

Data Engineering Glossary

Definitions for every term used across this handbook — one place to look things up.

Last reviewed 29 Sep 2026 ·

Prerequisites: None — good place to start

Related: [DE Concepts](https://sarangambekar1997.github.io/data-engineering-handbook/00-foundations/de-concepts/) (https://sarangambekar1997.github.io/data-engineering-handbook/00-foundations/de-concepts/)

A

ABAC (Attribute-Based Access Control) — Access policies evaluated against attributes of the data (such as classification tags) and of the user (such as group or region), so one policy can govern thousands of tables. See [Governance & Lineage](https://sarangambekar1997.github.io/data-engineering-handbook/05-quality-governance/governance-lineage/) (https://sarangambekar1997.github.io/data-engineering-handbook/05-quality-governance/governance-lineage/).

Accumulating Snapshot Fact — A fact table pattern where one row tracks an entire business process lifecycle (e.g., one row per order that gets updated with shipped_at, delivered_at as events occur). Contrast with transaction facts.

ANN (Approximate Nearest Neighbor) — A search algorithm that finds vectors similar to a query vector without scanning all vectors. Trades tiny accuracy loss for large speed gains. Used in vector databases.

Apache Arrow — A columnar in-memory data format shared by many engines (pandas, Polars, DuckDB, Spark), allowing data to move between them with little or no copying.

Architecture Decision Record (ADR) — A short document that captures one significant architecture decision: the context, the options considered, the choice, its consequences, and the signals that would reopen it. It is kept in the repository beside the code. See [Choosing a Stack](https://sarangambekar1997.github.io/data-engineering-handbook/08-architecture/choosing-a-stack/) (https://sarangambekar1997.github.io/data-engineering-handbook/08-architecture/choosing-a-stack/).

Aspect and URN (DataHub) — In DataHub, an aspect is one facet of an entity (ownership, tags, schema) and the smallest unit that can be written, and a URN is the stringified key of an entity, such as `urn:li:dataset:`

`(urn:li:dataPlatform:postgres,shop.public.orders,PROD)` . See [Data Catalogs in Practice](https://sarangambekar1997.github.io/data-engineering-handbook/05-quality-governance/data-catalogs/) (https://sarangambekar1997.github.io/data-engineering-handbook/05-quality-governance/data-catalogs/).

Asset Check — A validation attached to a data asset (e.g., no negative amounts) whose pass or fail result is recorded next to the asset. A *blocking* check stops downstream assets when it fails. See [Dagster](https://sarangambekar1997.github.io/data-engineering-handbook/03-orchestration/dagster-reference/) (https://sarangambekar1997.github.io/data-engineering-handbook/03-orchestration/dagster-reference/).

Asset (Software-Defined Asset) — In Dagster, a declaration that a table, file or model should exist, together with the code that produces it and its upstream dependencies. The orchestrator tracks each asset's materializations, checks and lineage. See [Dagster](https://sarangambekar1997.github.io/data-engineering-handbook/03-orchestration/dagster-reference/) (https://sarangambekar1997.github.io/data-engineering-handbook/03-orchestration/dagster-reference/).

Avro — A row-based binary data format with schema embedded in the file. Used widely in Kafka for its schema evolution support. Contrast with Parquet (columnar).

B

Backfill — Re-running a pipeline for past time periods, typically to populate historical data or fix incorrect past runs.

Blameless Postmortem — A written review of an incident that identifies contributing causes in the system without blaming any individual or team, and ends in action items with owners. It assumes people acted reasonably with what they knew, which makes them willing to surface problems. See [DataOps](https://sarangambekar1997.github.io/data-engineering-handbook/05-quality-governance/dataops-operations/) (https://sarangambekar1997.github.io/data-engineering-handbook/05-quality-governance/dataops-operations/).

Bronze Layer — The first layer in medallion architecture. Stores raw, unmodified data exactly as it arrived from source systems.

BM25 — A keyword-based document ranking algorithm used in search engines. The "B" in hybrid search (B = BM25, V = vector). More accurate than TF-IDF for sparse keyword queries.

C

Capacity Unit (CU, Microsoft Fabric) — The measure of compute power in a Microsoft Fabric capacity. SKUs are sized in CUs (an F64 has 64), and every workload in the attached workspaces draws on the same pool. See [Azure and Fabric](https://sarangambekar1997.github.io/data-engineering-handbook/01-storage/azure-fabric/) (https://sarangambekar1997.github.io/data-engineering-handbook/01-storage/azure-fabric/).

Cardinality — The number of unique values in a column. High cardinality = many unique values (user IDs, email addresses). Low cardinality = few unique values (status codes, boolean flags). Affects index effectiveness and partition strategies.

CDC (Change Data Capture) — A technique for capturing row-level changes (INSERT, UPDATE, DELETE) from a source database in real time, typically via database logs. Used to replicate data to a warehouse or data lake.

Change Data Feed (CDF) — A Delta Lake feature that records row-level changes (insert, update pre/post image, delete) so downstream jobs can process only what changed. See [Delta Lake](https://sarangambekar1997.github.io/data-engineering-handbook/01-storage/delta-lake/) (https://sarangambekar1997.github.io/data-engineering-handbook/01-storage/delta-lake/).

