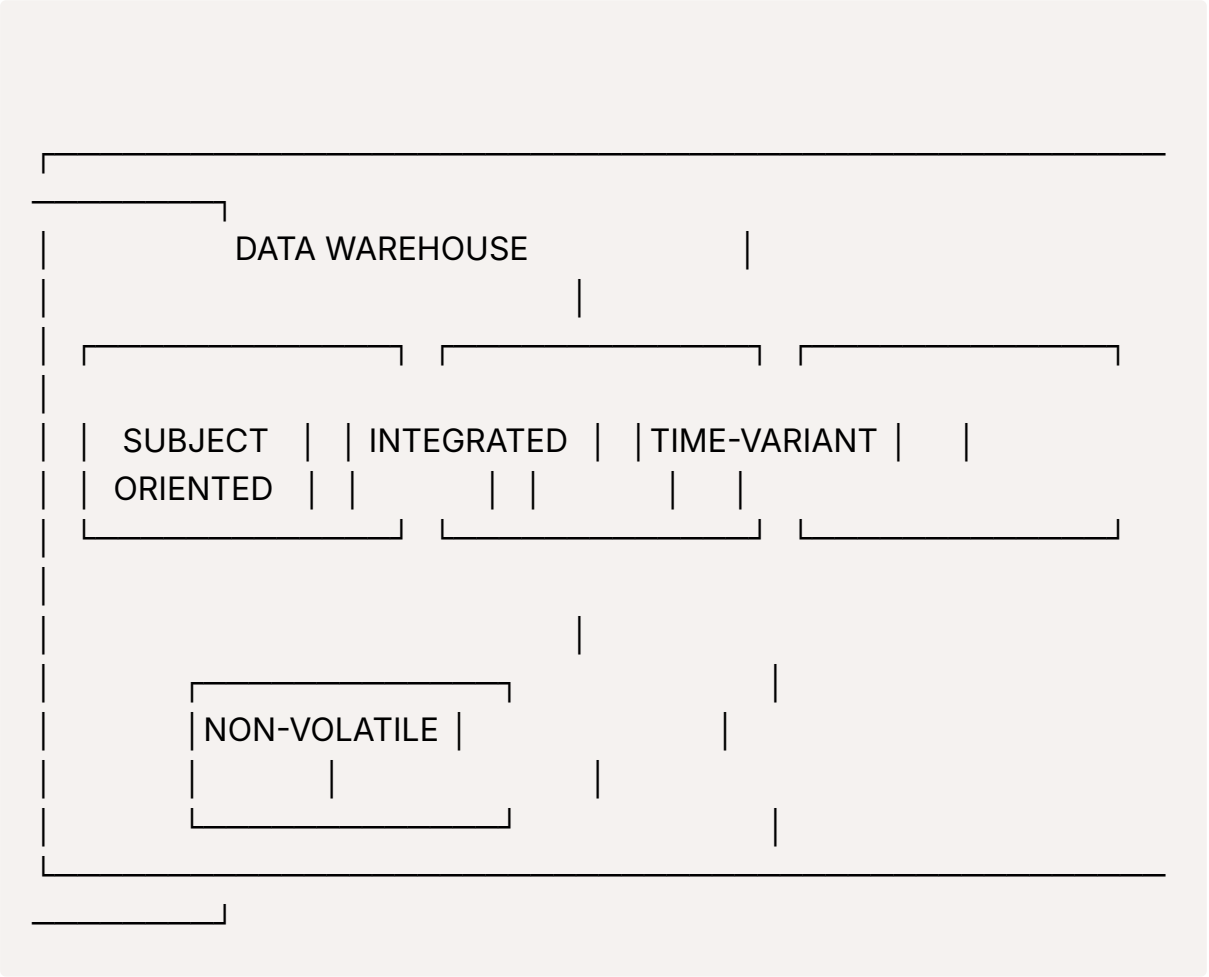
- **Subject-oriented**
- **Integrated**
- **Time-variant**

- **Non-volatile**

Primary Purpose

The primary purpose of a Data Warehousing (DW) process is to collect and manage data from various sources to provide meaningful business insights. It serves as the core of a Business Intelligence (BI) system built for data analysis and reporting.

Key Characteristics



1. Subject Oriented

- Focuses on specific subjects/topics relevant to business (sales, marketing, finance)
- Data is organized based on what people need to analyze

2. Time Variant

- Data is maintained over different intervals (weekly, monthly, annually)

- Tracks historical information showing changes over time
- Once stored, data generally cannot be modified to preserve historical accuracy

3. Integrated

- Combines information from diverse sources (databases, ERP, flat files, CRM)
- Requires consistency in naming conventions, column scaling, and encoding structures
- Handles various subject-related warehouses

4. Non-Volatile

- Data is permanent and read-only
- New data is inserted, but existing data is not erased or updated
- Refreshed at particular intervals
- No transaction processing, recovery, or concurrency control mechanisms needed

Real-World Example

Walmart's Enterprise Data Warehouse (EDW):

- Analyzes multi-year sales and inventory trends across all stores
- Integrates data from Point-of-Sale systems, supply chain databases, marketing platforms
- Stores historical sales data over many years (Time-Variant)
- Focused on subjects like sales performance (Subject-Oriented)
- Combines data from different operational systems (Integrated)
- Retains old sales records even as new ones are added (Non-Volatile)

Layman's Explanation

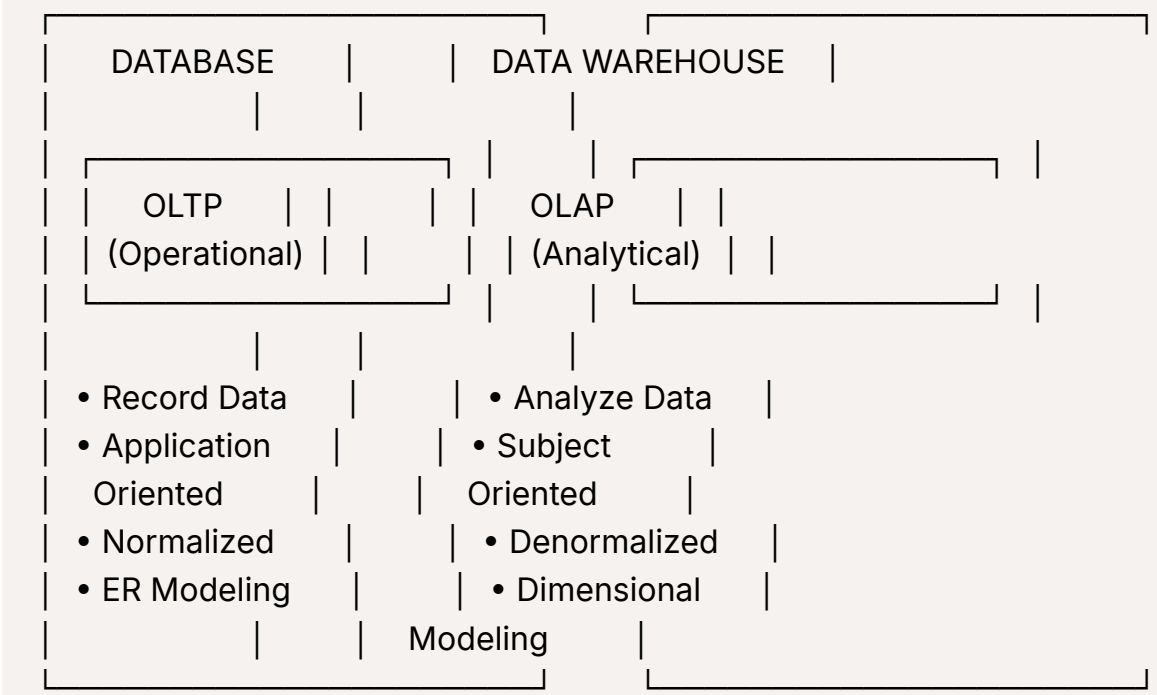
Imagine you have lots of different notebooks for different aspects of your life (spending, hobbies, grades). A Data Warehouse is like taking all that information and putting it into one giant, organized book, designed so you can

easily look back at different parts of your life to understand patterns and make better decisions. It's like a history book for your business.

---

2. DATABASE vs DATA WAREHOUSE

Fundamental Differences



Key Distinctions

AspectDatabaseData WarehousePurposeRecord dataAnalyze dataOrientation Application-orientedSubject-orientedProcessingOLTP (Online Transactional Processing)OLAP (Online Analytical Processing)DesignNormalized tablesDenormalized tablesModelingEntity-Relationship (ER)Dimensional modelingOptimizationFast transactionsComplex queries on large datasetsData StructureHighly normalized, complex joinsEasier to understand and query

Example Scenario

Online Store:

- **Database (OLTP)**: Records every customer order, processes payments, manages inventory in real-time
- **Data Warehouse (OLAP)**: Stores historical records of all orders over several years for business analysts to understand sales trends, identify profitable products, and analyze customer purchasing patterns

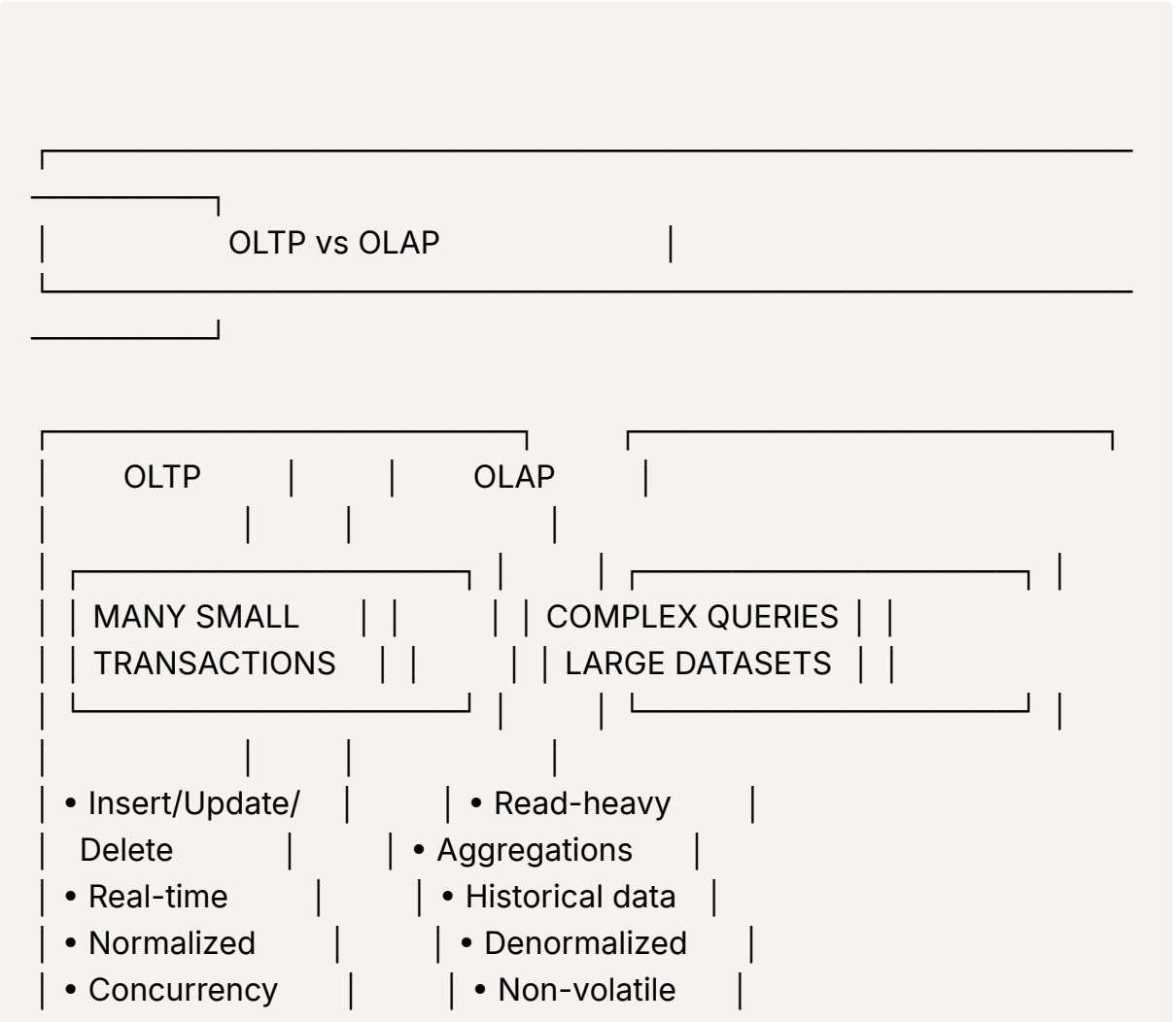
Layman's Explanation

- **Database**: Like a cashier's till at a busy shop - designed for speed and accuracy of individual transactions happening right now
- **Data Warehouse**: Like a detailed archive and analysis center - designed for looking back and understanding the big picture over time

---

3. OLTP vs OLAP

System Comparison



OLTP (Online Transactional Processing)

Characteristics:

- Designed for operational tasks
- Handles large number of concurrent, small transactions
- Involves inserting, updating, or deleting data
- Prioritizes speed and efficiency for day-to-day operations
- Ensures data integrity and concurrency control
- Highly normalized databases

Examples:

- Online banking
- E-commerce stores
- Order entry systems

OLAP (Online Analytical Processing)

Characteristics:

- Designed for analytical tasks
- Queries large volumes of historical data
- Enables complex analysis, reporting, and data mining
- Prioritizes read performance on aggregated data
- Used for business intelligence and strategic decision-making
- Often denormalized data structures

Examples:

- Business intelligence systems
- Data warehouses
- Reporting systems

Real-World Examples

OLTP Example: ATM Transaction

When you withdraw money from an ATM:

- Quickly checks your balance
- Debits your account
- Updates balance in real-time
- Small, quick transaction among many concurrent ones

