

Choosing the right database for your next project

***Looking at options beyond
PostgreSQL and MySQL***

PyCon MEA / GlobalDevSlam 2022 – 11.10.2022

Dubai, UAE

Marc-André Lemburg :: eGenix.com GmbH

Speaker Introduction

- Marc-André Lemburg
 - Python since 1994
 - Studied Mathematics
 - CEO eGenix.com GmbH
 - Consulting CTO, Senior Solutions Architect and Coach
 - EuroPython Society Fellow and former Chair
 - Python Software Foundation Fellow
 - Python Core Developer
 - Co-founder Python Meeting Düsseldorf
 - Based in Düsseldorf, Germany
 - More and for connecting: <http://malemburg.com>

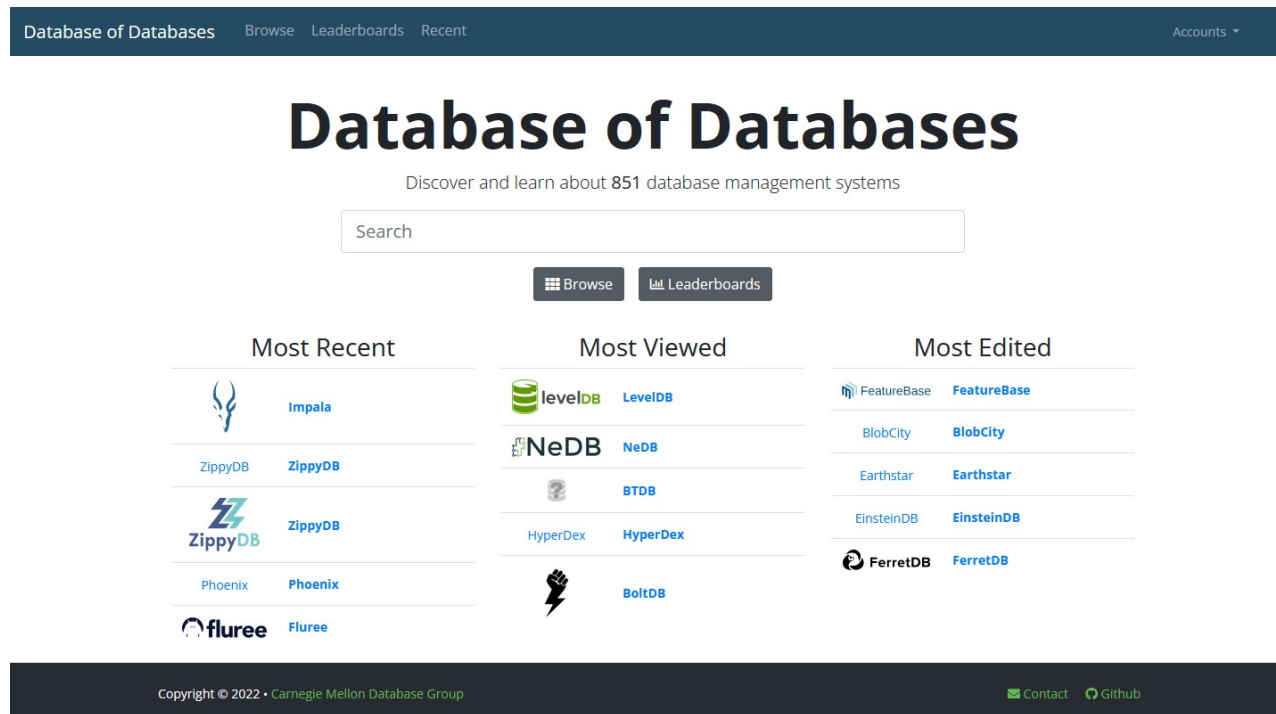


Motivation: You're thinking of building the next Big Thing ...


















Motivation: ... and you need a database for this.