Change Stream (MongoDB) — A feed of change events (insert, update, delete and others) from a MongoDB replica set or sharded cluster, each carrying a resume token so a consumer can continue after a restart. It is MongoDB's change-data-capture mechanism. See [NoSQL and Operational Stores](https://sarangambekar1997.github.io/data-engineering-handbook/01-storage/nosql-operational-stores/) (https://sarangambekar1997.github.io/data-engineering-handbook/01-storage/nosql-operational-stores/).

Checkpoint — In Spark Structured Streaming, a directory where Spark saves offsets and state so it can resume from the exact position after a restart.

Chunk — A piece of a larger document, split to fit within an LLM's context window for embedding or retrieval. Typical size: 256–512 tokens.

Cluster (Spark) — A group of machines that run a Spark job together: one driver and multiple executors.

Cluster Key (Snowflake) — Columns used to organize micro-partitions in Snowflake for faster pruning on large tables. Similar to a sort key.

Consumer Group (Kafka) — A set of Kafka consumers that collectively read from a topic. Each partition is assigned to exactly one consumer in the group. Enables parallel consumption and horizontal scaling.

Copy-on-Write (CoW) — A table-format update strategy that rewrites the whole data file when a row changes. Reads are fast and writes are heavier. Contrast with Merge-on-Read. See [Apache Hudi](https://sarangambekar1997.github.io/data-engineering-handbook/01-storage/apache-hudi/) (https://sarangambekar1997.github.io/data-engineering-handbook/01-storage/apache-hudi/).

Crypto-shredding — Making data unrecoverable by deleting its encryption key — used to honour deletion requests in immutable storage and backups.

CTE (Common Table Expression) — A named temporary result set defined within a SQL query using the `WITH` keyword. Makes complex queries more readable.

Credits (Snowflake) — The unit of compute cost in Snowflake. Each virtual warehouse size consumes credits per hour of active use.

D

DAG (Directed Acyclic Graph) — A graph where edges have direction and no cycles. In Airflow, a DAG represents a workflow where tasks are nodes and dependencies are edges.

Data Catalog — A searchable inventory of datasets with technical, operational, and business metadata such as schemas, owners, descriptions, lineage, and usage.

Data Contract — A formal agreement between a data producer and consumer specifying schema, semantics, quality guarantees, and SLA.

Data Diff — A comparison of a change's output with the current production output, listing the rows added, removed and changed, so the effect of a code change is visible before it merges. See [Testing and CI/CD](https://sarangambekar1997.github.io/data-engineering-handbook/06-infrastructure/testing-cicd/) (https://sarangambekar1997.github.io/data-engineering-handbook/06-infrastructure/testing-cicd/).

Data Downtime — Periods when data is missing, late, wrong or otherwise unusable. The failure that data observability aims to detect and shorten. See [Pipeline Observability](https://sarangambekar1997.github.io/data-engineering-handbook/05-quality-governance/pipeline-observability/) (https://sarangambekar1997.github.io/data-engineering-handbook/05-quality-governance/pipeline-observability/).

Data Lake — A storage system (usually object storage like S3) that holds raw data in any format without enforcing schema on write.

Data Lakehouse — A hybrid architecture combining the flexibility of a data lake with the ACID guarantees and performance of a data warehouse. Built on open table formats (Delta Lake, Iceberg, Hudi).

Data Lineage — A record of data's origin, movement, and transformation history — showing where data came from and what happened to it.

Data Mart — A subset of a data warehouse focused on a specific business domain (e.g., sales mart, finance mart).

Data Mesh — An organizational approach in which domain teams own and publish their data as products — with contracts, SLAs, and documentation — on a shared self-service platform. See [System Design](https://sarangambekar1997.github.io/data-engineering-handbook/08-architecture/system-design/) (https://sarangambekar1997.github.io/data-engineering-handbook/08-architecture/system-design/).

Data Steward — The person responsible for maintaining a dataset's definitions, classifications, and documentation on behalf of its owner.

Data Vault — A modeling methodology for enterprise data warehouses using Hubs (business keys), Links (relationships), and Satellites (attributes + history).

Data Warehouse — A centralized, structured analytical data store optimized for read-heavy query workloads. Examples: Snowflake, BigQuery, Redshift.

DBU (Databricks Unit) — The unit of Databricks compute cost. One DBU is one unit of processing capability per hour.

Dead Letter Queue (DLQ) — A queue where messages that fail processing are routed for later inspection and reprocessing.

Debezium — An open-source change data capture platform that reads database transaction logs and emits row-level change events, usually through Kafka Connect. See [Ingestion & CDC](https://sarangambekar1997.github.io/data-engineering-handbook/02-processing/ingestion-cdc/) (https://sarangambekar1997.github.io/data-engineering-handbook/02-processing/ingestion-cdc/).

Deduplication — Removing duplicate records, either exact duplicates or near-duplicates (semantic deduplication using embeddings).

Delta Lake — An open-source storage layer that brings ACID transactions, schema enforcement, time travel, and versioning to data lake files. Created by Databricks.

Deployment (Prefect) — The packaging of a flow with a schedule, parameters and a work pool, so Prefect can run it remotely or on a schedule. See [Prefect](https://sarangambekar1997.github.io/data-engineering-handbook/03-orchestration/prefect-reference/) (https://sarangambekar1997.github.io/data-engineering-handbook/03-orchestration/prefect-reference/).