OLAP Example: Bank Analysis

A bank analyst queries a Data Warehouse to:

- Identify trends in ATM withdrawals across regions
- Analyze peak withdrawal times
- Predict future cash needs at various locations
- Involves scanning large datasets and performing aggregations

Layman's Explanation

- **OLTP**: Busy supermarket cashier scanning items and processing payments quickly and accurately right now
- **OLAP**: Business analyst looking at months/years of sales data to understand which products sold best during holidays and plan future inventory

---

4. ETL vs ELT

Process Flow Diagrams

```
ETL PROCESS:
Source → Extract → Transform → Load → Target
          ↓         ↓          ↓      ↓
       Raw Data  Clean/     Staged  Data Warehouse
                 Structure  Data
```

```
ELT PROCESS:
Source → Extract → Load → Transform → Analysis
       ↓        ↓     ↓        ↓
    Raw Data  Direct  Transform  Results
              Load    in Target
```

ETL (Extract, Transform, Load)

Process Steps:

1. **Extract**: Data pulled from source systems
2. **Transform**: Data cleaned, structured, aggregated before loading (often on staging server)
3. **Load**: Transformed data loaded into target system (typically Data Warehouse)

Characteristics:

- Traditional approach for on-premises, relational, structured data
- Easier to implement for smaller data amounts
- Does not typically provide data lake support
- Transform happens outside the target system

ELT (Extract, Load, Transform)

Process Steps:

1. **Extract**: Data pulled from source systems
2. **Load**: Data loaded directly into target system (often powerful data warehouse/data lake)
3. **Transform**: Data transformed within target system using its processing power

Characteristics:

- Used for scalable cloud-based data sources
- Handles structured and unstructured data
- Preferred for large amounts of data

- Provides data lake support
- Requires more niche skills to implement
- Leverages target system's processing power

Comparison Table

AspectETLELTTransform LocationExternal staging serverWithin target syst emData TypesPrimarily structuredStructured & unstructuredData VolumeS maller amountsLarge amountsImplementationEasierRequires niche skillsDat a Lake SupportNoYesProcessing PowerExternalTarget systemUse CaseOn-premises, traditionalCloud-based, scalable

Examples

ETL Example: Survey Responses

- **Extract**: Collect survey responses from website forms and paper forms
- **Transform**: Standardize city name spellings, convert date formats, calculate summary scores
- **Load**: Insert cleaned, standardized data into central database

ELT Example: Website Clickstream

- **Extract**: Collect raw clickstream data from millions of visitors
- **Load**: Load directly into powerful cloud data warehouse/data lake
- **Transform**: Run queries within warehouse to filter by users, aggregate clicks, join with other datasets

Layman's Explanation

ETL: Organizing Clothes

You take clothes from the laundry basket (Extract), fold and iron them neatly (Transform), then put organized piles into your closet (Load). The sorting happens before storage.

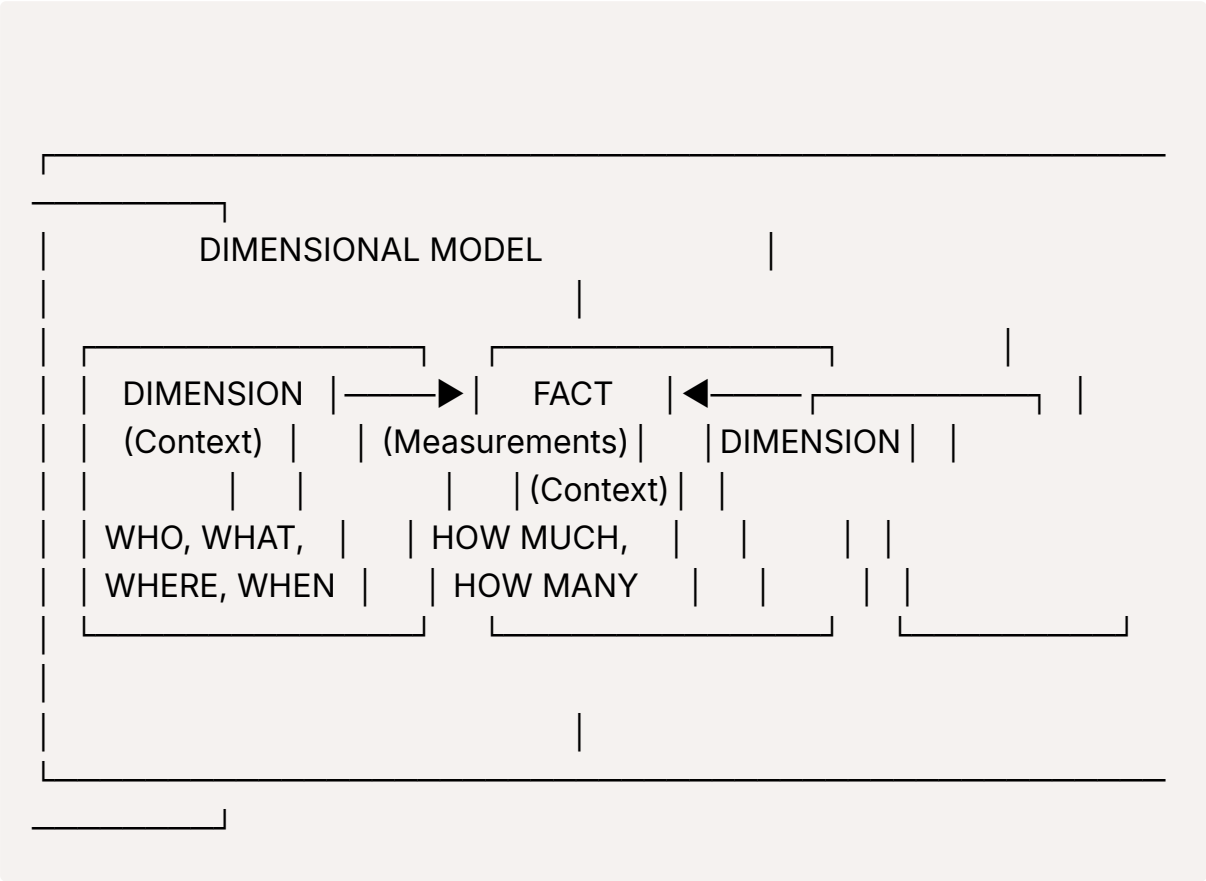
ELT: Storage Unit Approach

You dump all clothes straight into a big storage unit (Extract, Load), then when you need a specific outfit, you go in and sort through only what you need (Transform within storage).

---

5. FACT vs DIMENSION

Conceptual Diagram



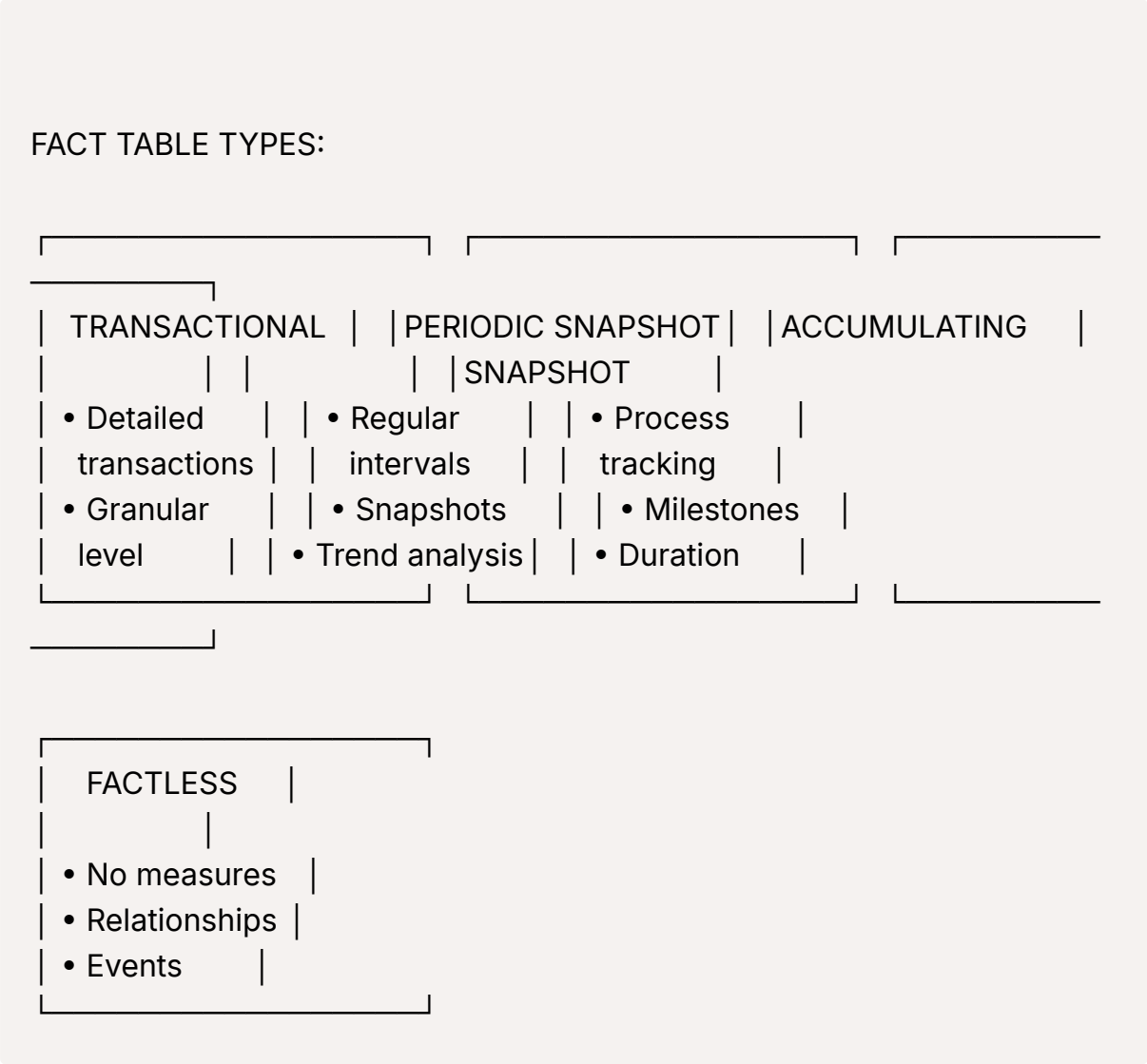
FACT TABLES

Definition: Store measurements or metrics from business processes

Characteristics:

- Contain numerical values that can be aggregated
- Answer questions like "How much?", "How many?"
- Contain foreign keys linking to dimension tables
- Provide data for understanding patterns, trends, and performance metrics

Types of Fact Tables



1. **Transactional Fact Table**
    - Captures detailed data about individual business transactions
    - Records every occurrence at granular level
    - Useful for operational reporting and auditing
2. **Periodic Snapshot Fact Table**
    - Captures data at regular intervals (daily, weekly, monthly)
    - Provides snapshot view at specific points in time
    - Used for trend analysis and comparison
3. **Accumulating Snapshot Fact Table**
    - Tracks progress of a process over time
    - Captures key milestones and duration

- Useful for process monitoring and performance analysis

4. **Factless Fact Table**
    - Contains no measures or numeric data
    - Captures relationships between entities
    - Represents events without numerical values

DIMENSION TABLES

Definition: Store descriptive information about business entities

Characteristics:

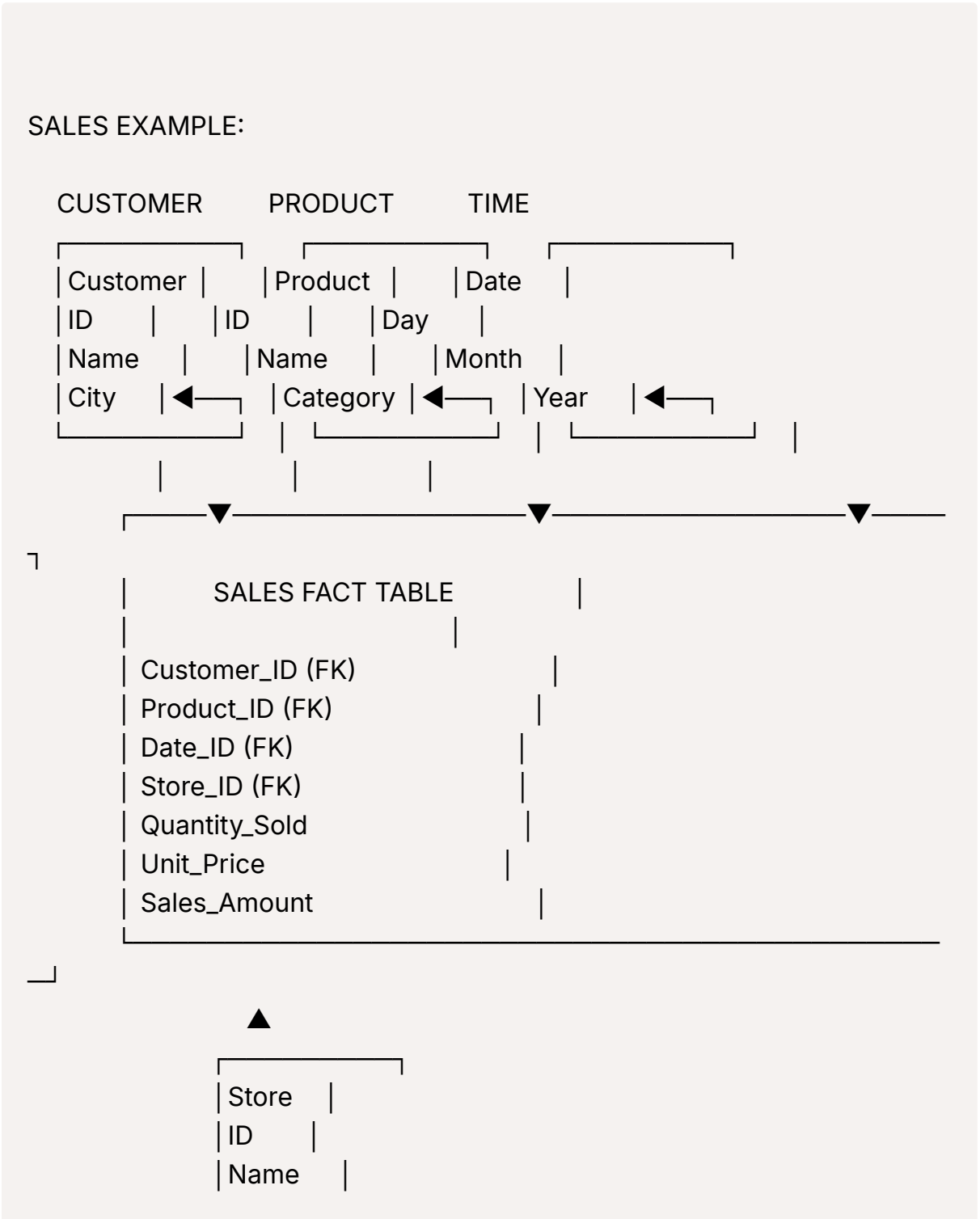
- Provide context surrounding business process events
- Answer questions like "Who?", "What?", "Where?", "When?"
- Hold descriptive attributes (names, descriptions, categories)
- Usually have low cardinality (fewer unique values)
- Often have hierarchical structure