There are **800+ databases** out there – <https://dbdb.io/>



The screenshot shows the homepage of the Database of Databases website. At the top, there is a navigation bar with links for "Database of Databases", "Browse", "Leaderboards", and "Recent", along with an "Accounts" dropdown menu. The main heading is "Database of Databases" with the subtitle "Discover and learn about 851 database management systems". Below this is a search bar and two buttons: "Browse" and "Leaderboards". The content is organized into three columns: "Most Recent", "Most Viewed", and "Most Edited".

Most Recent	Most Viewed	Most Edited
 Impala	 levelDB LevelDB	 FeatureBase FeatureBase
 ZippyDB ZippyDB	 NeDB NeDB	 BlobCity BlobCity
 ZippyDB ZippyDB	 BTDB BTDB	 Earthstar Earthstar
 Phoenix Phoenix	 HyperDex HyperDex	 EinsteinDB EinsteinDB
 fluree Fluree	 BoltDB BoltDB	 FerretDB FerretDB

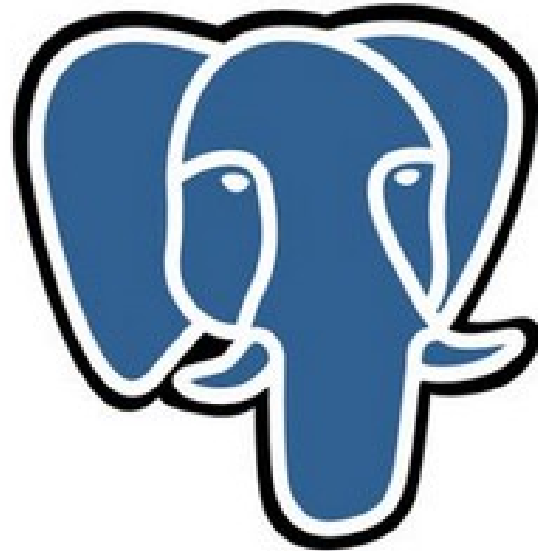
At the bottom of the page, there is a footer with "Copyright © 2022 • Carnegie Mellon Database Group" on the left and "Contact" and "Github" links on the right.

Developing databases has become the new hot thing

- Plenty of new databases were created in recent years
- Often by large Internet companies, e.g. Google, Facebook, LinkedIn, Uber, etc. or by smaller startups looking to fill a niche in the market
- Goals
 - Better performance
 - Faster intake
 - Faster analytics
 - More scalable, more reliable
 - Clustered, containerized, running on Kubernetes
 - Using GPUs, in-memory setups, separating compute / storage



Now, you could just pick one of these and be done ...



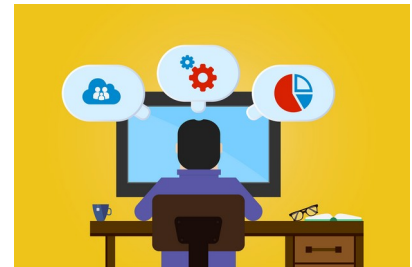
PostgreSQL

But you might lose a competitive edge ...



How to select a database engine ?

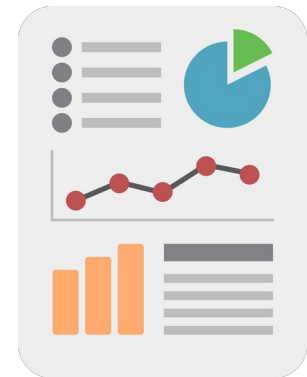
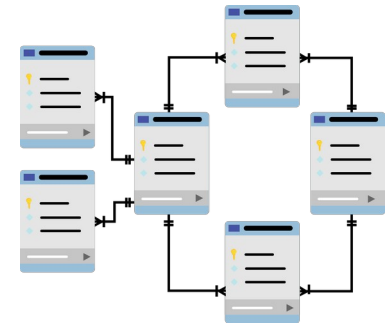
1. Determine your typical **data workload**
 2. Look at the **intended use cases** to **prioritize features**
 3. Consider **additional decision factors / constraints**
 4. **Get an overview** of **what's available on the market**
 5. **Pick the best fit !**
- *For larger projects:*
Create a POC to help determine the right choice