Dimension (semantic layer) — An attribute used to group or filter a metric, such as date, country or status. See [Semantic Layer & Metrics](https://sarangambekar1997.github.io/data-engineering-handbook/02-processing/semantic-layer-metrics/) (https://sarangambekar1997.github.io/data-engineering-handbook/02-processing/semantic-layer-metrics/).

Dimension Table — In a star schema, a table that provides descriptive context for facts (who, what, where, when). Examples: dim_customer, dim_product, dim_date.

Distribution Key (Redshift) — The column whose hash determines which node slice stores each row. Matching distribution keys on large joined tables avoids moving data during joins. See [Amazon Redshift](https://sarangambekar1997.github.io/data-engineering-handbook/01-storage/redshift-reference/) (https://sarangambekar1997.github.io/data-engineering-handbook/01-storage/redshift-reference/).

Driver (Spark) — The JVM process that runs the `main()` function of a Spark application. Coordinates executors, builds the execution plan, and collects results.

DuckDB — An in-process analytical SQL database that queries Parquet, CSV, and JSON files directly and handles larger-than-memory data on a single machine. See [DuckDB & Polars](https://sarangambekar1997.github.io/data-engineering-handbook/02-processing/duckdb-polars/) (https://sarangambekar1997.github.io/data-engineering-handbook/02-processing/duckdb-polars/).

E

Egress — Data transferred out of a cloud provider or region. Usually billed per GB and often the largest cost in multi-cloud or cross-region designs.

ELT (Extract, Load, Transform) — A modern data integration pattern: data is extracted from sources, loaded raw into the destination, then transformed using the destination's compute power (e.g., dbt on Snowflake). Contrast with ETL.

Embedding — A dense vector (list of floats) that represents the semantic meaning of a piece of text, image, or other data. Semantically similar items have similar vectors.

Error Budget — The amount of failure an SLO allows over a period (e.g., about 3 late days a year at a 99% SLO). See [Pipeline Observability](https://sarangambekar1997.github.io/data-engineering-handbook/05-quality-governance/pipeline-observability/) (https://sarangambekar1997.github.io/data-engineering-handbook/05-quality-governance/pipeline-observability/).

ETL (Extract, Transform, Load) — A traditional data integration pattern: data is extracted, transformed before loading, then loaded into the destination. Contrast with ELT.

Event Time — The time an event actually happened, as opposed to processing time, when the pipeline sees it. Windows defined on event time give results that reflect when things occurred, even when data arrives late or out of order. See [Beam and Dataflow](https://sarangambekar1997.github.io/data-engineering-handbook/04-streaming/beam-dataflow/) (https://sarangambekar1997.github.io/data-engineering-handbook/04-streaming/beam-dataflow/).

Execution Accuracy — A text-to-SQL metric: a generated query is correct when it returns the same rows as a reference query, whatever its text, with row order ignored unless the reference orders its rows. See [MCP and Text-to-SQL](https://sarangambekar1997.github.io/data-engineering-handbook/07-ai/mcp-text-to-sql/) (https://sarangambekar1997.github.io/data-engineering-handbook/07-ai/mcp-text-to-sql/).

Executor (Spark) — A JVM process on a worker node that runs tasks. Each executor has a number of cores and a memory allocation.

F

Fact Table — In a star schema, the central table that stores measurable business events (orders, clicks, payments). Contains foreign keys to dimensions and numeric measures.

Fan-out — A messaging pattern where one message or event triggers multiple independent downstream consumers or processes.

Feature Store — A centralized repository for ML features — precomputed, versioned, and shareable across models and teams.

Federated Query — A query that reads from more than one data source, such as a lake and a database, in a single statement, without copying the data first. Its efficiency depends on how much work the connectors push down to each source. See [Trino](https://sarangambekar1997.github.io/data-engineering-handbook/02-processing/trino-federation/) (https://sarangambekar1997.github.io/data-engineering-handbook/02-processing/trino-federation/).

FinOps — The practice of making cloud spend visible, attributable, and efficient through collaboration between engineering, finance, and business teams. See [Cost Optimization](https://sarangambekar1997.github.io/data-engineering-handbook/08-architecture/cost-optimization/) (https://sarangambekar1997.github.io/data-engineering-handbook/08-architecture/cost-optimization/).

Fitness Function — An automated check that a system keeps a desired architectural property, such as "no streaming layer without a freshness requirement" or "every layer has an owner". It turns a principle into something that fails a build. See [Choosing a Stack](https://sarangambekar1997.github.io/data-engineering-handbook/08-architecture/choosing-a-stack/) (https://sarangambekar1997.github.io/data-engineering-handbook/08-architecture/choosing-a-stack/).

Flow (Prefect) — A Python function decorated with `@flow` that Prefect runs, tracks and can schedule. It can contain tasks and other flows. See [Prefect](https://sarangambekar1997.github.io/data-engineering-handbook/03-orchestration/prefect-reference/) (https://sarangambekar1997.github.io/data-engineering-handbook/03-orchestration/prefect-reference/).

Freshness SLA — A commitment that data in a table will be available within a defined time window (e.g., "gold layer data available by 6am UTC").