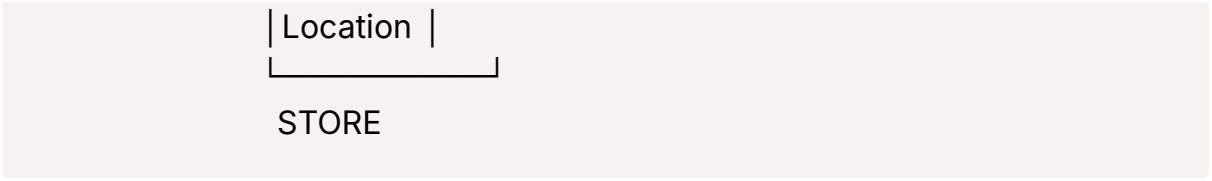
Types of Dimension Tables

1. **Role-Playing Dimension Table**
    - Single dimension used in multiple ways
    - Same entity from different perspectives
    - Example: Date dimension for Order Date, Ship Date, Delivery Date
2. **Conformed Dimension Table**
    - Shared across multiple data marts
    - Ensures consistency in reporting
    - Same dimension used organization-wide
3. **Junk Dimension Table**
    - Groups low-cardinality, non-meaningful attributes
    - Attributes that don't fit well in other dimensions
    - Example: Various promotional flags
4. **Degenerate Dimension Table**

- Attributes that are part of fact table itself
- Serve as unique identifiers
- Don't require separate dimension table
- Example: Order Number in sales fact table

Sales Business Process Example





Facts: Quarterly sales number, price, quantity sold

Dimensions: Customer Name, Location, Product Name, Date

Lemonade Stand Example

Facts (What happened):

- Made $50 today
- Sold 25 cups
- Cost of lemons: $10

Dimensions (Context):

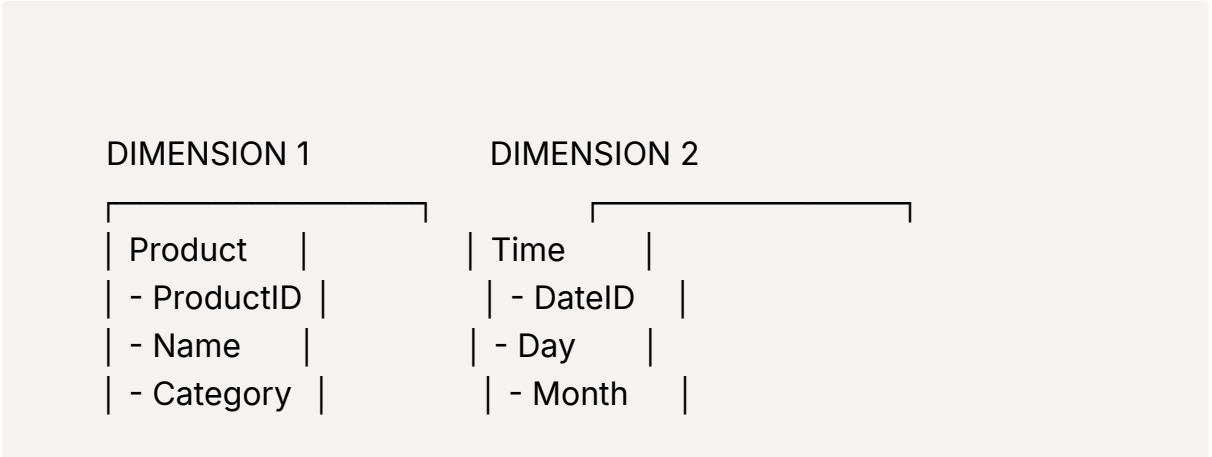
- **When**: Today's date (Time dimension)
- **Where**: Location of stand (Location dimension)
- **What**: Type of lemonade (Product dimension)
- **Who**: Customer type (Customer dimension)

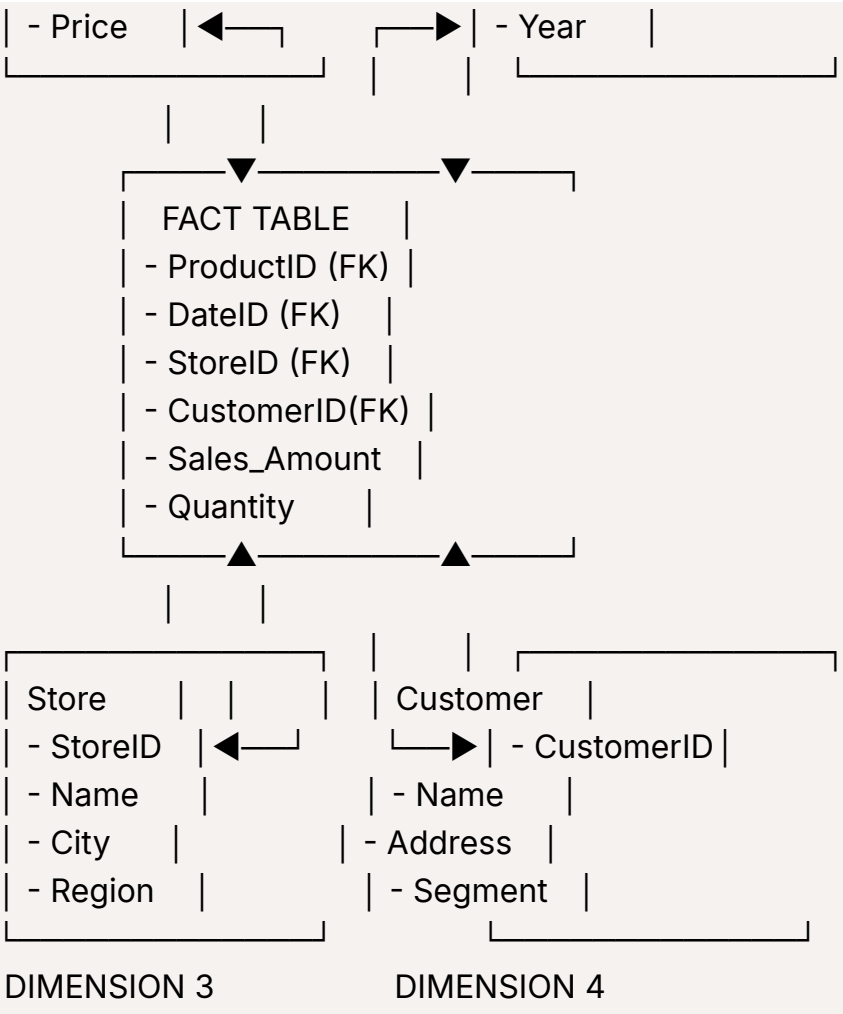
---

6. STAR, SNOWFLAKE & FACT CONSTELLATION SCHEMA

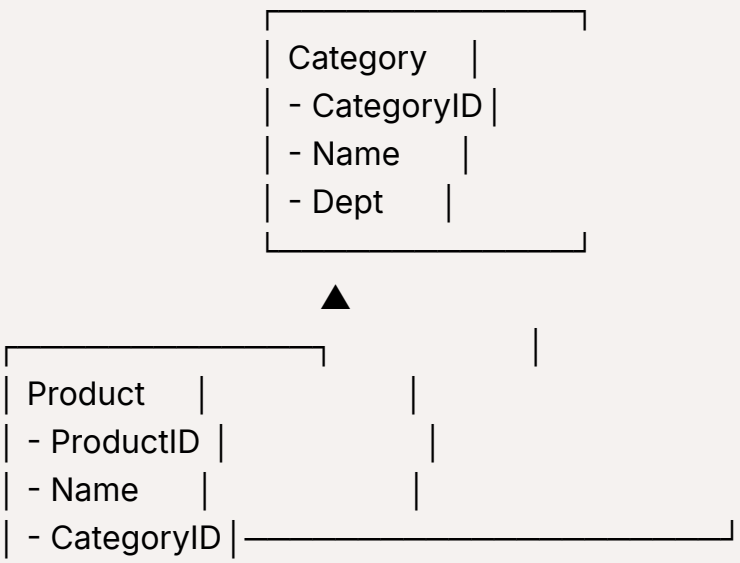
Schema Diagrams

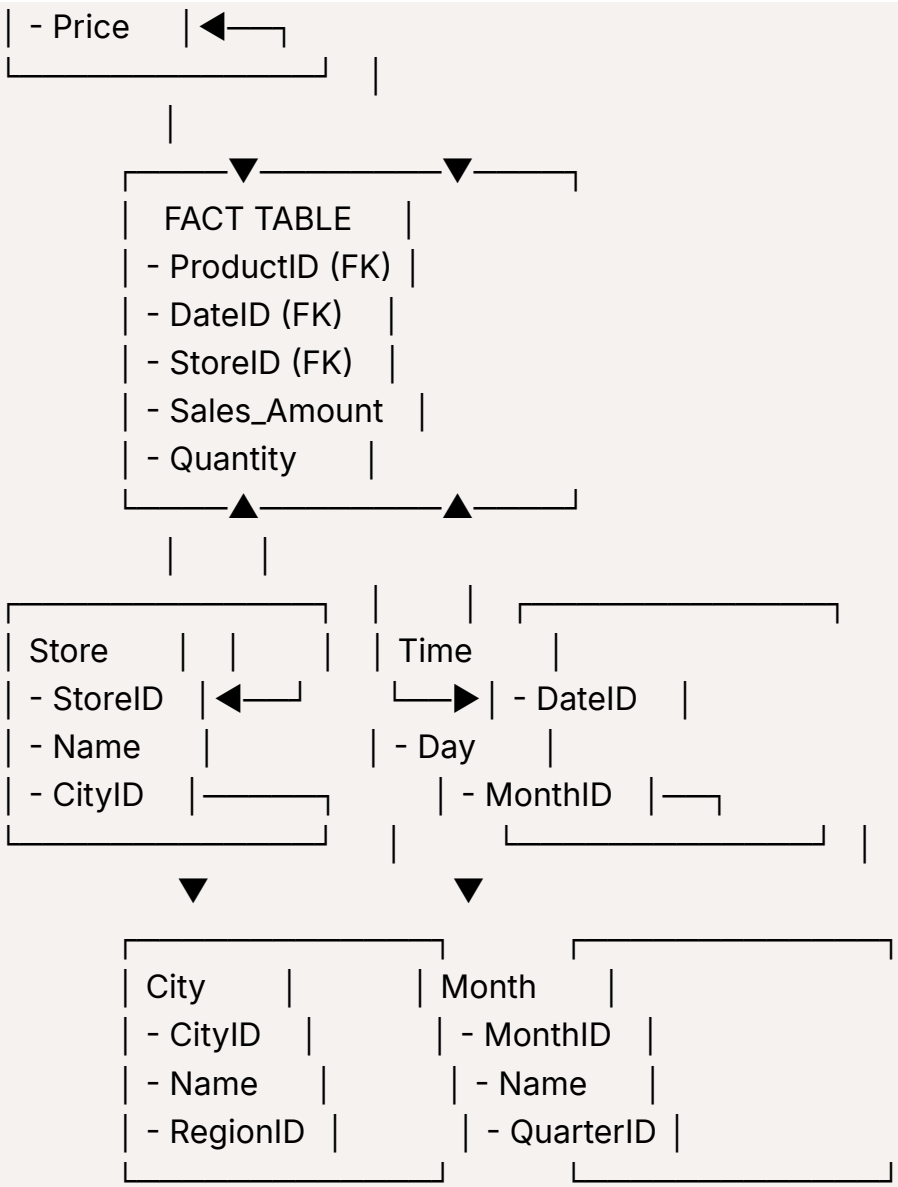
Star Schema





Snowflake Schema





Schema Comparison

AspectStar SchemaSnowflake SchemaFact ConstellationStructureSimple, denormalizedComplex, normalizedMultiple fact tablesJoinsFewer, simplerMore, complexShared dimensionsQuery PerformanceFasterSlowerVariableStorageMore redundancyLess redundancyModerateMaintenanceEasierMore difficultComplexUnderstandingSimpleDifficultModerate

Detailed Characteristics

1. Star Schema

- **Design**: Central fact table directly connected to dimension tables
- **Dimensions**: Denormalized (all attributes in single table)
- **Methodology**: Top-down (Bill Inmon approach)
- **Query Execution**: Fast, low complexity
- **Normalization**: Less normalized, high data redundancy
- **Foreign Keys**: Fewer

2. Snowflake Schema

- **Design**: Fact table connected to normalized dimension tables
- **Dimensions**: Normalized into multiple sub-dimension tables
- **Methodology**: Bottom-up (Ralph Kimball approach)
- **Query Execution**: Slower, higher complexity
- **Normalization**: Highly normalized, low data redundancy
- **Foreign Keys**: More

3. Fact Constellation Schema (Galaxy Schema)

- **Design**: Multiple fact tables sharing common dimensions
- **Structure**: Collection of star/snowflake schemas
- **Use Case**: Different business processes sharing dimensional information

Real-World Examples

Star Schema Example

Sales Fact table directly connected to:

- Time Dimension (Date, Month, Year)
- Product Dimension (ProductID, Name, Category)
- Store Dimension (StoreID, Name, City)

Snowflake Schema Example

Sales Fact table connected to Product Dimension, which connects to separate Category Dimension table. Store Dimension connects to separate City Dimension table.

