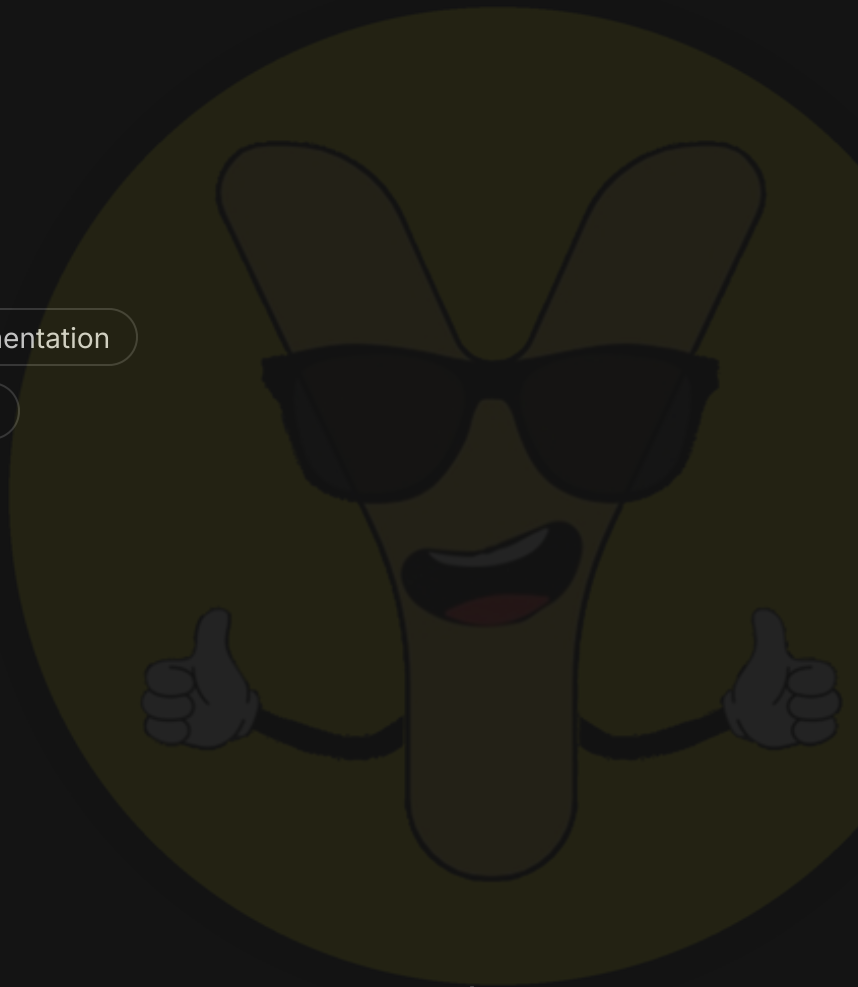
AKTU · 3RD YEAR · SEMESTER 5 · BCS058

Data Warehouse & Data Mining

Unit 2 — Data Warehouse Process and Technology

- Warehousing Strategy
- Warehouse Management and Support Processes
- Warehouse Planning and Implementation
- Hardware and Operating Systems for Data Warehousing
- Client/Server Computing Model & Data Warehousing
- Parallel Processors & Cluster Systems
- Distributed DBMS Implementations
- Warehousing Software
- Warehouse Schema Design



DECODED BY

Chirag Jain

Founder, Ycotes · Gold Medalist in Artificial Intelligence, IIT Madras · taught 10,000+ students