G

Global Secondary Index (GSI) — In DynamoDB, a second copy of a table's data keyed by different attributes, so another access pattern can be served by a key lookup. It costs extra writes and storage, and it is updated asynchronously. See [NoSQL and Operational Stores](https://sarangambekar1997.github.io/data-engineering-handbook/01-storage/nosql-operational-stores/) (https://sarangambekar1997.github.io/data-engineering-handbook/01-storage/nosql-operational-stores/).

Gold Layer — The final layer in medallion architecture. Contains business-ready, aggregated tables consumed by BI tools, APIs, and ML models.

Grain — The level of detail in a fact table — what one row represents. E.g., "one row per order" or "one row per order line item per day". Must be defined explicitly.

Granule (ClickHouse) — A block of rows, 8192 by default, that the sparse primary index points to. A query skips the granules whose key range cannot match its filter. See [Real-Time Analytics Databases](https://sarangambekar1997.github.io/data-engineering-handbook/01-storage/realtime-olap/) (https://sarangambekar1997.github.io/data-engineering-handbook/01-storage/realtime-olap/).

H

HMAC (Keyed Hash) — A hash computed with a secret key. Used to pseudonymise identifiers so the same input always gives the same token, while an attacker without the key cannot reverse it by hashing guesses. See [Data Security & Privacy](https://sarangambekar1997.github.io/data-engineering-handbook/05-quality-governance/data-security-privacy/) (https://sarangambekar1997.github.io/data-engineering-handbook/05-quality-governance/data-security-privacy/).

HNSW (Hierarchical Navigable Small World) — The most common ANN index algorithm used in vector databases. Builds a multi-layer graph structure for fast approximate search.

Hot Partition — A partition that receives a disproportionate share of traffic, so it reaches its throughput limit while the rest of the table is idle. Caused by low-cardinality or skewed keys. In DynamoDB each partition is designed for at most 3,000 read units and 1,000 write units per second. See [NoSQL and Operational Stores](https://sarangambekar1997.github.io/data-engineering-handbook/01-storage/nosql-operational-stores/) (https://sarangambekar1997.github.io/data-engineering-handbook/01-storage/nosql-operational-stores/).

Hudi (Apache Hudi) — An open table format built for record-level upserts and incremental queries, with Copy-on-Write and Merge-on-Read table types. See [Apache Hudi](https://sarangambekar1997.github.io/data-engineering-handbook/01-storage/apache-hudi/) (https://sarangambekar1997.github.io/data-engineering-handbook/01-storage/apache-hudi/).

HyDE (Hypothetical Document Embeddings) — A RAG retrieval technique: generate a hypothetical answer to the question, embed it, and use that vector to search. Improves recall when queries are vague.

Idempotent — An operation that produces the same result whether run once or many times. Critical for reliable pipeline design — re-running an idempotent pipeline doesn't create duplicates.

Incremental Load — A pipeline pattern that processes only new or changed data since the last run, rather than reprocessing everything.

Incremental View Maintenance (IVM) — Keeping a materialized view current by applying each change as a signed delta (a retraction and an insertion for an update), instead of recomputing the whole query. Streaming databases such as RisingWave and Materialize work this way. See [Streaming SQL](https://sarangambekar1997.github.io/data-engineering-handbook/04-streaming/streaming-sql/) (https://sarangambekar1997.github.io/data-engineering-handbook/04-streaming/streaming-sql/).

I

Iceberg (Apache Iceberg) — An open table format for huge analytic datasets. Like Delta Lake but more portable — supported by Spark, Flink, Trino, Snowflake, and others.

Incident Commander — The person who holds the overall state of an incident, structures the response and assigns roles, without fixing the problem personally. Other roles are the operations lead, communications and planning. See [DataOps](https://sarangambekar1997.github.io/data-engineering-handbook/05-quality-governance/dataops-operations/) (https://sarangambekar1997.github.io/data-engineering-handbook/05-quality-governance/dataops-operations/).

IVFFlat — A vector index algorithm that partitions vectors into clusters (inverted file) and searches only nearby clusters. Faster than brute force, lower memory than HNSW.

J

Job (Kubernetes) — A Kubernetes object that runs pods to completion and retries failures. It is the unit of batch work, and a CronJob creates Jobs on a schedule. See [Kubernetes](https://sarangambekar1997.github.io/data-engineering-handbook/06-infrastructure/kubernetes-for-de/) (https://sarangambekar1997.github.io/data-engineering-handbook/06-infrastructure/kubernetes-for-de/).

Join Skew — When one key in a join has disproportionately many rows, causing one executor to do most of the work. Common cause of slow Spark joins.

Junk Dimension — A dimension table that consolidates low-cardinality flags and codes from the fact table (e.g., is_first_order, is_gift, has_promotion).

K

Kafka Lag — The difference between the latest offset on a Kafka partition and the consumer's current offset. High lag = consumer is falling behind.

Kafka Offset — A sequential integer that identifies a message's position in a Kafka partition. Consumers track their own offsets to know where to resume.

Kafka Streams — A Java library for stateful stream processing that reads from and writes to Kafka, running inside the application rather than on a separate cluster.

Kappa Architecture — A streaming-only architecture in which all processing runs on an event log, and history is reprocessed by replaying the log. Contrast with Lambda architecture.