Fact Constellation Example

Sales Fact table and Inventory Fact table sharing:

- Common: Time Dimension, Product Dimension
- Unique to Sales: Customer Dimension
- Unique to Inventory: Warehouse Dimension

Layman's Explanation

Star Schema

Organizing shop records with main sales list directly pointing to separate lists for Date, Items, and Store. All details are right there in each list - simple and direct.

Snowflake Schema

Same sales list, but Item list points to Category list, which might point to Department list. Store list points to City list. Details are spread across connected lists - more organized but requires following more links.

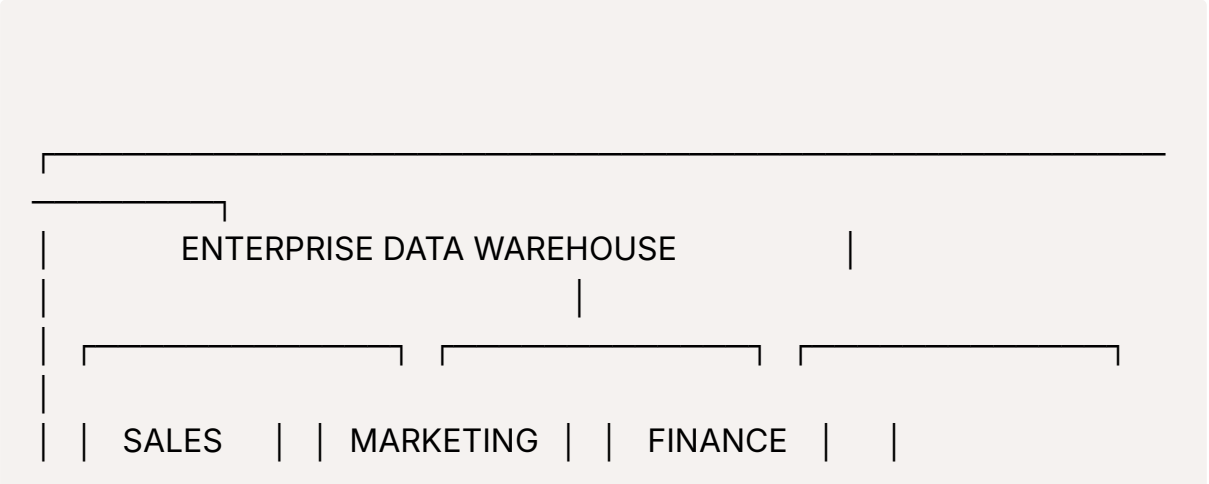
Fact Constellation Schema

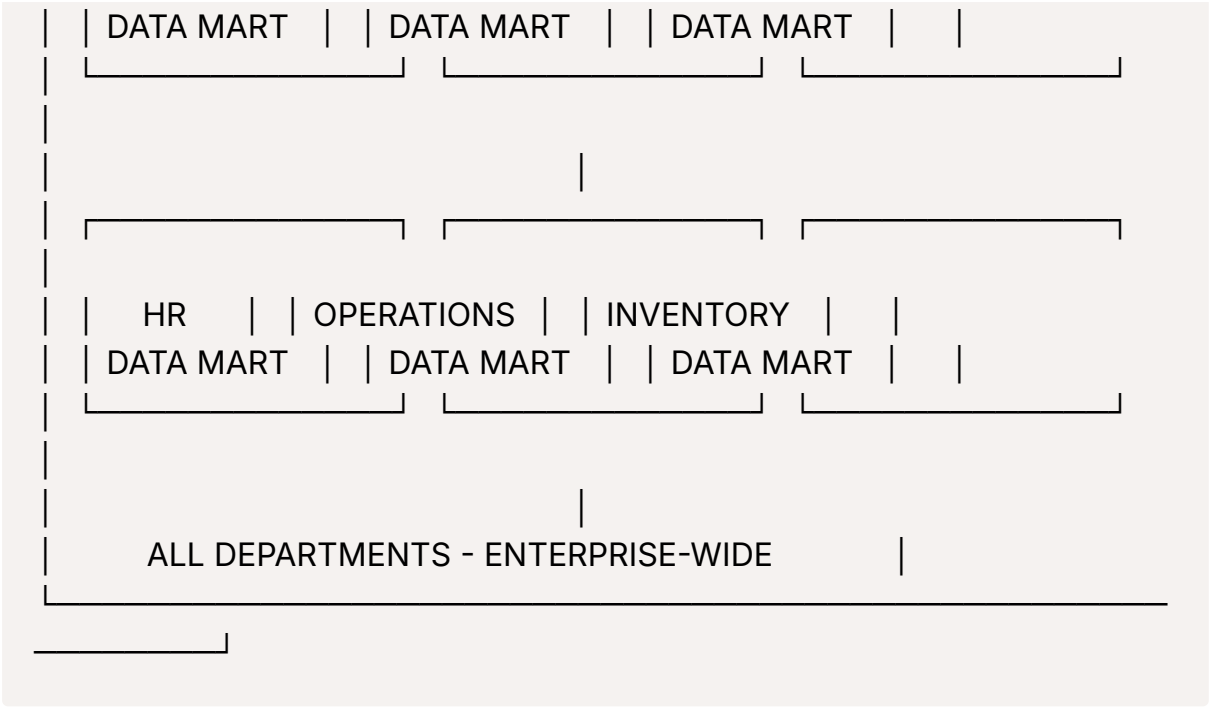
Multiple main lists (Sales and Inventory) sharing some of the same other lists (Items, Dates) but each having unique lists too. Like multiple star patterns connecting at some points.

---

7. DATA MART

Data Mart vs Enterprise Data Warehouse





Definition and Characteristics

Data Mart: A simplified form of data warehouse focusing on specific department, subject area, or business function within an organization.

Key Characteristics:

- Subtype of data warehouse
- Department-specific focus (marketing, sales, finance)
- Smaller scope compared to Enterprise Data Warehouse
- Easier and faster to design and implement
- Faster data handling due to smaller size
- Typically less than 100 GB (EDW: 100 GB to 1 TB+)

Types of Data Storage Systems

1. Enterprise Data Warehouse (EDW)

- **Purpose**: Central, historical data for enterprise-wide analytics
- **Scope**: All departments and business units
- **Example**: Walmart analyzing multi-year sales & inventory trends across all stores

2. Operational Data Store (ODS)

- **Purpose**: Real-time data store for operational systems
- **Scope**: Near-real-time operational reporting
- **Example**: Bank of America providing live transaction info to support agents

3. Data Mart

- **Purpose**: Department-focused data for fast, relevant insights
- **Scope**: Specific business function or department
- **Example**: Amazon's marketing team optimizing campaign performance

Comparison Table

AspectEnterprise Data WarehouseData MartScopeEnterprise-wideDepartment-specificSize100 GB to 1 TB+Less than 100 GBDesign ComplexityHighLowerImplementation TimeLongerFasterData HandlingSlower (large volume)Faster (smaller volume)UsersAll departmentsSpecific departmentCostHigherLowerMaintenanceComplexSimpler

Real-World Example

Large Retail Company:

Enterprise Data Warehouse:

- Contains sales, inventory, customer, supplier, employee data
- Covers all regions and business units
- Comprehensive but can be slow to navigate

Marketing Data Mart:

- Extracts only customer and sales data relevant to marketing
- Aggregated weekly/monthly for campaign analysis
- Allows marketing team to quickly analyze campaign effectiveness
- No need to access entire EDW

Layman's Explanation

Enterprise Data Warehouse: Giant, multi-story library containing every book the company owns on every subject. Comprehensive but slow to navigate for specific topics.

Data Mart: Small section of that library (e.g., just "Marketing" books) in a dedicated reading room. Focused on specific area and quicker to access for those who need that specific information.

When to Use Data Marts

Choose Data Mart when:

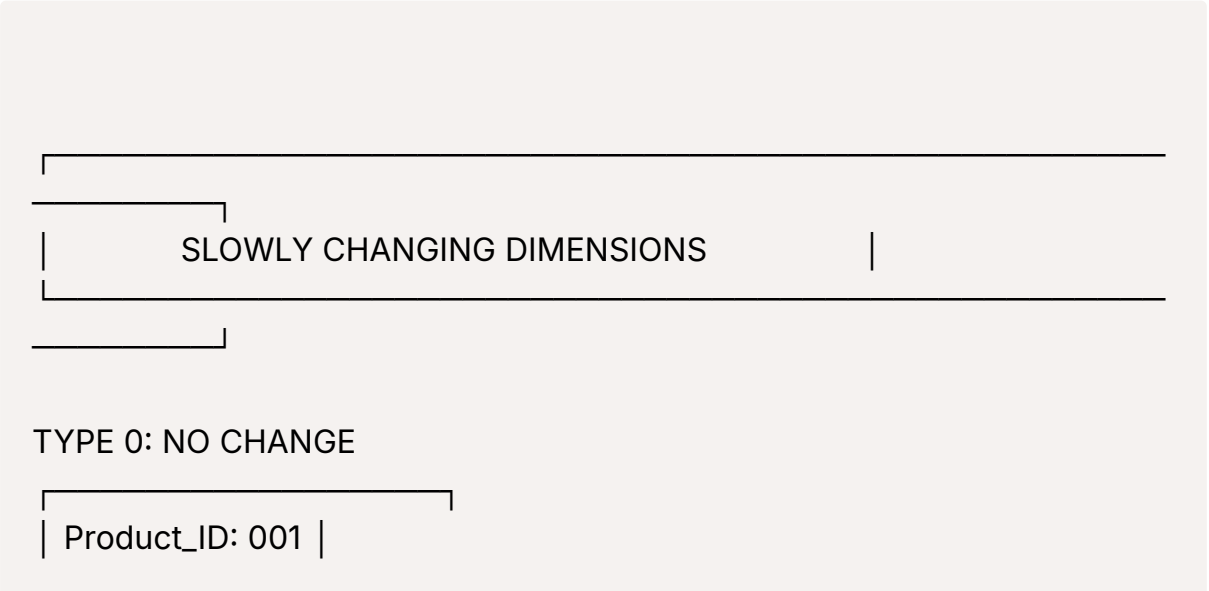
- Specific department has unique analytical needs
- Faster implementation required
- Limited budget and resources
- Department needs quick access to relevant data
- Pilot project before full EDW implementation

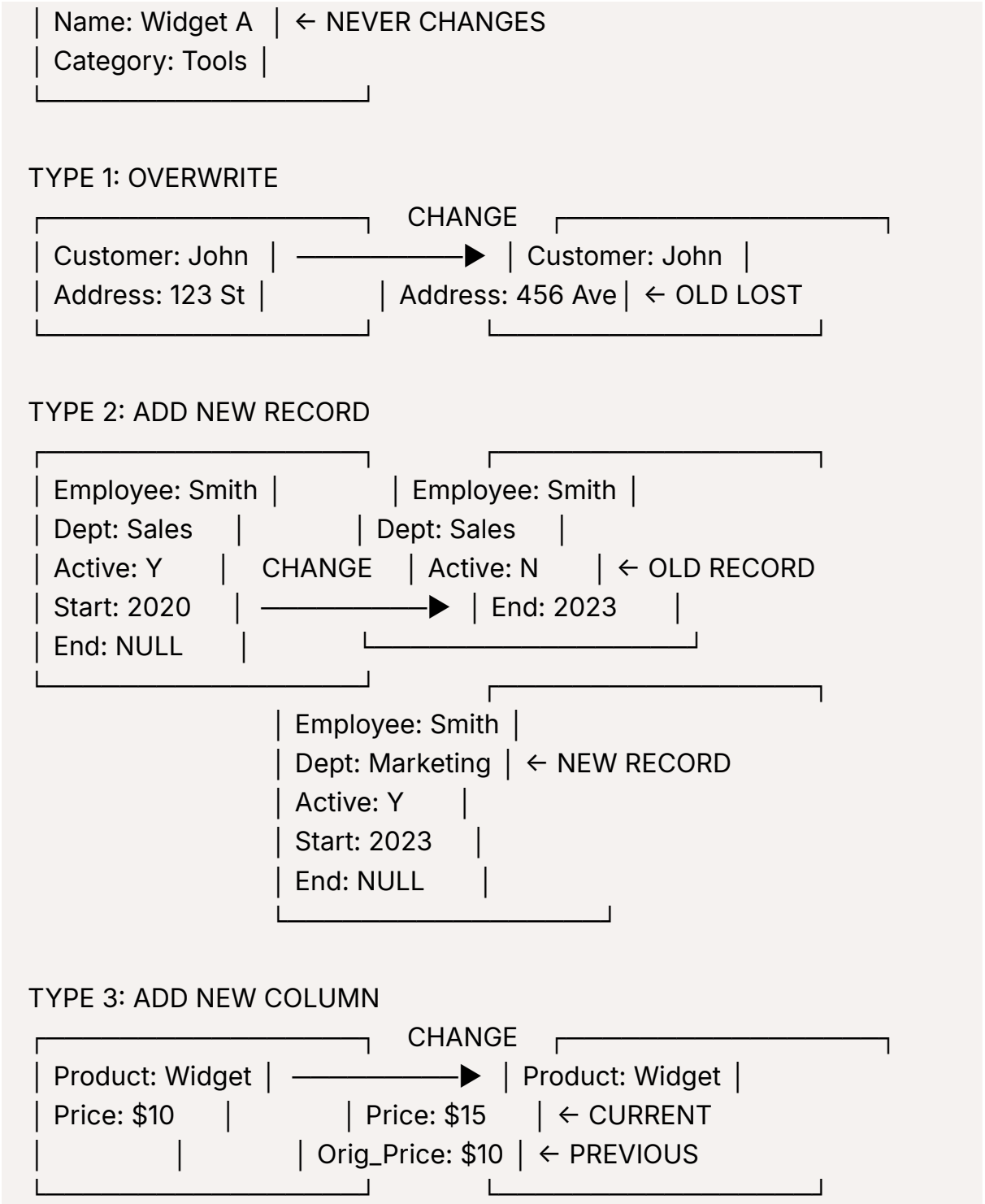
Potential Issues:

- **Data Mart Sprawl**: Disconnected data marts leading to inconsistencies
- **Integration Challenges**: Difficulty in cross-departmental analysis
- **Duplication**: Same data stored in multiple marts

8. SLOWLY CHANGING DIMENSIONS (SCD)

SCD Types Overview





Detailed SCD Types

SCD Type 0: No Change

Explanation: Dimension attributes never change. Once loaded, data remains static.