theycotes.com

Warehousing Strategy: Overview

DATA WAREHOUSING STRATEGY

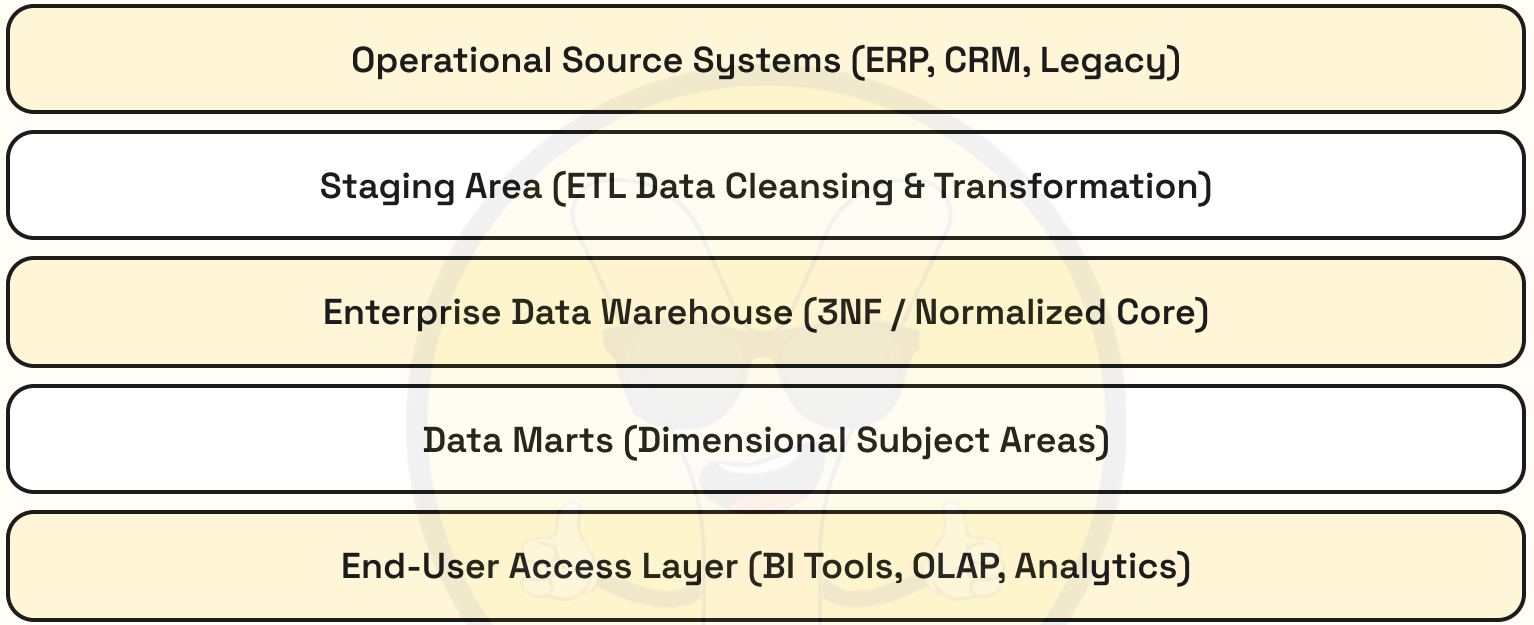
A Data Warehousing Strategy is a structured plan defining how an organization collects, integrates, manages, and utilizes enterprise data to support business intelligence and strategic decision-making.

- Aligns data warehouse goals directly with enterprise business objectives and ROI metrics.
- Determines enterprise architecture choice: Top-down enterprise warehouse vs. Bottom-up data marts.
- Establishes data governance frameworks, security standards, and metadata ownership.
- Defines technology stack, scalability options, and long-term hardware acquisition plans.

Warehousing Strategy: Inmon vs. Kimball

Parameter	Inmon Approach (Top-Down)	Kimball Approach (Bottom-Up)
Philosophy	Enterprise-wide single source of truth	Dimensional data marts for business processes
Data Model	Normalized 3NF relational data model	Dimensional star/snowflake schemas
Development	High initial effort, longer setup time	Iterative setup, fast initial delivery
Redundancy	Low data redundancy	Higher redundancy in dimension tables
Cost	High upfront investment	Lower initial cost, incremental expense
Flexibility	High flexibility for new requirements	Requires rebuilding marts for structural changes

