

Lesson 07 — Databases

SQL / NoSQL / vector / search — what to store where, and the cost of getting it wrong.

IT Landscape

Module 2

Why a database, instead of files?

A folder of `.csv` files is a database — for one user. It collapses past that.

- **Many users at once** — concurrent writes corrupt a flat file.
 - **Durable across crashes** — mid-write power loss = broken row.
 - **Queryable** — “give me all orders from yesterday over \$100” — not a grep job.
 - **Enforces structure** — the “email” column is actually an email.
 - **Joins** — combine two datasets by a shared key without writing custom code.
-

SQL — the 50-year incumbent

Invented in the 1970s. Still the most-deployed kind of database in the world.

Brands you'll hear (as of 2026):

- **PostgreSQL** — free, broadly loved, default for new projects.
 - **MySQL / MariaDB** — free, classic LAMP stack.
 - **Microsoft SQL Server** — the enterprise Windows world.
 - **Oracle** — paid, enterprise, deeply entrenched.
 - **SQLite** — a database in a single file. Runs in your phone, your browser, your IDE.
-

Tables, rows, queries

A tiny SQL query:

```
SELECT name, email  
FROM users  
WHERE country = 'UA'  
ORDER BY name;
```

*You don't need to write this — just see the shape:
column names, a filter, an ordering.*

The `users` table:

name	email	country
Olena	olena@example.com	UA
Petro	petro@example.com	UA
Jane	jane@example.com	US

NoSQL — when SQL doesn't fit

In the 2000s, web companies hit limits with single-machine SQL:

- Too much write traffic.
- Too many JSON-shaped objects.
- Too rigid a schema for fast-evolving products.

NoSQL is a catch-all for the alternatives that emerged. There are **four main shapes** — the next four slides.

Document stores — MongoDB & friends

Store **JSON-like documents** instead of rows. Schema can vary per document.

- **Good for:** content (blog posts), product catalogs, user profiles with varying fields.

Names you'll hear (as of 2026):

- **MongoDB** — the icon.
 - **DynamoDB** (AWS), **Firestore** (Google), **Cosmos DB** (Azure).
 - **Couchbase**.
-

Key-value — Redis & friends

Just a giant **hash map**. Lives in memory. Ridiculously fast.

```
SET user:42:session "abc123"  
GET user:42:session    -> "abc123"
```

Used as:

- Cache, session store, leaderboard, rate limiter, pub/sub bus.

Names (as of 2026): **Redis** (king), **Valkey** (fork), **Memcached** (older), **DynamoDB** (also key-value).

Search — Elasticsearch & friends

When you need **full-text search** — typos, synonyms, ranking by relevance — SQL `LIKE` `'%foo%'` is the wrong tool.

Search engines build **inverted indexes** over text and rank results.

Names (as of 2026):

- **Elasticsearch** (and **OpenSearch**, the AWS fork) — the heavyweight.
 - **Algolia** — SaaS, dev-friendly.
 - **MeiliSearch**, **Typesense** — lightweight upstarts.
-

Vector databases — for AI (as of 2026)

LLMs and embeddings need to search by **semantic similarity**, not keyword.

Vector DBs store high-dimensional vectors (~1024 numbers per item) and find **nearest neighbors** fast.

Names (as of 2026):

- **Pinecone** — SaaS pioneer.
- **Weaviate, Qdrant, Chroma.**
- **pgvector** — Postgres extension; the “just use Postgres” choice.

This space is the most volatile — vendors come and go quarterly. Refresh annually.

Picking the right one

If your data looks like ...	Reach for ...
Transactional / accounts / money	SQL (Postgres)
User-shaped data with varying fields	Document store (Mongo)
Session, cache, leaderboard	Key-value (Redis)
Search bar with typo tolerance	Search engine (Elasticsearch / Algolia)
RAG — “find similar to this prompt”	Vector DB (pgvector / Pinecone)

The cost of getting it wrong

- **Data loss** — a NoSQL store dropped writes during a primary failover (sometimes silently).
- **Wrong shape forever** — denormalized so hard that adding a new query takes weeks.
- **Egress cost** — cross-region replication = five-figure monthly cloud bill.
- **Schema rigidity** — changing a column type in a 1 TB SQL table = hours of downtime.
- **Search relevance** — “nobody can find anything in our app” — a 6-month fix.

Database choice is a long-tail commitment. You will live with this one for years.

"Just use Postgres"

A common 2026 take. Also serious advice.

Modern PostgreSQL handles:

- **JSON** — like a document store (`JSONB`).
- **Key-value** — `UNLOGGED` tables or `hstore` .
- **Full-text search** — built-in `tsvector` .
- **Vectors** — `pgvector` extension.
- **Time-series** — TimescaleDB extension.

One database, fewer moving parts. *Caveat: at extreme scale, specialized stores still win.*

Recap

- Databases are **not just “SQL”** — five families, each with a sweet spot.
- The right database is the one that **matches your query patterns**.
- For most teams, **“just use Postgres”** is the right starting point.

Coming up: Plan C / Module 3 — getting software actually running for real users.