KTable — In Kafka Streams, a table view of a stream holding the latest value per key (a changelog). Contrast with KStream, an unbounded stream of independent events.

L

Lakehouse — See Data Lakehouse.

Lambda Architecture — An architecture that runs a batch layer (complete, accurate) and a speed layer (low-latency, approximate) in parallel and merges their results at query time.

Lazy Evaluation — In Spark, transformations are not executed immediately — they build an execution plan (DAG) that runs only when an action is called. Enables optimization.

Liquid Clustering — A Delta Lake layout feature (`CLUSTER BY`) that replaces partitioning and Z-ordering, and lets clustering keys change without rewriting the whole table. See [Delta Lake](https://sarangambekar1997.github.io/data-engineering-handbook/01-storage/delta-lake/) (https://sarangambekar1997.github.io/data-engineering-handbook/01-storage/delta-lake/).

LLM (Large Language Model) — A neural network trained on large amounts of text, capable of understanding and generating human language. Examples: Claude models, GPT series, Gemini.

LSN (Log Sequence Number) — A position in a database's transaction log (for example the Postgres WAL). CDC tools use it to order changes and resume from an exact point.

M

Medallion Architecture — A three-layer data architecture: Bronze (raw) → Silver (cleaned) → Gold (business-ready). Each layer adds quality and structure.

MERGE (Upsert) — A single SQL statement that inserts, updates and deletes rows in a target table based on a match with a source. The core operation for applying CDC and making loads idempotent. See [Delta Lake](https://sarangambekar1997.github.io/data-engineering-handbook/01-storage/delta-lake/) (https://sarangambekar1997.github.io/data-engineering-handbook/01-storage/delta-lake/).

Merge-on-Read (MoR) — A table-format update strategy that appends changes to log files and merges them with base files at read time (or during compaction). Writes are cheap and fresh, and reads cost more until compaction. See [Apache Hudi](https://sarangambekar1997.github.io/data-engineering-handbook/01-storage/apache-hudi/) (https://sarangambekar1997.github.io/data-engineering-handbook/01-storage/apache-hudi/).

Metastore — A catalog that stores metadata about tables — schema, location, partitioning. Examples: Hive Metastore, AWS Glue Catalog, Databricks Unity Catalog.

Metric (semantic layer) — A named business definition, such as revenue or conversion rate, built from measures with filters and rules and defined once in code. See [Semantic Layer & Metrics](https://sarangambekar1997.github.io/data-engineering-handbook/02-processing/semantic-layer-metrics/) (https://sarangambekar1997.github.io/data-engineering-handbook/02-processing/semantic-layer-metrics/).

Micro-partition — Snowflake's internal storage unit. Each micro-partition holds 50–500MB of compressed data. Snowflake prunes irrelevant micro-partitions at query time.

Micro-batch — Spark Structured Streaming's default processing mode: collect data into small time-window batches and process each one. Contrast with continuous processing.

Model Context Protocol (MCP) — An open standard for connecting AI applications to external systems. A host application runs one client per server, and a server exposes tools, resources and prompts over JSON-RPC, on stdio for local servers or Streamable HTTP for remote ones. See [MCP and Text-to-SQL](https://sarangambekar1997.github.io/data-engineering-handbook/07-ai/mcp-text-to-sql/) (https://sarangambekar1997.github.io/data-engineering-handbook/07-ai/mcp-text-to-sql/).

N

Namespace (vector DB) — A logical partition within a vector index. Useful for multi-tenancy — one index, separate namespaces per customer or team.

Natural Key — The identifier from the source system (e.g., `customer_id = 'CUST-001'`). Used to join back to source data. Contrast with surrogate key.

Normalization — Organizing a relational database to reduce redundancy and dependency. Levels: 1NF, 2NF, 3NF, BCNF.

O

OLAP (Online Analytical Processing) — Systems optimized for complex analytical queries over large datasets. Read-heavy, columnar storage. Examples: Snowflake, BigQuery, Spark.

OLTP (Online Transaction Processing) — Systems optimized for fast, concurrent read-write transactions. Row-based storage. Examples: PostgreSQL, MySQL, DynamoDB.

OneLake — The single logical data lake every Microsoft Fabric tenant receives. It is built on Azure Data Lake Storage Gen2 and stores tables in open formats (Delta Parquet or Iceberg). See [Azure and Fabric](https://sarangambekar1997.github.io/data-engineering-handbook/01-storage/azure-fabric/) (https://sarangambekar1997.github.io/data-engineering-handbook/01-storage/azure-fabric/).

OpenLineage — An open standard for lineage metadata: jobs emit run events describing their inputs and outputs, and a backend assembles the lineage graph.

Orchestration — Coordinating the execution order, scheduling, and dependencies of pipeline tasks. Examples: Airflow, Prefect, Dagster.

P

Parquet — A columnar binary file format. Stores data column by column, enabling efficient compression and predicate pushdown. The standard format for data lakes.

Partition (data) — Dividing a dataset into sub-groups based on a column value (e.g., by date). Reduces data scanned per query if queries filter on the partition column.

Partition (Kafka) — A log within a Kafka topic. Messages within a partition are ordered. Partitions enable parallelism — more partitions = more consumer parallelism.