Characteristics:

- Updates from source system are ignored
- Historical accuracy for specific attribute is essential
- Attribute treated as constant over time

Use Case: Product dimension where product name, category, and description are static in DW

Example:

```
Product Table:
Product_ID | Name     | Category | Description
001        | Widget A | Tools    | Basic Tool
(This data never changes in DW even if source system updates it)
```

SCD Type 1: Overwrite (No Historical Information)

Explanation: Changes overwrite existing values. Old data is lost.

Characteristics:

- No preservation of historical changes
- Only most recent information is kept
- Simplest implementation

Use Case: When historical data for an attribute is not important

Example:

```
Customer Table - Before Change:
Customer_ID | Name | Address
001         | John | 123 Main St

Customer Table - After Change:
Customer_ID | Name | Address
001         | John | 456 Oak Ave  (Old address lost)
```

SCD Type 2: Add New Record (with Active Flag & Date)

Explanation: Track changes by creating new records. Preserves full history.

Characteristics:

- New record created for changed attribute
- Old record marked inactive/expired
- Often includes StartDate, EndDate, Active flag
- Complete history maintained

Use Case: When tracking historical changes is essential for accurate reporting

Example:

```
Employee Table - Before Change:
Emp_ID | Name  | Department | Active | Start_Date | End_Date
001    | Smith | Sales      | Y      | 2020-01-01 | NULL

Employee Table - After Department Change:
Emp_ID | Name  | Department | Active | Start_Date | End_Date
001    | Smith | Sales      | N      | 2020-01-01 | 2023-06-30
001    | Smith | Marketing  | Y      | 2023-07-01 | NULL
```

SCD Type 3: Add New Column

Explanation: Limited history by adding columns to track specific changes.

Characteristics:

- Tracks only previous value alongside current value
- Updates existing record while retaining some historical info
- Compact model with limited history

Use Case: When you only need to track one or limited previous values

Example:

```
Product Table - Before Price Change:
Product_ID | Name   | Price
001        | Widget | $10

Product Table - After Price Change:
Product_ID | Name   | Price | Original_Price
001        | Widget | $15   | $10
```

Customer Address Change Examples

Type 0: Email address stored and considered static - changes ignored

Type 1: Address change replaces old address - only current address available

Type 2: Address change creates new row with new address and dates - full history preserved

Type 3: Address change keeps current address but adds "Previous_Address" column

Layman's Explanation - Contact Book Analogy

Type 0: Once you write a friend's email, you never update it even if they change it

Type 1: When friend changes phone number, cross out old one and write new one - only current number available

Type 2: When friend changes number, write new number on new line with date changed, mark old number "No Longer Used After [Date]" - both records kept

Type 3: When friend changes number, keep current number but add "Previous Number" space next to it - remember only current and one previous

Decision Matrix

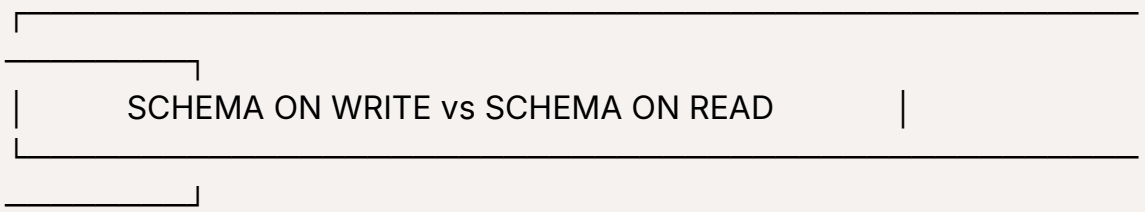
NeedSCD TypeReasonNo historical tracking neededType 1Simple, saves sp aceFull historical accuracy requiredType 2Complete audit trailLimited previ

ous values neededType 3Compact with some historyAttribute should never changeType 0Data integrity

---

9. SCHEMA ON WRITE vs SCHEMA ON READ

Conceptual Comparison



SCHEMA ON WRITE:
Raw Data → Schema Definition → Validation → Structured Storage
↓ ↓ ↓
Define Check Data Warehouse
Structure Format (Tables/Columns)

SCHEMA ON READ:
Raw Data → Direct Storage → Schema Applied → Query Results
↓ ↓ ↓
Data Lake Interpret Analysis
(Files) on Query

Schema on Write

Definition: Schema is defined and enforced BEFORE data is written into storage system.

Process Flow:

1. Define schema (structure/columns/data types)
2. Validate data against predefined structure during ingestion
3. Reject or transform non-conforming data

4. Store structured data

Characteristics:

- **Used In**: Traditional Data Warehouses (Snowflake, Redshift)
- **Performance**: Faster reads, slower writes
- **Flexibility**: Less flexible - structure must be known upfront
- **Storage**: Structured tables with enforced columns and data types
- **Validation**: At write time

Schema on Read

Definition: Schema is applied when data is read or queried, NOT when written.

Process Flow:

1. Store raw data in original format
2. Apply schema interpretation at query time
3. Structure determined during analysis
4. Flexible interpretation per use case

Characteristics:

- **Used In**: Data Lakes (S3, HDFS) with query engines (Athena, Hive)
- **Performance**: Faster writes, slower reads
- **Flexibility**: Highly flexible - structure defined at query time
- **Storage**: Raw files (CSV, JSON, Parquet)
- **Validation**: At read time

Detailed Comparison

AspectSchema on WriteSchema on ReadWhen Schema AppliedBefore writi ng dataDuring data readingStorage SystemData WarehousesData LakesDat a TypesStructuredStructured, Semi-structured, UnstructuredWrite Perform anceSlower (validation)Faster (direct load)Read PerformanceFaster (pre-str uctured)Slower (schema interpretation)FlexibilityLowerHigherUpfront Plann ingRequiredOptionalStorage FormatTables/ColumnsFiles (various formats)

```
Use CaseKnown structure, BI/reportingExploratory analysis, unknown futur
e use
```

Real-World Examples

Schema on Write Example: SQL Database

```
-- Must define table structure first
CREATE TABLE Customers (
    CustomerID INT,
    Name VARCHAR(100),
    OrderDate DATE,
    Amount DECIMAL(10,2)
);

-- Data must conform to structure
INSERT INTO Customers VALUES (1, 'John', '2024-01-01', 100.50);
-- If data doesn't match (e.g., text in Amount field), insert fails
```

Schema on Read Example: Data Lake Analysis

```
-- Raw log files stored directly in S3 (various formats)
-- Query applies schema at runtime:

SELECT user_id, COUNT(*) as clicks
FROM s3://raw-logs/clickstream/
WHERE date = '2024-01-01'
GROUP BY user_id;

-- Schema interpreted during query execution
```

Performance Implications

Schema on Write

- **Advantages**: Fast queries, data quality ensured, optimized for BI
- **Disadvantages**: Slow ingestion, rigid structure, upfront planning required

Schema on Read

- **Advantages**: Fast ingestion, flexible analysis, handles variety of data
- **Disadvantages**: Slower queries, potential data quality issues, complex query logic

Layman's Explanation

Schema on Write - Filing Cabinet

You build a filing cabinet with labeled drawers ("Invoices - 2023", "Receipts - 2024") and decide invoices must have date, company, amount in specific order. You organize and label everything before filing. Finding documents later is easy, but initial sorting takes time.

Schema on Read - Storage Box

You throw all papers (invoices, receipts, notes) into a big box as they arrive. When you need a specific invoice, you open the box and figure out which papers are invoices and find the information you need on the spot. Quick to store, but takes longer to find specific information.

When to Choose Each Approach

Choose Schema on Write When:

- Data structure is well-known and stable
- Performance for reads/queries is critical
- Strong data quality requirements
- Traditional BI and reporting needs
- Compliance and governance requirements

Choose Schema on Read When:

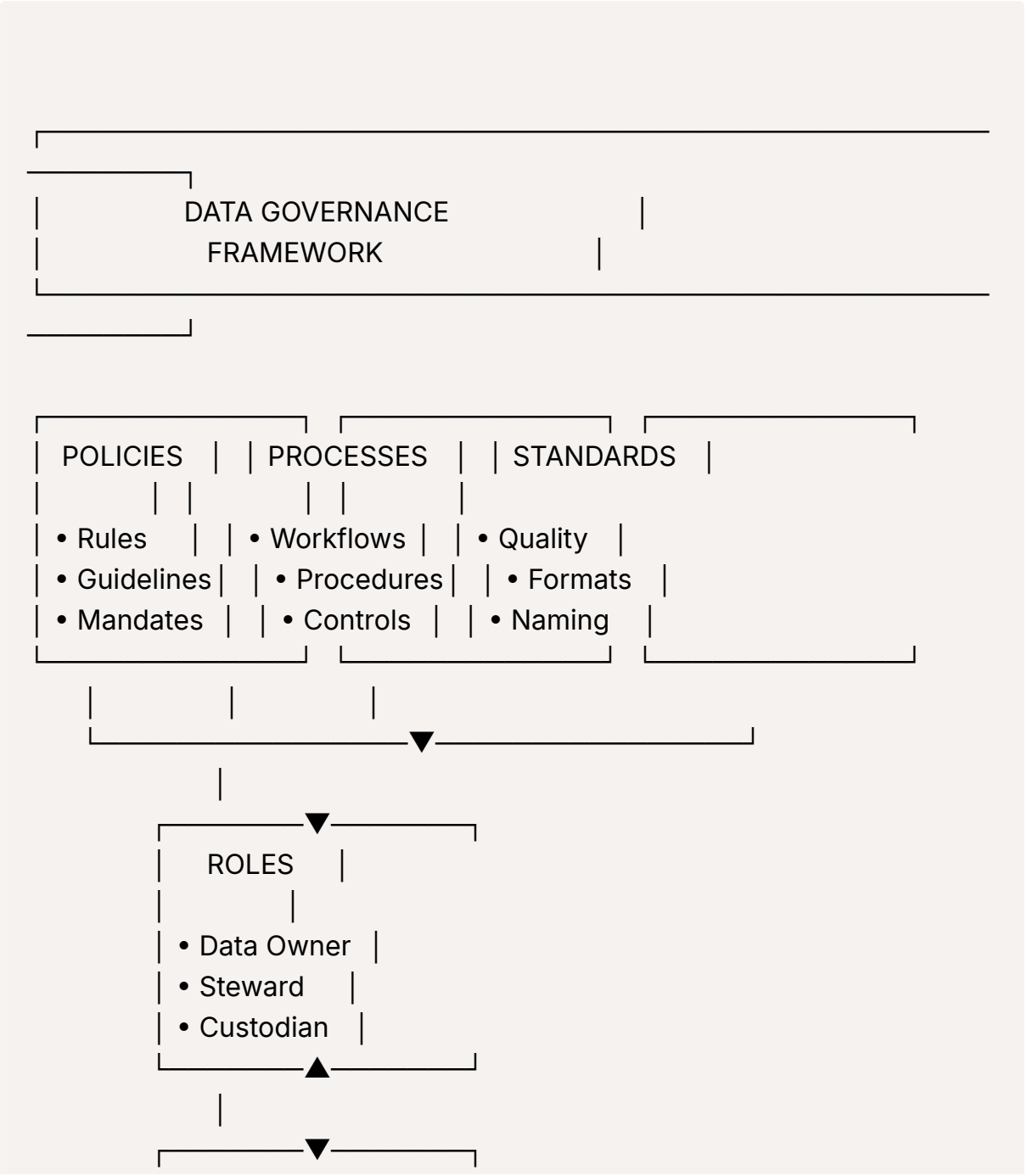
- Data structure is unknown or varies

- Need to handle diverse data sources
- Exploratory data analysis requirements
- Fast data ingestion is priority
- Working with unstructured/semi-structured data

---

10. DATA GOVERNANCE

Data Governance Framework



```
        │    TOOLS     │
        │              │
        │ • Catalogs   │
        │ • Monitoring │
        │ • Access     │
        └──────────────┘
```

Definition

Data governance is a comprehensive framework encompassing policies, processes, standards, roles, and tools. Its main goal is to ensure that an organization's data is managed effectively and responsibly.

