data warehouse — Digest

Definition

A data warehouse is a centralized store of structured, historical data — typically modeled with facts and dimensions (star/snowflake schema) — organized specifically to support analytical querying and reporting, as opposed to transactional operations.

Real-life usage

- Retail/e-commerce: `fact_sales` (quantity, revenue) joined to `dim_product`, `dim_store`, `dim_date`, `dim_customer` — slice revenue any way, on demand.
- Banking/fintech: regulatory reporting (risk exposure, AML) needs consistent historical snapshots, not just current state.
- Airlines: `fact_bookings`/`fact_flights` joined to `dim_route`, `dim_aircraft`, `dim_time` to analyze load factors and revenue per route over years.
- Marketing: `fact_impressions`/`fact_conversions` joined to `dim_campaign`, `dim_channel`, `dim_audience`.
- Analogy: a physical warehouse doesn't manufacture goods — it receives finished goods and organizes them onto labeled shelves by category (dimensions) with clear counts (facts), so anyone can pull exactly what they need without digging through the factory floor.

5 Whys — Root Cause

- Why not just use a regular RDBMS with a star schema? Row-store databases store full rows together on disk — a query needing 2 of 30 columns still has to read all 30 off disk.
- **Root cause 1 — columnar storage:** warehouse engines store each column contiguously, so aggregation queries read only the columns they need. This also enables better compression and vectorized execution.
- Why also separate facts and dimensions if columnar storage already solves read efficiency? Flattening everything into one wide table causes massive redundancy (a customer's address repeated across every transaction row) and update anomalies (changing one dimension value means rewriting/missing rows across a billion-row fact

table).
- **Root cause 2 — fact/dimension separation:** avoids redundancy and update anomalies by separating high-volume immutable events (facts) from low-volume mutable reference data (dimensions), each managed according to its actual rate of change (see: slowly changing dimensions).

Value vs. Risk

- **Value:** understanding columnar storage + fact/dimension separation changes how you design any analytical store — not just SQL warehouses. E.g., a medallion architecture built on doc-oriented stores (like Elasticsearch) still needs to factor out dimension keys instead of embedding full dimension attributes in every aggregate document, or it silently rebuilds the flat-table problem.
- **Risk — no dedicated warehouse layer:** flat/embedded dimension data means a dimension change (e.g., product recategorized) can't be propagated without reindexing every historical record referencing it — leading to silent inconsistency between old and new records with no single source of truth to detect or correct the drift.

Revision #1
Created 31 August 2026 09:01:11 by N8N
Updated 31 August 2026 09:01:12 by N8N