Partition Key and Sort Key — In DynamoDB and similar stores, the partition key decides which partition stores an item, and the optional sort key orders items within it, so a range query inside one partition is efficient. Cassandra's equivalent is the partition key and the clustering columns. See [NoSQL and Operational Stores](https://sarangambekar1997.github.io/data-engineering-handbook/01-storage/nosql-operational-stores/) (https://sarangambekar1997.github.io/data-engineering-handbook/01-storage/nosql-operational-stores/).

PCollection (Beam) — A distributed, immutable dataset in an Apache Beam pipeline. It is bounded if it comes from a fixed source such as a file, and unbounded if it comes from a continuous source such as a stream. See [Beam and Dataflow](https://sarangambekar1997.github.io/data-engineering-handbook/04-streaming/beam-dataflow/) (https://sarangambekar1997.github.io/data-engineering-handbook/04-streaming/beam-dataflow/).

Pod (Kubernetes) — The smallest unit Kubernetes schedules: one or more containers that run together on a node. A Spark executor and a batch job run are each a pod. See [Kubernetes](https://sarangambekar1997.github.io/data-engineering-handbook/06-infrastructure/kubernetes-fundamentals/) (https://sarangambekar1997.github.io/data-engineering-handbook/06-infrastructure/kubernetes-fundamentals/).

Polars — A multi-threaded DataFrame library with a lazy query optimizer and a streaming engine for larger-than-memory data.

Predicate Pushdown — Pushing filter conditions down to the storage layer so only matching data is read. Supported by Parquet, Delta Lake, and columnar databases.

Producer (Kafka) — A client that writes messages to a Kafka topic.

Prompt Caching — An Anthropic API feature that caches repeated prompt prefixes (system prompts, documents) to reduce latency and cost.

Property-Based Testing — Testing a rule that must hold for all inputs, such as "one row per key", by generating many inputs, including edge cases such as empty input, instead of listing examples by hand. See [Testing and CI/CD](https://sarangambekar1997.github.io/data-engineering-handbook/06-infrastructure/testing-cicd/) (https://sarangambekar1997.github.io/data-engineering-handbook/06-infrastructure/testing-cicd/).

Pseudonymisation — Replacing direct identifiers with tokens or keyed hashes so records can still be joined but not attributed to a person without extra information. The result is still personal data under [GDPR](https://sarangambekar1997.github.io/data-engineering-handbook/05-quality-governance/data-security-privacy/). See [Data Security & Privacy](https://sarangambekar1997.github.io/data-engineering-handbook/05-quality-governance/data-security-privacy/) (https://sarangambekar1997.github.io/data-engineering-handbook/05-quality-governance/data-security-privacy/).

R

RAG (Retrieval-Augmented Generation) — An LLM architecture that retrieves relevant documents from a knowledge base and includes them in the prompt before generating an answer.

Re-ranking — A post-retrieval step that uses a more expensive cross-encoder model to re-score and reorder retrieved chunks. Improves RAG precision.

Referential Integrity — A database constraint ensuring that foreign key values always point to an existing primary key.

Repartition — In Spark, redistributing data across partitions. Expensive (full shuffle), but fixes skew or right-sizes partitions before writing.

Replication Slot — A Postgres object that tracks how far a logical replication consumer (such as a [CDC](https://sarangambekar1997.github.io/data-engineering-handbook/06-infrastructure/kubernetes-for-de/) connector) has read. An unused slot makes the database retain [WAL](https://sarangambekar1997.github.io/data-engineering-handbook/06-infrastructure/kubernetes-for-de/) indefinitely.

Requests and Limits (Kubernetes) — Per-container resource settings. A request is what the scheduler reserves when it places a pod. A limit is the cap enforced at run time: CPU is throttled, and a container that exceeds its memory limit is killed (OOMKilled, exit code 137). See [Kubernetes](https://sarangambekar1997.github.io/data-engineering-handbook/06-infrastructure/kubernetes-for-de/) (https://sarangambekar1997.github.io/data-engineering-handbook/06-infrastructure/kubernetes-for-de/).

Reverse ETL — Syncing modeled data from the warehouse back into operational tools such as CRM, marketing, or support systems.

Role-Playing Dimension — When the same dimension table is used multiple times in a fact table with different semantic roles (e.g., dim_date used as order_date and ship_date).

Rollup (Druid) — Ingestion-time summarisation that combines rows with identical dimension values and the same timestamp, after truncation to the query granularity, into a single row. It shrinks storage at the cost of being able to query individual events. See [Real-Time Analytics Databases](https://sarangambekar1997.github.io/data-engineering-handbook/01-storage/realtime-olap/) (https://sarangambekar1997.github.io/data-engineering-handbook/01-storage/realtime-olap/).

Row-Level Security (RLS) — Restricting which rows a user or role can see, by attaching a filter to a table or dataset. It can be enforced in the warehouse, in a BI tool, or through signed tokens for embedded analytics. Rules enforced only in a BI tool can be bypassed by direct database access. See [BI Tools](https://sarangambekar1997.github.io/data-engineering-handbook/02-processing/bi-tools/) (https://sarangambekar1997.github.io/data-engineering-handbook/02-processing/bi-tools/).