Data Warehouse Architecture & Strategy Layers



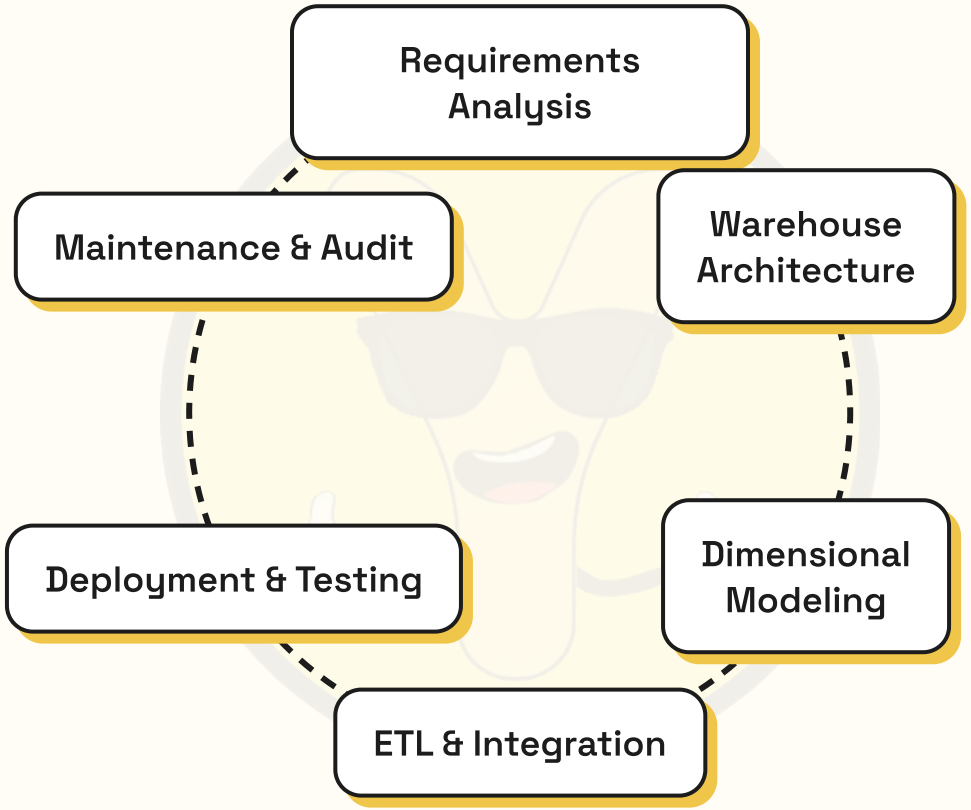
Strategic Data Flow from Operational Sources to Analytics Layer

Warehouse Management and Support Processes

Core administrative and support processes maintain warehouse performance, security, and data freshness.

- Data Load & Refresh Management
  - Schedules batch and incremental ETL loading jobs.
  - Monitors transformation job failures and data latency.
- Data Archiving & Purging
  - Moves aged historical data to low-cost near-line storage.
  - Purges temporary staging tables to free disk capacity.
- Backup & Disaster Recovery
  - Executes cold/hot database backups and transaction logs.
  - Ensures high availability with Recovery Point Objective (RPO) targets.
- Metadata Management
  - Maintains technical and business metadata dictionaries.
  - Tracks data lineage from operational source to BI reports.

Warehouse Implementation Lifecycle



Iterative Cycle of Planning, Building, and Managing a Data Warehouse

Warehouse Planning and Implementation Steps

A disciplined methodology minimizes project risk and ensures alignment with enterprise goals.

- Construct business case, define return on investment (ROI), and secure executive sponsorship.
- Gather detailed user requirements to map business metrics and key performance indicators.
- Perform capacity planning estimating initial data volume and multi-year growth rate $V_{total} = V_{init} \times (1 + g)^t$.
- Conduct Technical Proof of Concept (PoC) to evaluate DBMS, ETL, and hardware suitability.
- Select deployment methodology: Big-Bang deployment vs. Incremental data-mart release strategy.

