CHAPTER 6

DATABASE NORMALIZATION & DATA WAREHOUSING

Normalization organises relational databases to **minimise redundancy** and ensure **data integrity** (1NF → 2NF → 3NF). **Data warehousing** stores & analyses **historical data** for business intelligence — unlike **OLTP** systems that handle real-time transactions. Also: the **ETL** process and **data marts**.

CHAPTER ROADMAP

1. Database normalization — 1NF · 2NF · 3NF
2. Data warehousing — 4 characteristics
3. Online transaction processing — OLTP
4. Extract, Transform, Load — ETL
5. Data marts — dependent · independent

Database management is the backbone of data-driven organisations — efficient **storage, organisation and retrieval** of structured data. A **DBMS** provides a structured framework to store, retrieve and manipulate data while ensuring **security, consistency and integrity**. Key concepts: normalization, data warehousing, ETL, DBMS architecture, and **OLTP** systems (real-time transactions) vs **data warehouses** (analytical processing).

1 DATABASE NORMALIZATION

minimise redundancy · ensure integrity

DEFINITION

A systematic process of organising data in a relational database to minimise redundancy and ensure data integrity.

Breaks **large, unwieldy tables** into **smaller, logically related** ones with clear relationships → reduces risk of **anomalies** during insertions, updates or deletions.

Why normalize?

- **Redundancy** — same data in multiple places wastes space & increases risk of **inconsistency**.
- **Anomalies** — updating, inserting or deleting leads to errors (e.g., **orphaned records**, inconsistent values).

Normal Forms — each builds on the previous one

1NF
Atomic values
Goal: eliminate repeating groups; ensure **atomicity** (indivisible values in each column).

2NF
No partial dependency
Prereq: 1NF. Non-key attributes depend on the **whole** (composite) primary key.

3NF
No transitive dependency
Prereq: 2NF. Attributes depend **directly** on the primary key, not via another attribute.

1.1 FIRST NORMAL FORM (1NF)

Rules

- All data in a column is the **same type** (no mixing strings & numbers).
- **No repeating groups or arrays** in one column (e.g., not a list of phone numbers).
- Each row **uniquely identifiable** (typically a **primary key**).

× VIOLATES 1NF — "PRODUCTSORDERED" HOLDS MANY VALUES

OrderID	CustomerName	ProductsOrdered
1	John Doe	Laptop, Mouse, Keyboard
2	Jane Smith	Monitor, Printer

FIX Create a **separate row for each product** so every field holds a single, indivisible (**atomic**) value.

✓ IN 1NF

OrderID	CustomerName	ProductOrdered
1	John Doe	Laptop
1	John Doe	Mouse
1	John Doe	Keyboard
2	Jane Smith	Monitor
2	Jane Smith	Printer

1.2 SECOND NORMAL FORM (2NF)

remove partial dependencies

Goal
All non-primary-key attributes depend **fully** on the primary key.

Prerequisite
Table must already be in **1NF**.

Rule
No attribute depends on **only part of a composite primary key**.

1NF TABLE + CUSTOMERADDRESS · COMPOSITE KEY = ORDERID + PRODUCTORDERED

OrderID 🔑	CustomerName	ProductOrdered 🔑	CustomerAddress
1	John Doe	Laptop	123 Main St
1	John Doe	Mouse	123 Main St
2	Jane Smith	Monitor	456 Oak Ave

Partial dependency: CustomerName & CustomerAddress depend **only on OrderID**, not on ProductOrdered.

✓ 2NF — SPLIT INTO TWO TABLES

ORDERS

OrderID	ProductOrdered
1	Laptop
1	Mouse
2	Monitor

CUSTOMERS

OrderID	Name	Address
1	John Doe	123 Main St
2	Jane Smith	456 Oak Ave

1.3 THIRD NORMAL FORM (3NF)

remove transitive dependencies

Goal
Eliminate **transitive dependencies** — non-key attributes depending on **other non-key attributes**.

Prerequisite
Table must be in **2NF**.

Rule
All attributes depend **directly on the primary key**, not indirectly through another attribute.

CUSTOMERS + CUSTOMERCITY

OrderID 🔑	CustomerName	CustomerAddress	CustomerCity
1	John Doe	123 Main St	New York
2	Jane Smith	456 Oak Ave	Chicago

OrderID › CustomerAddress › CustomerCity

City depends on the address, **not directly on OrderID** → transitive dependency.