RPU (Redshift Processing Unit) — The unit of compute capacity in Amazon Redshift Serverless, billed per second while queries run.

S

Savepoint (Flink) — A manually triggered, portable snapshot of a Flink job's state, used to stop and resume jobs across upgrades, code changes, and rescaling. See [Apache Flink](https://sarangambekar1997.github.io/data-engineering-handbook/04-streaming/flink-reference/) (https://sarangambekar1997.github.io/data-engineering-handbook/04-streaming/flink-reference/).

SCD (Slowly Changing Dimension) — A dimension table where attribute values change over time. Types: 0 (ignore), 1 (overwrite), 2 (add new row), 3 (add column).

Schema Registry — A service that stores and validates Avro/Protobuf/JSON schemas for Kafka topics. Ensures producers and consumers agree on message format.

Schema-on-Read — Schema is applied when data is read, not when it's written. Enables flexible raw storage (data lake).

Schema-on-Write — Schema is enforced when data is written. Ensures consistency but requires upfront schema design (data warehouse).

Segment (Druid and Pinot) — The unit of storage in Druid and Pinot: a columnar file of data and its indexes, which is the unit that is replicated and queried in parallel. In Druid, segments are partitioned by time. See [Real-Time Analytics Databases](https://sarangambekar1997.github.io/data-engineering-handbook/01-storage/realtime-olap/) (https://sarangambekar1997.github.io/data-engineering-handbook/01-storage/realtime-olap/).

Semantic Layer — A layer between warehouse tables and consumers that defines metrics and dimensions once, and generates the correct SQL for BI tools, notebooks, APIs and AI assistants. See [Semantic Layer & Metrics](https://sarangambekar1997.github.io/data-engineering-handbook/02-processing/semantic-layer-metrics/) (https://sarangambekar1997.github.io/data-engineering-handbook/02-processing/semantic-layer-metrics/).

Semantic Search — Search by meaning rather than exact keyword matching. Powered by embeddings — finds documents conceptually similar to the query.

Shortcut (OneLake) — A reference in OneLake to data stored elsewhere, such as another workspace, ADLS Gen2 or Amazon S3, so it can be queried without copying. Changes at the source are visible immediately. See [Azure and Fabric](https://sarangambekar1997.github.io/data-engineering-handbook/01-storage/azure-fabric/) (https://sarangambekar1997.github.io/data-engineering-handbook/01-storage/azure-fabric/).

Showback / Chargeback — Reporting cloud costs to the teams that incur them (showback), or billing those costs to their budgets (chargeback).

Silver Layer — The second layer in medallion architecture. Data is cleaned, typed, deduplicated, and lightly joined. Conformed to business rules.

Skew — Uneven distribution of data across partitions or tasks. One partition has far more data than others, causing bottlenecks.

SLA (Service Level Agreement) — A commitment about data availability, freshness, or quality. E.g., "data available within 2 hours of source update."

SLI (Service Level Indicator) — A measurement of service quality, such as minutes between the source's last event and the table's newest row. See [Pipeline Observability](https://sarangambekar1997.github.io/data-engineering-handbook/05-quality-governance/pipeline-observability/) (https://sarangambekar1997.github.io/data-engineering-handbook/05-quality-governance/pipeline-observability/).

SLO (Service Level Objective) — An internal target for an SLI, such as "`fct_orders` is ready by 07:00 UTC on 99% of days". Stricter than the SLA it supports. See [Pipeline Observability](https://sarangambekar1997.github.io/data-engineering-handbook/05-quality-governance/pipeline-observability/) (https://sarangambekar1997.github.io/data-engineering-handbook/05-quality-governance/pipeline-observability/).

Slot (BigQuery) — A unit of compute capacity that BigQuery uses to execute queries; billed on demand by bytes processed or through reserved, autoscaling capacity. See [BigQuery](https://sarangambekar1997.github.io/data-engineering-handbook/01-storage/bigquery-reference/) (https://sarangambekar1997.github.io/data-engineering-handbook/01-storage/bigquery-reference/).

Snowflake Schema — A normalized star schema where dimension tables reference other dimension tables. More normalized but more joins than a star schema.

Sort Key (Redshift) — The column order in which Redshift stores rows on disk. Zone maps (min/max per block) let filters on the sort key skip blocks.

Star Schema — A dimensional modeling pattern with one central fact table surrounded by dimension tables. Optimized for analytical queries.

Star-Tree Index (Pinot) — A Pinot index that pre-aggregates across chosen dimensions, so matching aggregation queries have a bounded latency, in exchange for extra storage. See [Real-Time Analytics Databases](https://sarangambekar1997.github.io/data-engineering-handbook/01-storage/realtime-olap/) (https://sarangambekar1997.github.io/data-engineering-handbook/01-storage/realtime-olap/).

Streaming — Processing data continuously as it arrives, rather than in batches. Examples: Kafka, Spark Structured Streaming, Flink.

Streaming Database — A database that keeps the results of SQL views continuously up to date as data arrives from sources such as Kafka and [CDC](#), using incremental view maintenance, and serves them over a standard SQL interface. Examples: RisingWave and Materialize. See [Streaming SQL](#) (<https://sarangambekar1997.github.io/data-engineering-handbook/04-streaming/streaming-sql/>).