Hardware & Operating Systems for Data Warehousing

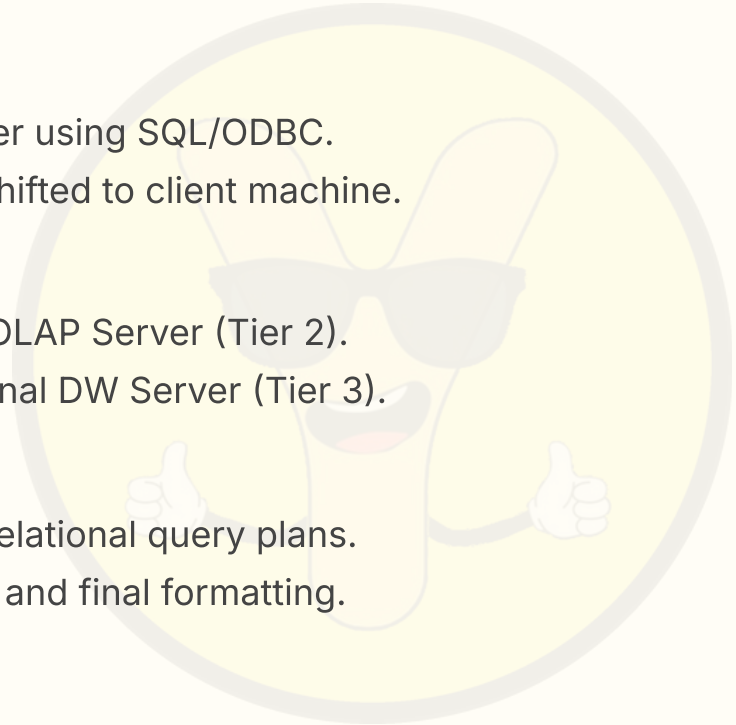
Data warehouses demand high disk I/O throughput, large memory capacity, and robust multi-processor OS support.

- Memory & Storage Infrastructure
  - Requires large RAM to cache massive dimensional joins.
  - High I/O bandwidth using RAID 10 or NVMe storage pools.
- 64-bit Operating Systems
  - Schedules parallel kernel threads efficiently across CPU cores.
  - Supports addressable memory spaces beyond 4 GB limits.
- Multi-Processor Server Hardware
  - SMP and MPP processor layouts accelerate complex query scans.
  - High-speed interconnect buses prevent system bottlenecks.

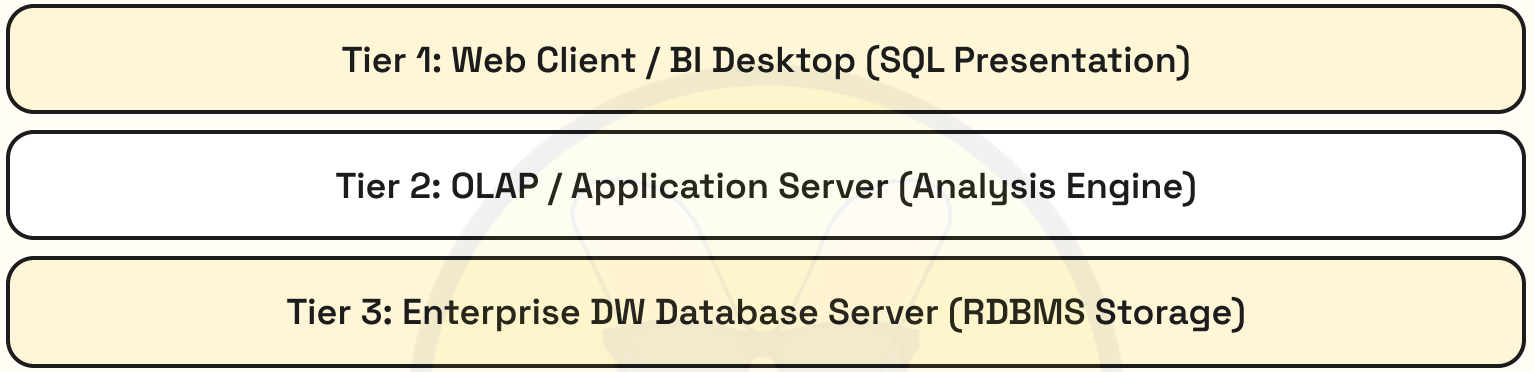
Client/Server Computing Model in DW

Data warehousing relies on multi-tier client/server architectures to distribute query processing workloads.