Core Objectives

- Ensure data is **accurate**, **consistent**, **secure**, and **accessible**
- Build trust in data
- Comply with regulations
- Generate value from data assets
- Manage and control data throughout lifecycle

Key Components

1. Policies

- Data usage rules and guidelines
- Security and privacy requirements
- Compliance mandates
- Access control policies

2. Processes

- Data lifecycle management
- Quality assurance workflows
- Security procedures
- Audit and monitoring processes

3. Standards

- Data quality metrics
- Naming conventions
- Format specifications
- Classification standards

4. Roles and Responsibilities

- **Data Owners**: Business executives responsible for data domains
- **Data Stewards**: Day-to-day data management and quality
- **Data Custodians**: Technical implementation and maintenance
- **Data Users**: End users consuming data

5. Tools

- Data catalogs for discovery
- Quality monitoring systems
- Access control mechanisms
- Audit and lineage tracking

Key Aspects of Data Governance

```
DATA GOVERNANCE LIFECYCLE:

Creation → Storage → Usage → Archival → Deletion
   ↓        ↓       ↓        ↓         ↓
 Quality   Security Access  Retention Disposal
Standards Controls  Rules   Policies  Protocols
```

Essential Elements:

- **Data Standards and Policies**: Defining how data should be formatted, stored, and used

- **Data Ownership and Accountability**: Clear assignment of responsibility for data domains
- **Data Quality, Security, and Privacy**: Processes ensuring data integrity and protection
- **Data Lifecycle Management**: Managing data from creation to deletion
- **Regulatory Compliance**: Ensuring adherence to laws (GDPR, HIPAA, SOX)
- **Access Controls and Auditing**: Managing who can access what data and tracking usage

Real-World Example: Healthcare Provider

Data Governance Implementation:

1. **Standards**: Require validation of patient address formats, standardize medical coding systems
2. **Ownership**: Head of Patient Records owns demographic data, Chief Medical Officer owns clinical data
3. **Processes**:
    - Secure access procedures for patient data
    - Only authorized personnel can view sensitive information
    - Regular data quality audits
4. **Compliance**: Ensure all data handling meets HIPAA requirements
5. **Tools**:
    - Data catalog for finding approved datasets
    - Access control system for permission management
    - Audit trail for tracking data usage

Benefits of Data Governance

Business Benefits

- Improved decision-making through trusted data
- Reduced risks and compliance violations
- Enhanced operational efficiency

- Better customer experiences

Technical Benefits

- Consistent data across systems
- Reduced data duplication and conflicts
- Improved data quality and reliability
- Faster time to insights

Data Governance vs Data Management

AspectData GovernanceData ManagementFocusStrategy and oversightTactical executionScopePolicies and standardsTools and processesRoleWhat and whyHow and whenLevelStrategicOperational

Layman's Explanation - Library Analogy

Data Governance is like running a well-organized town library with strict rules:

Policies: Rules on how books are cataloged, borrowed, and returned

Processes: Procedures for checking books in/out, handling late returns, repairing damaged books

Standards: Organization system like Dewey Decimal

Roles: Librarians, section managers, each responsible for specific tasks

Tools: Catalog system, barcode scanners, security systems

This ensures the library's collection (data) is:

- **Accurate**: Books are in the right place
- **Consistent**: Properly cataloged
- **Secure**: Books aren't stolen, patron privacy protected
- **Accessible**: People can find books easily
- **Used Responsibly**: Borrowers follow rules

Implementation Challenges

Common Obstacles:

- Organizational silos and resistance to change
- Lack of executive sponsorship
- Unclear roles and responsibilities
- Insufficient funding and resources
- Technical complexity in large organizations
- Cultural issues around data sharing

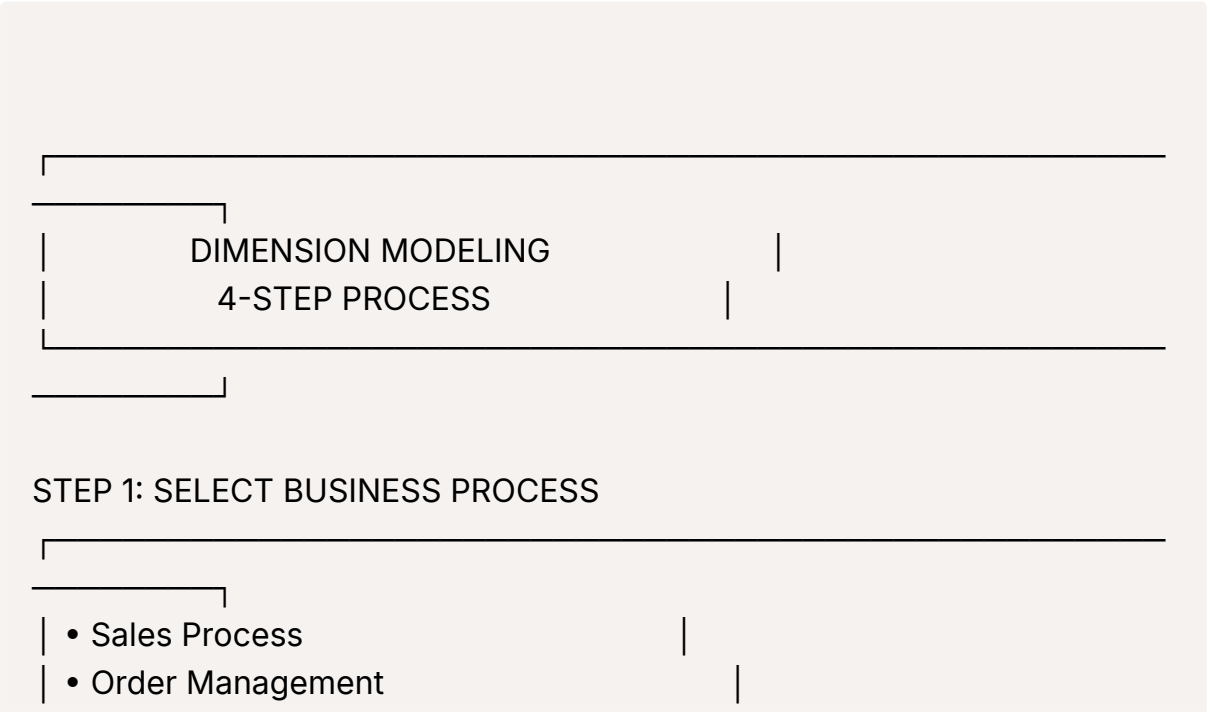
Success Factors:

- Strong leadership commitment
- Clear communication of benefits
- Start with pilot programs
- Establish data governance council
- Invest in training and change management
- Choose appropriate tools and technologies

---

11. DIMENSION MODELING (4-STEP PROCESS)

4-Step Process Overview



Step 1: Select the Business Process

Definition: Identify the specific operational process you want to model and analyze.

Examples of Business Processes:

- Sales transactions
- Order fulfillment
- Inventory management

- Customer service interactions
- Website visits
- Marketing campaigns
- Employee performance
- Financial transactions

Key Considerations:

- Choose process that drives business value
- Ensure sufficient data availability
- Consider stakeholder requirements
- Align with business priorities

Step 2: Declare the Grain

Definition: Determine the lowest level of detail at which data will be recorded in the fact table.

Why Critical:

- Defines exactly what a single row in fact table represents
- Impacts all subsequent design decisions
- Determines available dimensions and facts
- Affects query capabilities and performance

Grain Examples:

- **Each line item on sales order**: Most detailed level
- **Each individual transaction**: Single transaction level
- **Daily snapshot totals**: Aggregated by day
- **Monthly summary by product**: Aggregated by month and product

```
GRAIN EXAMPLES:

DETAILED GRAIN:
Each line item = One row per product in each order
```

```
Order 001: Product A (Row 1), Product B (Row 2), Product C (Row 3)

AGGREGATED GRAIN:
Daily totals = One row per day
Jan 1: Total sales across all products and customers
```

Step 3: Identify the Dimensions

Definition: Based on the business process and grain, identify entities that provide context to the facts.

Dimension Questions: WHO, WHAT, WHERE, WHEN, HOW

- **WHO**: Customer, Employee, Supplier
- **WHAT**: Product, Service, Promotion
- **WHERE**: Store, Region, Country
- **WHEN**: Date, Time, Fiscal Period
- **HOW**: Channel, Method, Campaign

Common Dimensions:

- **Date/Time**: When events occurred
- **Product**: What was sold/used
- **Customer**: Who was involved
- **Geography**: Where it happened
- **Employee**: Who performed action
- **Promotion**: Special offers applied

Step 4: Identify the Facts

Definition: Identify measurements or metrics that result from the business process at the declared grain level.

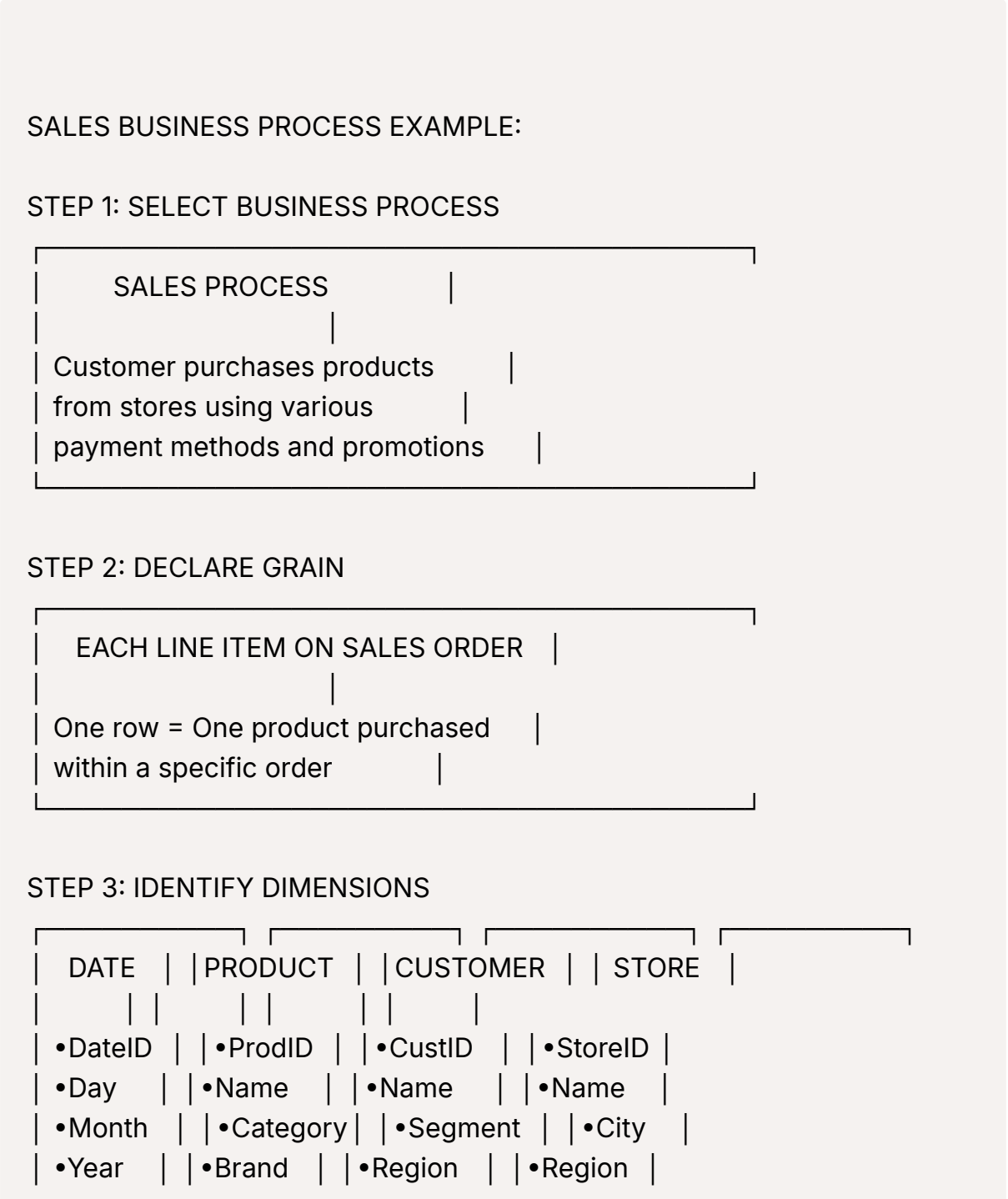
Fact Characteristics:

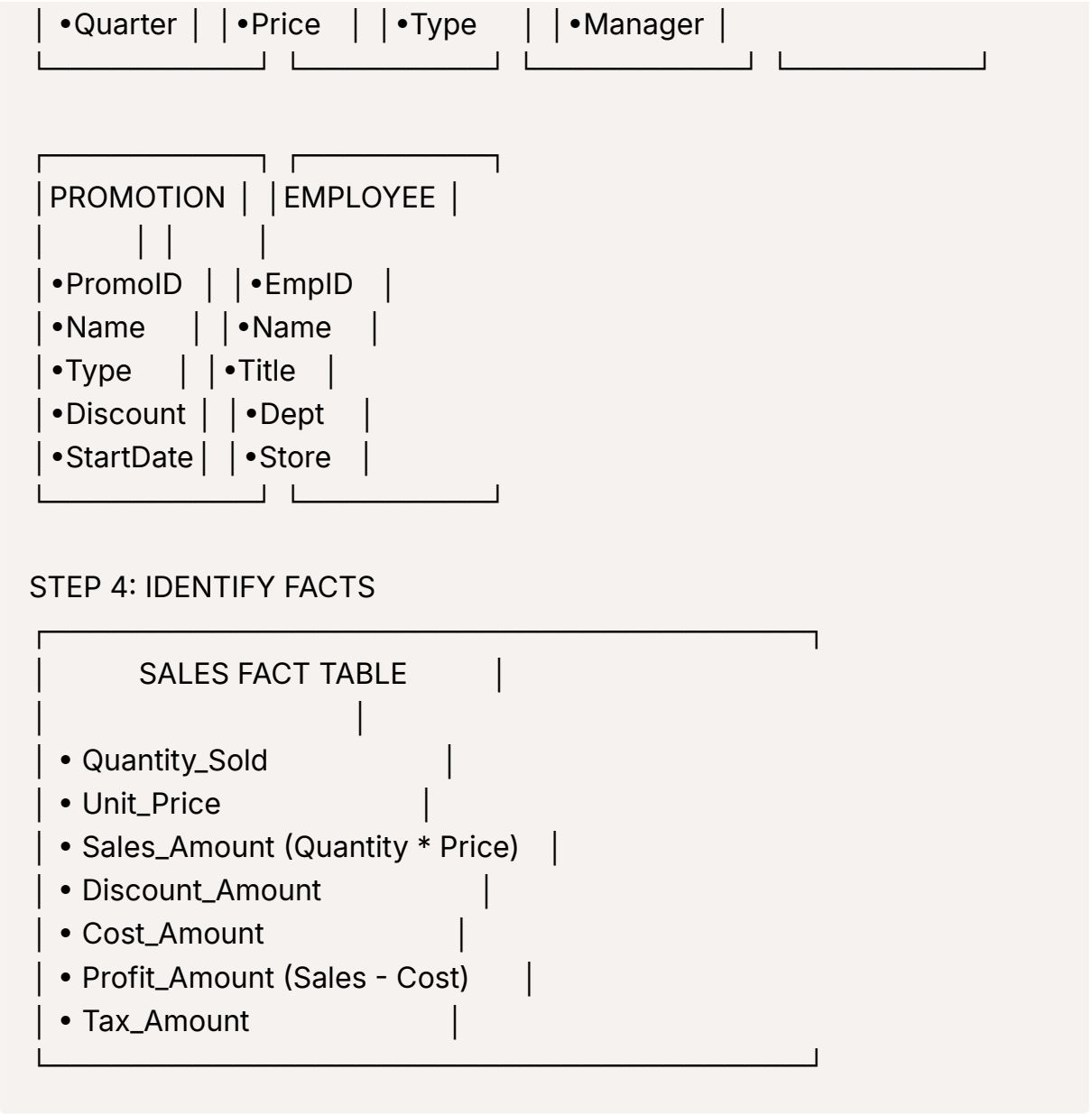
- Numerical values that can be aggregated
- Additive, semi-additive, or non-additive
- Answer "How much?" or "How many?"

Common Fact Types:

- **Transaction Facts**: Quantity sold, unit price, sales amount
- **Snapshot Facts**: Account balance, inventory level
- **Accumulating Facts**: Days to ship, total process time
- **Factless Facts**: Event occurrence (promotion applied)

Detailed Example: Retail Sales Process





Alternative Example: Online Orders Process

Step 1: Business Process: Processing Online Orders

Step 2: Grain: Each product ordered (line item level)

Step 3: Dimensions:

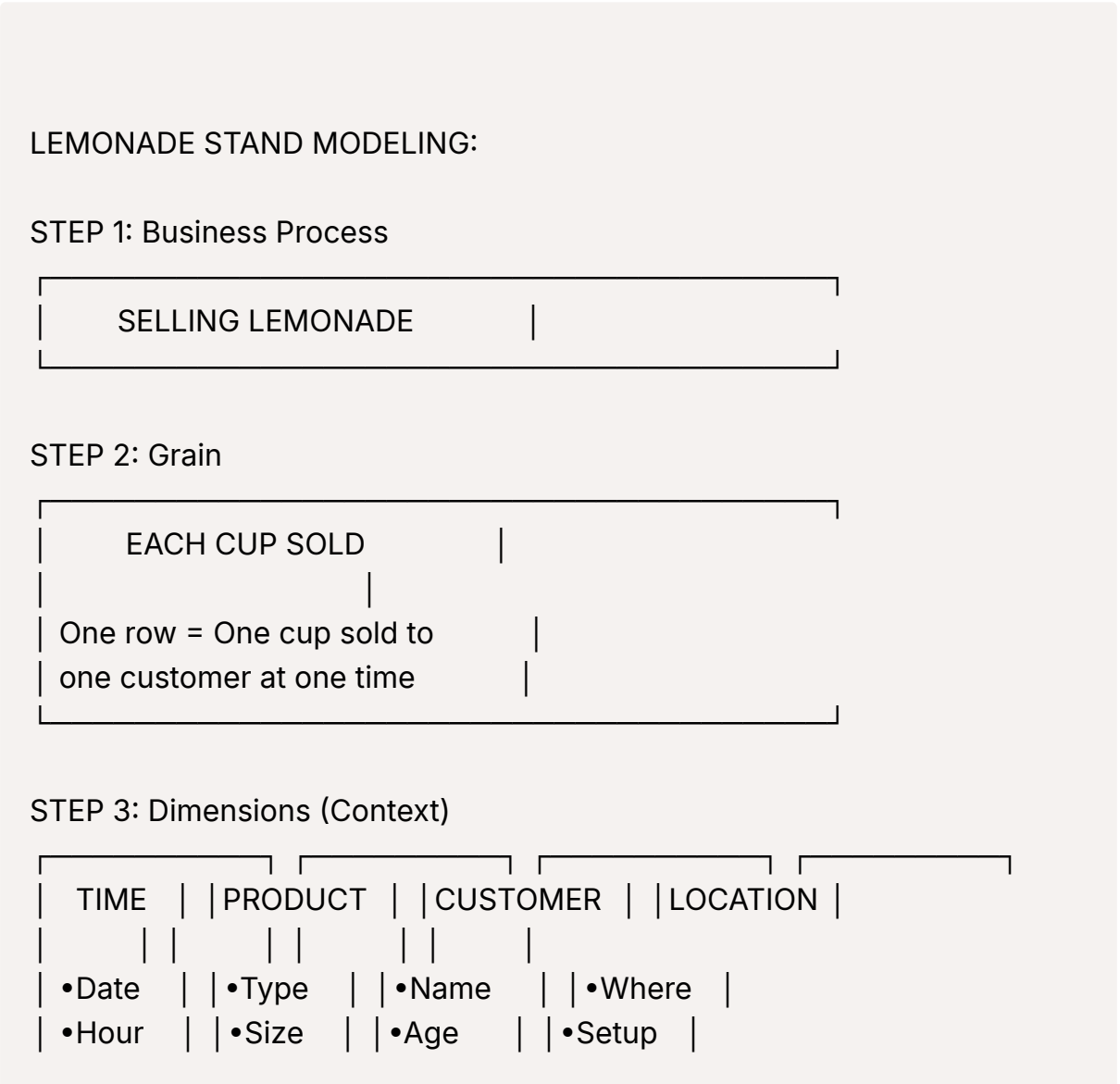
- Date (order date, ship date)
- Product (item ordered)
- Customer (who ordered)
- Payment Method (how paid)

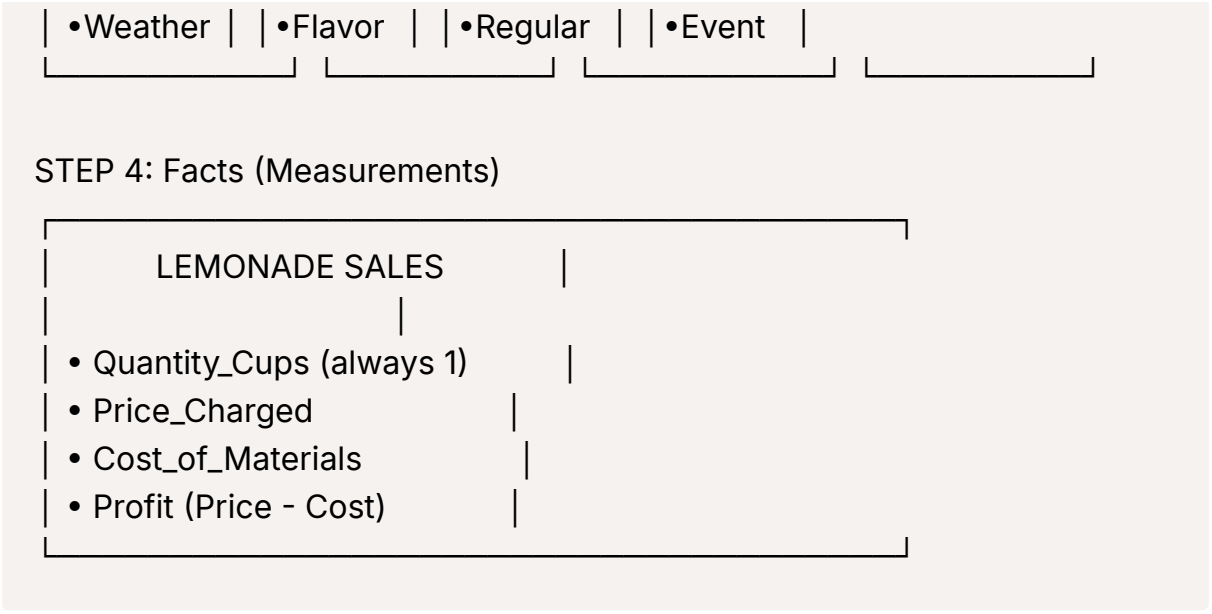
- Shipping Method (delivery type)
- Promotion (discounts applied)

Step 4: Facts:

- Quantity ordered
- Unit price
- Extended price
- Shipping cost
- Tax amount
- Discount amount

Lemonade Stand Example





Design Principles

Grain Selection Guidelines

- **Start with most atomic level** (detailed grain)
- **Consider future requirements** - easier to aggregate than disaggregate
- **Balance detail vs. performance** - more detail = larger tables
- **Ensure consistent grain** across all facts in same table

Dimension Design Best Practices

- **Include descriptive attributes** for drill-down analysis
- **Use meaningful business names** not technical codes
- **Consider hierarchies** (Year → Quarter → Month → Day)
- **Plan for slowly changing dimensions**

Fact Design Best Practices

- **Choose numeric, additive measures** when possible
- **Include calculated measures** that support business analysis
- **Ensure facts match the grain** - no facts more detailed than grain
- **Consider fact table types** (transaction, snapshot, accumulating)

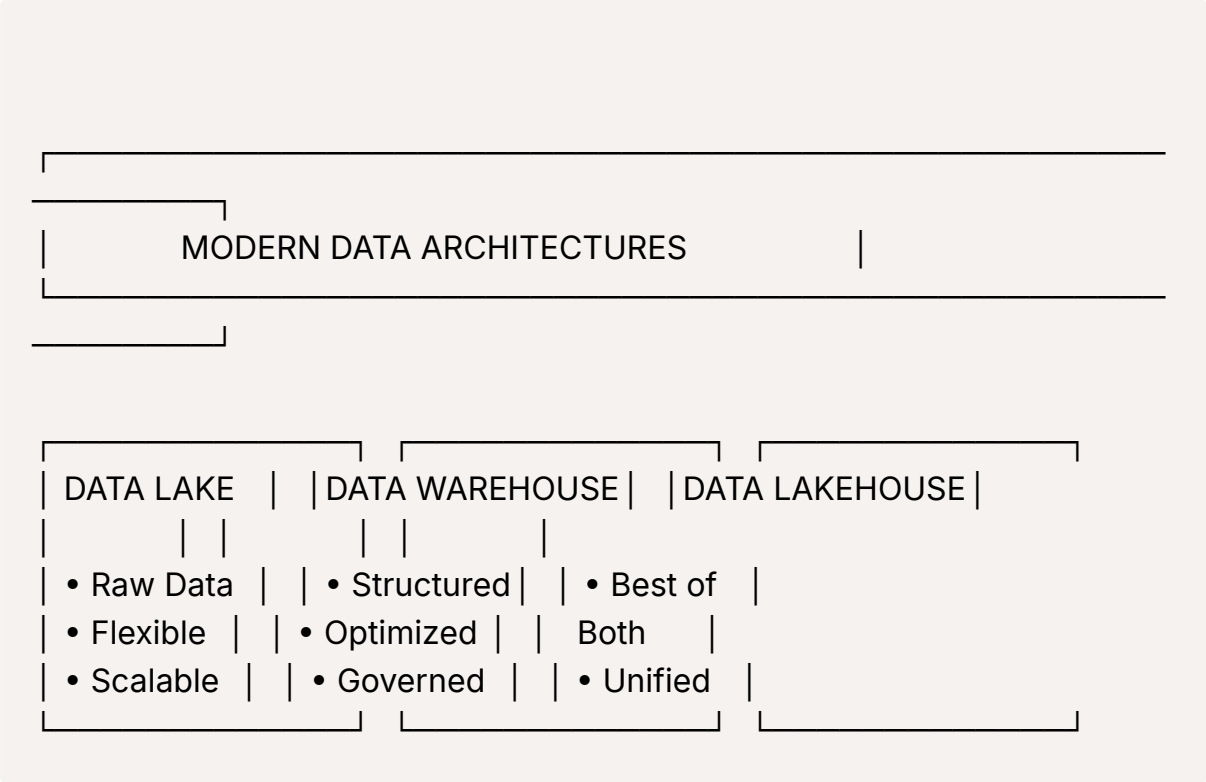
Common Modeling Mistakes

1. **Unclear Grain**: Not precisely defining what each row represents
2. **Mixing Grains**: Having facts at different detail levels in same table
3. **Missing Dimensions**: Not including important context for analysis
4. **Non-Additive Facts**: Including percentages or ratios without base measures
5. **Too Many Facts**: Including every possible measure rather than focusing on key metrics

---

12. ADVANCED TOPICS

Modern Data Architecture Landscape



1. Data Vault 2.0

Definition: Data modeling methodology designed for auditing, tracing data back to source, and handling historical changes.

Key Components:

- **Hubs**: Core business entities (Customer, Product)
- **Links**: Relationships between hubs

- **Satellites**: Descriptive attributes and historical changes

Benefits:

- Highly auditable and traceable
- Handles complex business relationships
- Supports agile development
- Excellent for regulatory compliance

2. OBT (One Big Table)

Definition: Concept where data from various sources/dimensions is flattened into a single, very wide table for analytical purposes.