Surrogate Key — A warehouse-generated integer key used to join fact and dimension tables. Stable, independent of the source system's natural key.

T

Taint and Toleration (Kubernetes) — A taint on a node repels pods, and a toleration on a pod allows it to be scheduled there. They reserve node pools, for example spot capacity for batch work. See [Kubernetes](#) (<https://sarangambekar1997.github.io/data-engineering-handbook/06-infrastructure/kubernetes-for-de/>).

Temporal Filter (Materialize) — A `WHERE` condition on `mz_now()`, Materialize's current virtual timestamp, such as `mz_now() <= event_ts + INTERVAL '1 hour'`. As time advances, rows that no longer satisfy it are retracted from the result, giving a sliding window that expires old data and bounds state. See [Streaming SQL](#) (<https://sarangambekar1997.github.io/data-engineering-handbook/04-streaming/streaming-sql/>).

Text-to-SQL — Using a language model to turn a natural-language question into a SQL query. The hard parts are schema linking, business definitions and ambiguity, and safety comes from validating and restricting the SQL, not from the prompt. See [MCP and Text-to-SQL](#) (<https://sarangambekar1997.github.io/data-engineering-handbook/07-ai/mcp-text-to-sql/>).

Time Travel — The ability to query historical versions of a table. Supported natively by Delta Lake, Snowflake (up to 90 days), and Apache Iceberg.

Tombstone (Kafka) — A message with a key and a null value. On a compacted topic it tells Kafka to remove earlier messages with that key. Debezium sends one after each delete event so the deleted row's history can be compacted away; a consumer that applies changes must not treat it as a change. See [Ingestion & CDC](#) (<https://sarangambekar1997.github.io/data-engineering-handbook/02-processing/ingestion-cdc/>).

Tool Use — An LLM feature where the model can call functions defined by the developer — search, run SQL, call APIs — and use their results to answer questions.

Transaction (database) — A group of SQL operations that succeed or fail together (ACID). Ensures data consistency.

Trigger (Spark) — The scheduling rule for when a streaming micro-batch runs: once, continuously, or on a fixed interval.

U

Unit Economics (Data) — Cost expressed per unit of value — per pipeline run, per table, per query, or per customer — to track efficiency as volume grows.

Upsert — Insert if the record doesn't exist, update if it does. Implemented with `MERGE` in SQL, `mode("overwrite")` in Spark, or `upsert` in vector DBs.

V

Vacuum — In Delta Lake, removes old data files no longer needed by the current version. Reclaims storage. Default retention: 7 days.

Vector Database — A database optimized for storing and querying embedding vectors using approximate nearest neighbor (ANN) search.

Virtual Warehouse (Snowflake) — An independent compute cluster in Snowflake. Billed by the hour only when active. Multiple warehouses can query the same data simultaneously.

W

Watermark (Spark Streaming) — A threshold that tells Spark how long to wait for late-arriving data before closing a time window.

Windowed Aggregation — Aggregating streaming data over a sliding or tumbling time window (e.g., count of events per 5-minute window).

Window Function (SQL) — A SQL function that performs a calculation across a set of rows related to the current row without collapsing them (unlike `GROUP BY`). Examples: `ROW_NUMBER`, `RANK`, `LAG`, `LEAD`, `SUM OVER`.

Work Pool (Prefect) — A queue of flow runs bound to an infrastructure type (process, Docker, Kubernetes, serverless). Workers in your own environment poll it and start the runs. See [Prefect](https://sarangambekar1997.github.io/data-engineering-handbook/03-orchestration/prefect-reference/) (https://sarangambekar1997.github.io/data-engineering-handbook/03-orchestration/prefect-reference/).

Workload Identity Federation — Exchanging a workload's native identity token (from a cloud, CI system, or Kubernetes) for short-lived credentials in another system, avoiding long-lived access keys.

Write-Ahead Log (WAL) — A database's log of every change, written before the change is applied to the data files, so the database can recover after a crash. Log-based CDC reads it (Postgres needs `wal_level = logical`), and an unread replication slot makes Postgres

keep it. See [Ingestion & CDC](https://sarangambekar1997.github.io/data-engineering-handbook/02-processing/ingestion-cdc/) (https://sarangambekar1997.github.io/data-engineering-handbook/02-processing/ingestion-cdc/).

Write-Audit-Publish (WAP) — A publishing pattern: write the output to a staging table, audit it, and swap it in only if every audit passes, so consumers never see invalid or partial data. See [Testing and CI/CD](https://sarangambekar1997.github.io/data-engineering-handbook/06-infrastructure/testing-cicd/) (https://sarangambekar1997.github.io/data-engineering-handbook/06-infrastructure/testing-cicd/).

X

XCom (Airflow) — Cross-communication between Airflow tasks. Allows a downstream task to access values returned by an upstream task.

Z

Z-Order — A data skipping optimization in Delta Lake that colocalizes related data in the same files based on column values, improving query performance for multi-column filters.

Zero-Copy Clone (Snowflake) — Creates a copy of a Snowflake table, schema, or database that shares the underlying storage until modified. Fast and nearly free until changes are made.

Back to: [Index](https://sarangambekar1997.github.io/data-engineering-handbook/) (https://sarangambekar1997.github.io/data-engineering-handbook/)

 September 29, 2026