- Two-Tier Architecture
  - Client directly queries central DW server using SQL/ODBC.
  - High network traffic; processing load shifted to client machine.
- Three-Tier Architecture
  - Client (Tier 1) connects to Application/OLAP Server (Tier 2).
  - OLAP Server queries underlying relational DW Server (Tier 3).
- Server Roles
  - Warehouse server executes intensive relational query plans.
  - Client application handles UI rendering and final formatting.



3-Tier Client/Server Architecture for DW



Three-Tier Processing Model separating Presentation, Business Logic, and Storage

Parallel Processors & Cluster Systems

Parallel processing accelerates data warehouse execution by dividing large data scans across multiple processors.

- Symmetric Multiprocessing (SMP)
  - Shared memory, shared disk architecture controlled by single OS.
  - Limited scalability due to memory bus contention.
- Massively Parallel Processing (MPP)
  - Shared-nothing architecture with independent nodes.
  - Each node contains dedicated CPU, memory, and disk storage.
- Cluster Systems
  - Interconnected independent commodity servers operating as one.
  - Cost-effective alternative to proprietary MPP hardware.

Comparison of Parallel Architectures

Feature	Shared Memory (SMP)	Shared Disk	Shared Nothing (MPP)
Memory Access	Shared across all CPUs	Private per node	Private per node
Disk Access	Shared across all CPUs	Shared across all nodes	Private per node
Scalability Limit	Limited (typically < 64 CPUs)	Medium (up to 128 nodes)	High (1000s of nodes)
Interconnect Load	High memory bus traffic	High disk channel traffic	Low (Message Passing)
Cost	High (Specialized Hardware)	Moderate to High	Low to Moderate (Clusters)
Fault Tolerance	Single point of OS failure	Node failover supported	High node independence

Speedup and Scaleup Metrics

Parallel system efficiency is evaluated using mathematical speedup and scaleup formulas.

- Speedup Concept
  - Measures time reduction when adding processors for fixed problem size.
  - Formula: $Speedup = \frac{T_1}{T_p}$ where T_1 is single-CPU time, T_p is p-CPU time.
- Linear Speedup
  - Ideal condition where running on p processors takes $\frac{1}{p}$ time.
  - Real systems suffer overhead due to communication and synchronization.
- Scaleup Concept
  - Measures capability to maintain time when workload and processors grow proportionally.
  - Formula: $Scaleup = \frac{T_1}{T_p(p \times Workload)}$.
- Batch vs Transaction Scaleup
  - Data warehousing focuses on Batch Scaleup (larger query sizes).

Worked Example: Speedup Calculation

A data warehouse scan query takes 120 seconds on a single-processor system. The same query is executed on a parallel system with 8 processors and takes 18 seconds. Calculate the Speedup and the Parallel Efficiency.

1. Step 1: Write Speedup Formula: $Speedup = \frac{T_1}{T_p}$
2. Step 2: Substitute values: $Speedup = \frac{120}{18} = 6.67$
3. Step 3: Write Efficiency Formula: $Efficiency = \frac{Speedup}{p} \times 100\%$
4. Step 4: Calculate Efficiency: $Efficiency = \frac{6.67}{8} \times 100\% = 83.375\%$

The Speedup achieved is 6.67 (out of maximum ideal 8.0), and the Parallel Efficiency is 83.38%.

Distributed DBMS Implementations in DW

Distributed DBMS spreads data across multiple physical locations while providing a unified database view.

- Data Fragmentation
  - Horizontal: Splitting rows across sites (e.g., region-wise sales).
  - Vertical: Splitting attributes across sites to optimize access.
- Data Replication
  - Stores duplicate copies of dimension tables across nodes.
  - Reduces cross-network join latency during query execution.
- Distributed Query Optimization
  - Minimizes data transfer costs over network interconnects.
  - Executes local joins before transmitting partial aggregates.
- Distributed DW vs Centralized DW
  - Centralized offers easier maintenance; Distributed offers local autonomy.