Characteristics:

- Denormalized structure
- All dimensions and facts in one table
- Optimized for specific analytical queries
- Trade-off between storage and query performance

Use Cases:

- Specific reporting requirements
- Performance optimization for known queries
- Self-service analytics platforms

3. Streaming vs Batch Processing

```
BATCH PROCESSING:
Data → Collect → Process in Batches → Store → Analyze
     (Hours/Days)    (Scheduled)

STREAMING PROCESSING:
Data → Process Continuously → Real-time Results
     (Milliseconds/Seconds)
```

Batch Processing

- Processes data in large volumes periodically
- Higher latency, higher throughput
- Better for complex transformations
- Examples: ETL jobs, monthly reports

Streaming Processing

- Processes data continuously as it arrives
- Lower latency, lower throughput per record
- Better for real-time requirements
- Examples: Fraud detection, live dashboards

4. Data Architecture Decision Matrix

```
┌──────────────────────────────────────────────────────────────────────────┐
│       WHEN TO CHOOSE WHICH ARCHITECTURE                                  │
└──────────────────────────────────────────────────────────────────────────┘

DATA LAKE - Choose When:
┌──────────────────────────────────────────────────────────────────────────┐
│ • Early data maturity stage                                              │
│ • Use cases not known upfront                                            │
│ • Need to store diverse data types                                       │
│ • Exploratory analytics focus                                            │
│ • Cost optimization priority                                             │
└──────────────────────────────────────────────────────────────────────────┘

DATA WAREHOUSE - Choose When:
┌
```

```
┌──────────────────────────────────────────────────────────────┐
│ • Well-defined reporting requirements                        │
│ • Consistent, structured data                                │
│ • High-performance BI/dashboards needed                      │
│ • Strong governance and compliance requirements              │
│ • Traditional analytics focus                                │
└──────────────────────────────────────────────────────────────┘
```

DATA LAKEHOUSE - Choose When:

```
┌──────────────────────────────────────────────────────────────┐
│ • Need both flexibility and performance                      │
│ • Real-time and batch processing requirements                │
│ • Scaling governed pipelines on lake foundation              │
│ • Want to avoid full DW overhead                             │
│ • Modern cloud-native architecture                           │
└──────────────────────────────────────────────────────────────┘
```

5. Data Virtualization

Definition: Approach to access data from multiple sources without replicating it, providing unified view.

Benefits:

- Real-time access to source systems
- Reduced data movement and storage
- Faster time to insights
- Simplified architecture

Challenges:

- Performance limitations
- Network dependency
- Complex query optimization
- Limited transformation capabilities

6. Data Mesh

Definition: Modern data architecture focusing on decentralized, domain-oriented data ownership.

Core Principles:

- **Domain-oriented decentralization**: Each business domain owns its data
- **Data as a product**: Treat data products with product thinking
- **Self-serve data infrastructure**: Platform providing common capabilities
- **Federated computational governance**: Standardized governance across domains

7. Normalization Types (Quick Reference)

First Normal Form (1NF)

- Each column contains atomic values
- No repeating groups
- Each row is unique

Second Normal Form (2NF)

- Must be in 1NF
- All non-key attributes fully dependent on primary key
- No partial dependencies

Third Normal Form (3NF)

- Must be in 2NF
- No transitive dependencies
- Non-key attributes depend only on primary key

Normalization vs. Denormalization:

- **OLTP systems**: Highly normalized (3NF) for data integrity
- **Data Warehouses**: Often denormalized for query performance

8. Advanced Interview Topics

Behavioral Questions

- "Describe a time you chose between different data architecture patterns"
- "How did you handle a data quality issue in a previous project?"
- "Explain a complex dimensional model you designed"

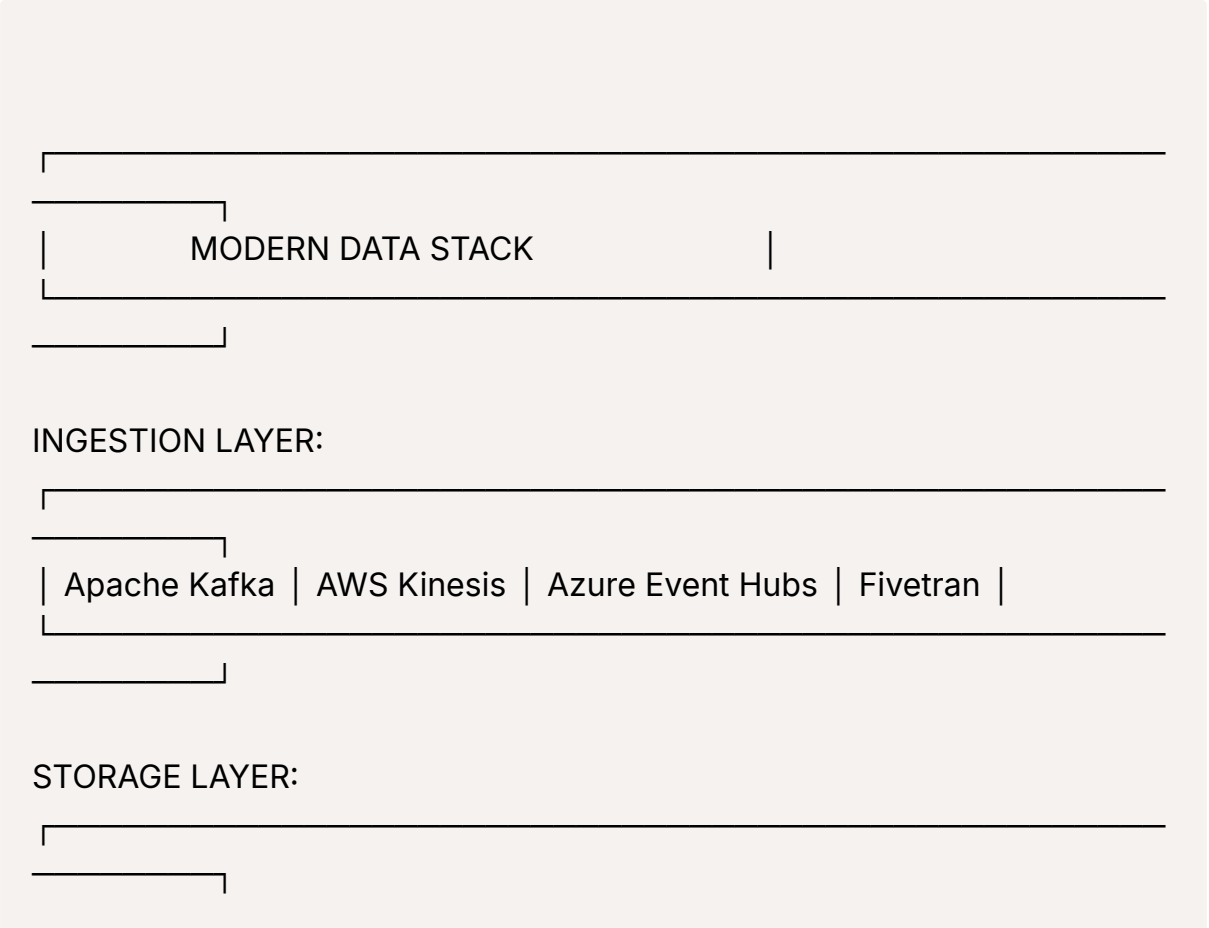
Scenario-Based Questions

- "Design a data architecture for a multi-channel retailer"
- "How would you handle real-time fraud detection requirements?"
- "What approach for customer 360-degree view implementation?"

Architecture Decisions

- "Compare pros/cons of star vs. snowflake for specific use case"
- "When would you recommend ELT over ETL?"
- "How to choose between Data Lake and Data Warehouse?"

9. Modern Technology Stack



```
│ S3/ADLS │ Snowflake │ Databricks │ BigQuery │ Redshift   │
└──────────────────────────────────────────────────────────────
──────────┘

PROCESSING LAYER:
┌──────────────────────────────────────────────────────────────
──────────┐
│ Apache Spark │ dbt │ Apache Airflow │ Prefect │ Dagster   │
└──────────────────────────────────────────────────────────────
──────────┘

CONSUMPTION LAYER:
┌──────────────────────────────────────────────────────────────
──────────┐
│ Tableau │ Power BI │ Looker │ ThoughtSpot │ Jupyter     │
└──────────────────────────────────────────────────────────────
──────────┘
```

10. Future Trends and Considerations

Emerging Technologies

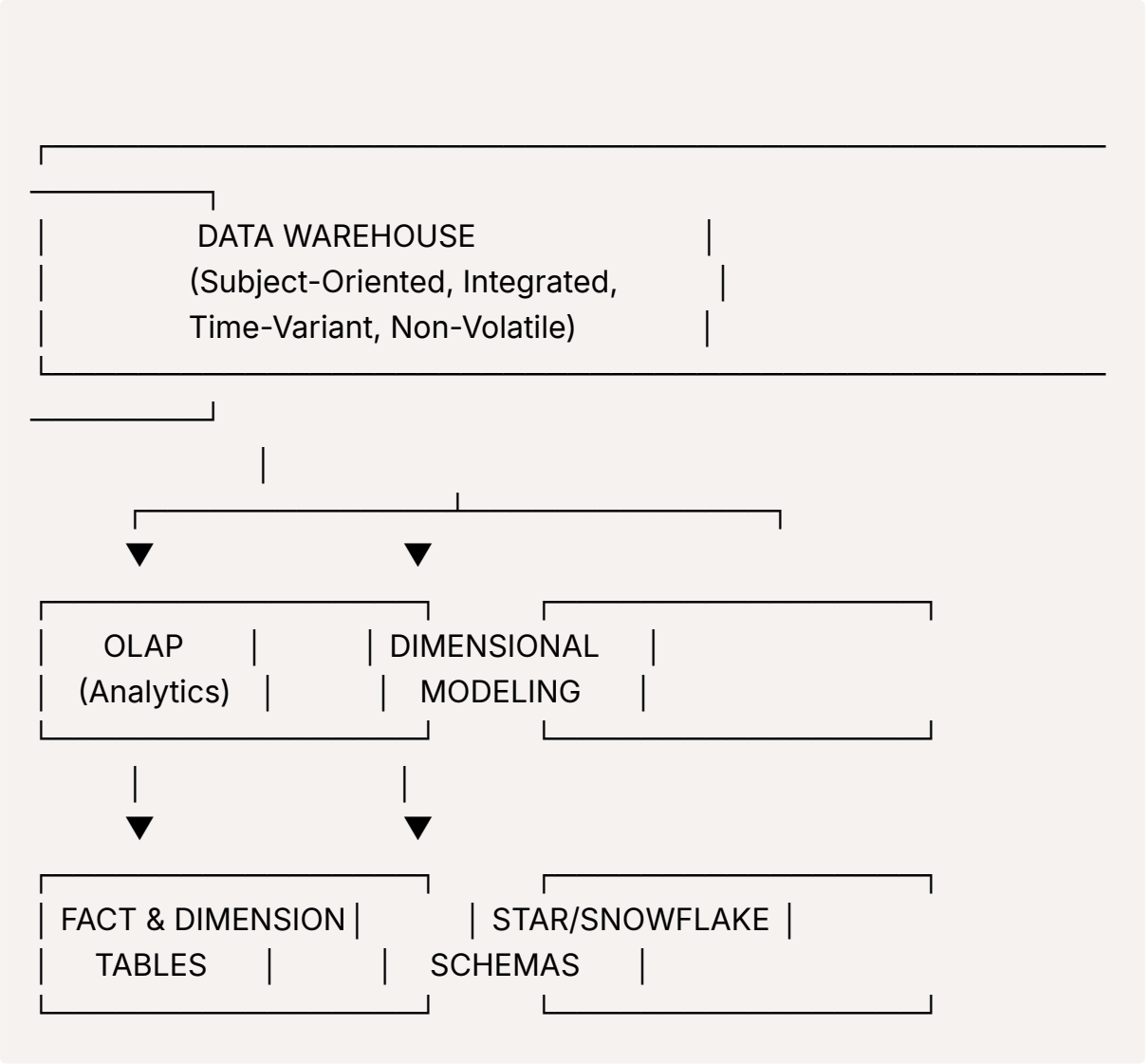
- **AI/ML Integration**: Native AI capabilities in data platforms
- **Real-time Analytics**: Stream processing becoming standard
- **Data Observability**: Monitoring data quality and lineage
- **Cloud-Native**: Everything designed for cloud-first

Governance Evolution

- **Data Contracts**: Formal agreements between data producers/consumers
- **Privacy by Design**: Built-in privacy protection
- **Automated Compliance**: Tools for regulatory adherence
- **Data Lineage**: End-to-end tracking of data flow

SUMMARY AND KEY TAKEAWAYS

Core Concepts Hierarchy



Essential Remember Points

1. **Data Warehouse**: Subject-oriented, integrated, time-variant, non-volatile
2. **OLTP vs OLAP**: Operational vs Analytical processing
3. **ETL vs ELT**: Transform location determines approach
4. **Facts vs Dimensions**: Measurements vs Context
5. **Star vs Snowflake**: Denormalized vs Normalized dimensions
6. **SCD Types**: How to handle dimension changes over time
7. **Schema on Write/Read**: When to apply data structure
8. **4-Step Modeling**: Process → Grain → Dimensions → Facts

Decision Framework

Use this guide to make architectural decisions:

1. **Understand Business Requirements** → Choose appropriate architecture
2. **Define Data Maturity Level** → Select suitable technologies
3. **Consider Performance Needs** → Design optimal schema
4. **Plan for Growth** → Implement scalable solutions
5. **Ensure Governance** → Establish proper controls

This study guide provides comprehensive coverage of data warehousing fundamentals through advanced concepts. Regular review of these concepts will build strong foundation for data engineering and analytics roles.