✓ 3NF — SPLIT FURTHER

CUSTOMERS

OrderID	CustomerName	AddressID
1	John Doe	1
2	Jane Smith	2

ADDRESSES

AddressID	CustomerAddress	CustomerCity
1	123 Main St	New York
2	456 Oak Ave	Chicago

✓ **Benefits** — reduced redundancy · fewer anomalies · easier maintenance.

× **Trade-offs** — increased complexity (more tables) · potentially slower queries due to **joins**.

MEMORY HOOK Every non-key attribute must depend on **the key** (1NF), **the whole key** (2NF), and **nothing but the key** (3NF).

2 DATA WAREHOUSING

historical data for BI

DEFINITION

A specialised type of database designed to store and analyse large volumes of historical data, supporting business intelligence (BI) and decision-making.

Unlike **operational databases** (current transactions), data warehouses are built for **querying and reporting over extended timeframes**.

2.1 Subject-Oriented

Organised around **business areas** (sales, inventory, HR), not operational processes.

e.g. "Sales" subject area: transactions, products, customers — ignores employee schedules.

2.2 Integrated

Data from diverse sources (CRM, ERP, flat files) **cleaned & unified** into a consistent format.

e.g. Oracle sales + SQL Server customers merged; "CustomerID" standardised.

2.3 Time-Variant

Stores **historical data** (e.g., 5–10 years), not just current snapshots → **trend analysis**.

e.g. Monthly sales 2015–2025 vs operational DB with only this month.

2.4 Non-Volatile

Once loaded, data **remains unchanged (read-only)** → stable base for analysis.

e.g. Last year's sales preserved as new data is added.

MEMORY HOOK **"SITN"** — **S**ubject-oriented · **I**ntegrated · **T**ime-variant · **N**on-volatile.

3 ONLINE TRANSACTION PROCESSING (OLTP)

real-time transactions

OLTP systems **manage and process transactional data in real-time** — optimised for a **large number of short, fast, atomic transactions** (insert, update, delete). Critical for **day-to-day operations**: processing orders, managing inventory, handling customer interactions.

OLTP
Optimised for **transactional processing** & **real-time updates**

vs

OLAP
Optimised for **complex data analysis** & **reporting**

Key Characteristics of OLTP

① Real-Time Processing
Data updated immediately — online order recorded instantly.

② High Transaction Volume
Many small concurrent transactions — thousands of orders/second in a sale.

③ Atomic Transactions
Single unit of work; failure → **rolled back** (failed payment cancels order).

④ Normalized Structure
Highly normalized — customers, orders, products in separate tables.

⑤ Integrity & Concurrency
Constraints (PK, FK); two users can't book the **same flight seat**.

⑥ Short & Fast Queries
Small amounts of data — checking a product's availability.

Examples

- **Banking** — ATM, deposits, withdrawals
- **E-commerce** — orders, payments, inventory
- **Airline reservations** — bookings, seats, tickets
- **Retail POS** — sales & inventory updates

Advantages

- **Real-time data access**
- **High performance** under heavy loads
- **Data consistency** (atomic + normalized)
- **Scalability** — many concurrent users

Challenges

- **Complexity** — normalized DBs hard to design
- **Limited analytical capabilities** — not for complex / historical analysis
- **Resource intensive** hardware & software

Feature	OLTP Systems	Data Warehousing
Purpose	Real-time transaction processing	Historical data analysis
Data Volume	Small, current datasets	Large, historical datasets
Data Structure	**Highly normalized**	**Typically denormalized**
Operations	Insert, update, delete	Read-only queries
Performance	Optimised for fast transactions	Optimised for complex queries
Query Complexity	Simple, short queries	Complex, long-running queries
Data Integrity	Enforces strict integrity constraints	Focuses on data consistency for analysis
Example Use Case	Processing online orders	Analysing sales trends over time

4 EXTRACT, TRANSFORM, LOAD (ETL)

backbone of data warehousing

Transforms raw data from **disparate sources** into a **structured, analysis-ready** format, ensuring **data quality and consistency**.

Source Systems → E› → Staging Area → T› → Transformed data → L› → Presentation System (warehouse)

EXTRACT

Collect data from **databases** (MySQL, PostgreSQL), **APIs**, **flat files** (CSV, Excel), **enterprise systems** (SAP, Salesforce).

Challenges: different formats, data availability, extraction frequency.

e.g. daily sales from regional stores + CRM feedback

→

TRANSFORM

Raw data **cleaned, standardised, enriched**:

- Remove duplicates & nulls
- Convert types (MM/DD/YYYY → YYYY-MM-DD)
- Apply business rules (total = qty × price)
- Merge sources (link customer IDs)

e.g. euros → dollars

→

LOAD

Transformed data **transferred into the warehouse**:

Initial load — all historical data at once (e.g., 5 years of sales).

