Relational Databases Considered Incredibly Useful

Simon Arneaud

https://theartofmachinery.com/

*** Spoiler Alert ***

Relational DBs have a learning curve

…but they're really, really useful

Often an RDB isn't the best tool for a task

…but it's often the best single tool for an application

Joins and ACID don't scale to large clusters

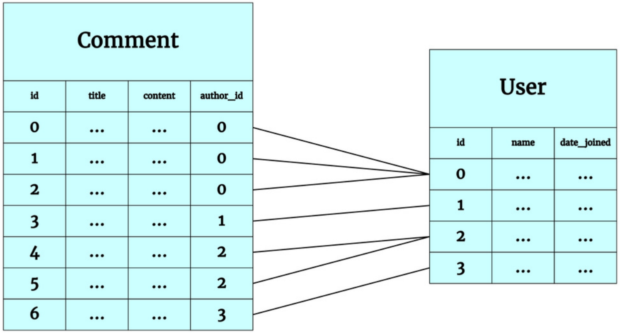
…but for most commercially viable applications,
that doesn't matter

In 2017, you can get

- multiple super fast CPU cores with
- over 1TB of RAM
- and several TB of storage
- ...on SSD

on a single machine

and there are still ~90k seconds in a day



Comment

id	title	content	author_id
0	...	...	0
1	...	...	0
2	...	...	0
3	...	...	1
4	...	...	2
5	...	...	2
6	...	...	3

User

id	name	date_joined
0	...	...
1	...	...
2	...	...
3	...	...

```
CREATE TABLE User (
  id INTEGER PRIMARY KEY,
  name TEXT NOT NULL,
  date_joined DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP
);

INSERT INTO User (name) VALUES ("simon"), ("chewxy");

SELECT name, date_joined FROM User WHERE id = 1;
SELECT 3 + 4;

SELECT Comment.title FROM Comment JOIN User
  ON Comment.author_id = User.id
  WHERE User.name = "simon";

INSERT INTO Comment (title, content, author_id) VALUES
  ("Important SyPy Rule",
   "You must tweet #sypy",
   (SELECT id FROM User WHERE User.name = "chewxy"));
```

Pythonistas can use SQLAlchemy to take away most of the clunkiness

https://www.sqlalchemy.org/

Major Players

FOSS

- MySQL (MariaDB)
- Postgres
- Sqlite
- Firebird

Proprietary

- MS SQL Server
- IBM DB2
- Oracle

Unfortunately, they're incompatible enough that you generally have to choose just one for a project

MySQL

- Historically extremely popular
- Has a reputation for performance and ease of use
- More commonly chosen for very large deployments (e.g., AWS DynamoDB is built on MySQL)
- But…

MySQL

- Performance today is practically on par with other DBs
- "Easy to use" really means it won't tell you when you do something wrong
- The query planner is pretty stupid and easily confused
- Multimaster deployments aside, plays feature catchup, even with Sqlite
- https://grimoire.ca/mysql/choose-something-else

Postgres

- Not pretty, but sane and reliable
- Impressive feature set
- Good range of authentication options
- Installation is a learning speed bump

Sqlite

- Instead of a server, DB = file + library
- Easy to install, deploy and even embed
- Great for learning SQL and relational DB theory
- Not strict about types (like MySQL)
- Has some documented limitations with query planning
- But, much, much more powerful than the name suggests

Sqlite

Rollback journal mode (a.k.a, "big fat lock mode")

- Default and easy to use
- Many concurrent readers, one writer
- Best performance on read-heavy loads

Write-ahead log (WAL) mode

- Many concurrent readers, and writers
- Slightly more complex requirements
- Slightly weaker and more difficult isolation guarantees
- Best performance on write-heavy loads

Applications

Using an RDB the way Codd intended

I.e. using a normalised schema and joins and stuff.

Contrary to popular myth,
denormalisation can easily hurt performance.

Key-value store

Just make a table with a "key" and "value" column.

Useful for migrating apps that have outgrown a KV DB.

Document store

Postgres, MySQL and Sqlite (with extensions) support JSON data types and queries.

Text search engine

https://www.postgresql.org/docs/current/static/textsearch.html

Geographic information system (GIS)

Spatial database extensions are available for all major DBs.

Message queue

Since 2016, can be implemented more efficiently using Postgres's SKIP LOCKED.

Pub/sub

Postgres has LISTEN/NOTIFY. It's not very scalable, but maybe you don't really need a broker.

Filesystem

The classic advice is to not store files in a DB,
but sometimes it's the better option.

https://www.sqlite.org/fasterthanfs.html

https://www.microsoft.com/en-us/research/publication/to-blob-or-not-to-blob-large-object-storage-in-a-database-or-a-filesystem/

Questions?

Simon Arneaud

https://theartofmachinery.com/

enquiries@taom.systems