Warehousing Software and Tool Categories

Data warehousing ecosystems consist of software specialized for extraction, storage, and analytics.

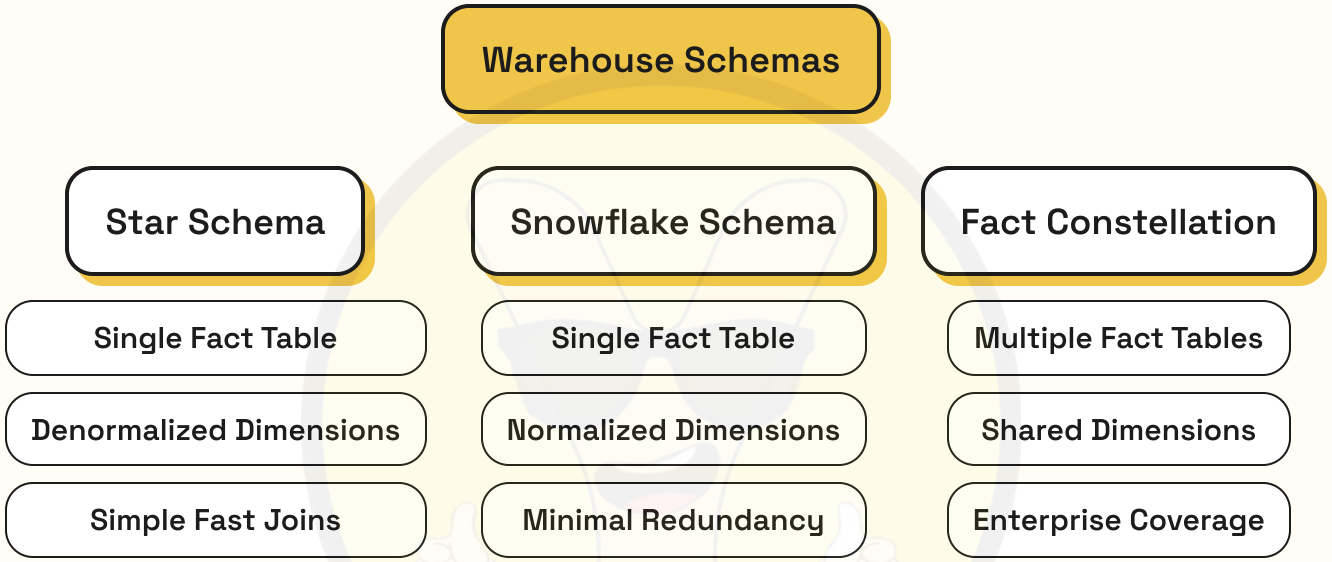
- ETL Tools
  - Extracts, transforms, and loads operational data into warehouse.
  - Examples: Informatica PowerCenter, Talend, IBM InfoSphere.
- DBMS & OLAP Engines
  - Relational engines optimized for read-heavy column scans.
  - MOLAP, ROLAP, and HOLAP engines for multi-dimensional analysis.
- Metadata Repositories
  - Stores definitions, transformations, business rules, and lineage.
- BI & Query Tools
  - Provides dashboards, ad-hoc queries, and visual reporting.
  - Examples: Tableau, PowerBI, MicroStrategy.

Warehouse Schema Design Fundamentals

Dimensional modeling organizes data into central facts and descriptive dimensions for intuitive querying.

- Fact Table
  - Central table containing quantitative numerical measures and foreign keys.
  - Example metrics: sales_amount, quantity_sold, profit.
- Dimension Table
  - Contains descriptive context attributes for filtering and grouping.
  - Example attributes: customer_name, store_region, product_category.
- Surrogate Keys
  - System-generated integer keys replacing operational primary keys.
  - Protects warehouse from source key changes and supports SCDs.
- Granularity
  - Defines the level of detail represented in a fact table record.