Incremental load — only new/updated records (e.g., daily).

e.g. into a "Sales" fact table

Consistency
Unifies heterogeneous data

Accuracy
Cleansing → reliable insights

Efficiency
Fast querying & reporting

EXAMPLE APPLICATION
ETL combines **sales, inventory and supplier** data into a warehouse → reports on **stock levels and profitability**.

5 DATA MARTS

focused subset of a warehouse



A **subset of a data warehouse** tailored to a **specific business unit or subject area** (e.g., marketing, finance) — provides **faster access** to relevant data for targeted analysis.

EXAMPLE
Marketing data mart stores **campaign performance** data, ignoring unrelated HR metrics.

DEPENDENT
Built **from an existing data warehouse** (sales mart from corporate warehouse)

vs

INDEPENDENT
Created **directly from source systems**, bypassing a central warehouse

Speed
Smaller scope → faster queries

Focus
Customised for departmental needs

EXAM FOCUS · MCQ TRAPS

commonly confused points

1 Which NF fixes what?
1NF = **repeating groups / atomicity**. 2NF = **partial** dependency. 3NF = **transitive** dependency.

2 Partial vs transitive
Partial = depends on **part of a composite key**. Transitive = non-key depends on **another non-key** (City → Address).

3 Prerequisites
2NF requires 1NF; 3NF requires 2NF — each form **builds on the previous one**.

4 Normalization trade-off
More tables → **more joins** → potentially slower queries & more complexity.

5 Normalized vs denormalized
OLTP = **highly normalized**. Data warehouse = **typically denormalized**.

6 Time-variant vs non-volatile
Time-variant = keeps **history** (5–10 yrs). Non-volatile = **read-only**, not overwritten.

7 OLTP vs OLAP
OLTP = real-time transactions. OLAP = complex **analysis & reporting**.

8 ETL stage tasks
Removing duplicates, converting date formats, currency conversion = **Transform** (not Load).

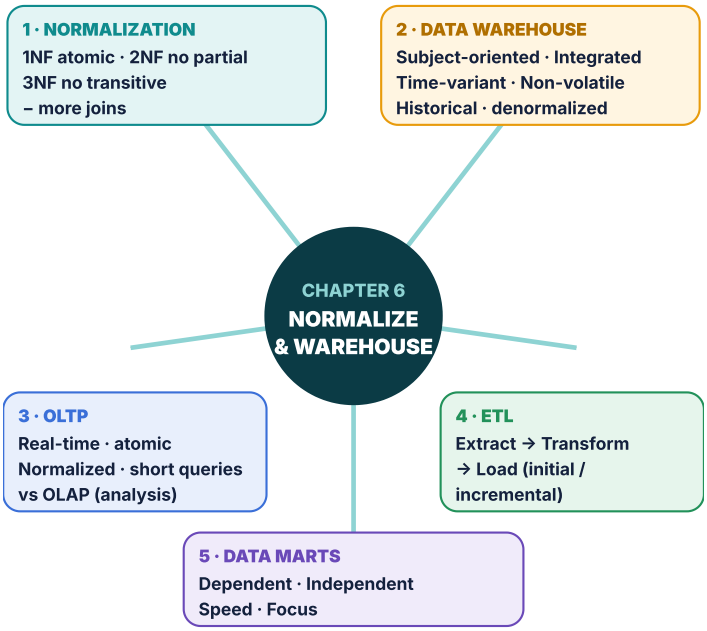
9 Initial vs incremental load
Initial = all history at once. Incremental = only **new / updated** records.

10 Dependent vs independent mart
Dependent = **from the warehouse**. Independent = **from source systems**.

CLASSIFY IT

1	"Laptop, Mouse" in one cell	Breaks 1NF
2	Address depends on part of composite key	Breaks 2NF
3	City depends on Address	Breaks 3NF
4	Stores 2015–2025 monthly sales	Time-variant
5	Oracle + SQL Server data unified	Integrated
6	ATM withdrawal	OLTP
7	Euros converted to dollars	Transform
8	Daily sales updates added	Incremental load
9	Mart built from source systems	Independent mart

REVISION MAP



Normalize
Smaller related tables, less redundancy & anomalies.

1–2–3 NF
Atomic → whole key → nothing but the key.

Warehouse
SITN: historical, read-only, for BI.

OLTP
Real-time, atomic, normalized, fast & short.

ETL & Marts
Feed the warehouse; marts give focused speed.