Warehouse Schema Design Types



Taxonomy of Data Warehouse Schema Architectures

Star Schema vs. Snowflake Schema

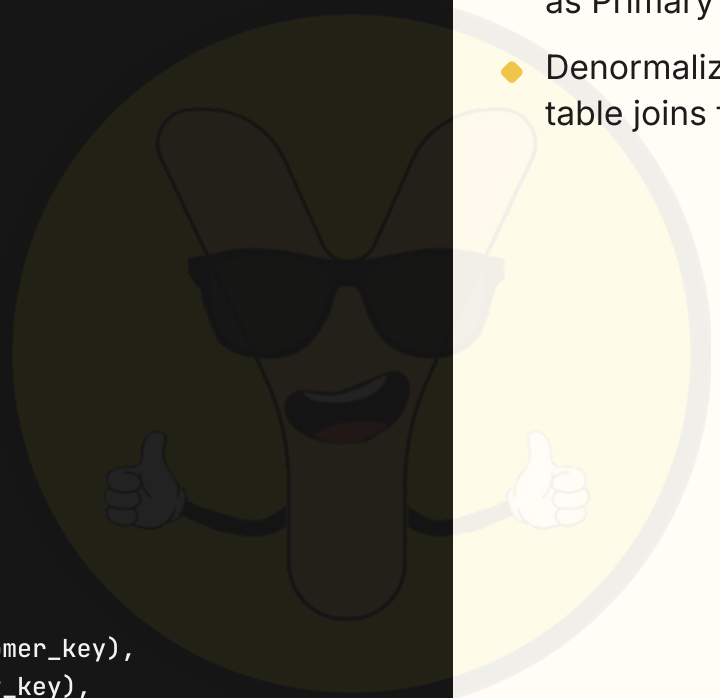
Feature	Star Schema	Snowflake Schema
Dimension Structure	Fully Denormalized	Normalized into lookup sub-tables
Query Complexity	Low (Fewer SQL JOIN operations)	High (Multiple SQL JOIN operations)
Storage Space	Higher redundancy, uses more space	Lower redundancy, space optimized
Data Maintenance	Complex updates due to redundancy	Easier updates in normalized tables
Performance	Faster query execution speeds	Slower due to multiple join overheads
Ease of Use	Simple for business users to understand	Complex structure for end-users

SQL Implementation: Star Schema Design

```
-- Dimension Table: Customer
CREATE TABLE dim_customer (
    customer_key INT PRIMARY KEY,
    customer_id VARCHAR(20),
    customer_name VARCHAR(100),
    region VARCHAR(50)
);

-- Dimension Table: Product
CREATE TABLE dim_product (
    product_key INT PRIMARY KEY,
    product_name VARCHAR(100),
    category VARCHAR(50)
);

-- Central Fact Table: Sales
CREATE TABLE fact_sales (
    sales_id INT PRIMARY KEY,
    customer_key INT REFERENCES dim_customer(customer_key),
    product_key INT REFERENCES dim_product(product_key),
    quantity INT,
    sales_amount DECIMAL(10,2)
);
```



- Uses surrogate integer keys (customer_key, product_key) as Primary & Foreign keys.
- Denormalized dimension tables reduce complex multi-table joins for fast queries.

Exam Corner: Likely AKTU Questions

- **Q1** Differentiate between Inmon's Top-Down approach and Kimball's Bottom-Up approach for data warehouse design.
- **Q2** Explain the 3-Tier Client/Server Architecture of a Data Warehouse with a neat diagram.
- **Q3** Compare Shared Memory (SMP), Shared Disk, and Shared Nothing (MPP) parallel processor architectures.
- **Q4** Define Speedup and Scaleup in parallel data processing. Calculate speedup if a single CPU takes 120s and an 8-CPU system takes 18s.
- **Q5** What is dimensional modeling? Compare Star Schema and Snowflake Schema with suitable diagrams and examples.
- **Q6** Describe the management and support processes involved in maintaining an enterprise Data Warehouse.
- **Q7** Write SQL DDL commands to implement a Star Schema consisting of one Fact table and two Dimension tables.

Unit 2 done. You are exam-ready.

Revise the Exam corner once more, then open Unit 3 on theycotes.com. Every AKTU subject, unit by unit, free. Notes by Chirag Jain, Ycotes.