1. Determine the typical data workload

- **OLTP** – Online transaction processing
 - Main purpose: **operations**
 - Optimized for fast CRUD, simple joins
 - Data changes frequently
 - *Examples: General purpose databases*

- **OLAP** – Online analytical processing
 - Main purpose: **analytics and reporting**
 - Optimized for read, complex queries, large joins
 - Data is rather stable, sometimes even read- or append-only
 - *Examples: Data warehouse solutions*



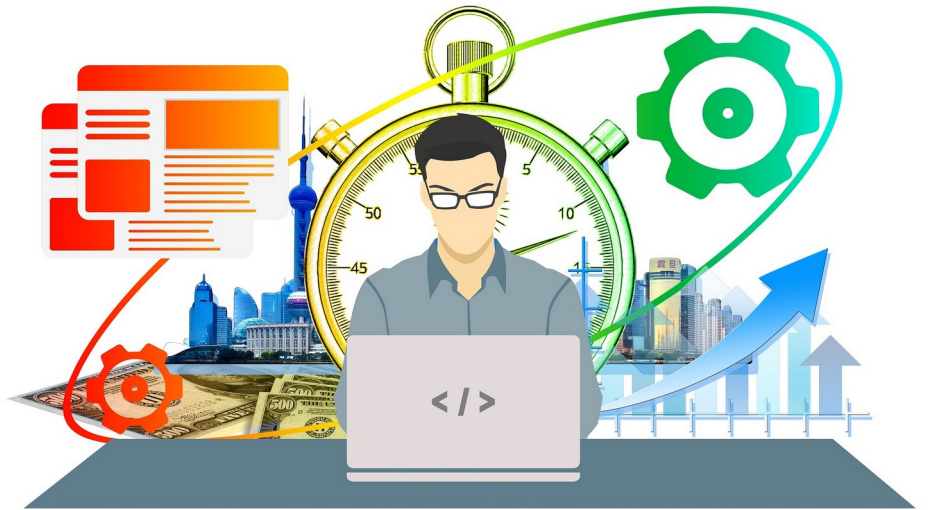
2. Use cases: Understanding your problem space

- **The Knowns:**
Your database use case will often require a particular **feature set**
- **The Unknowns:**
Plan for **future developments**
 - Scaling up
 - Change frequency of your designs / schemas
 - Additional requirements which you may not yet know



2. Use cases: Let's look at a few complexity dimensions

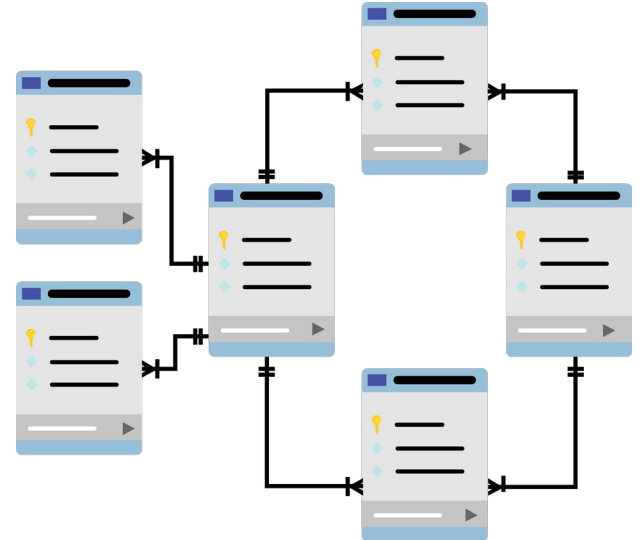
- **Schema** complexity
- **Cardinal** complexity
- **Temporal** complexity
- **Query** complexity
- **Deployment** complexity
- **Operational** complexity



(you may have more dimensions to add)

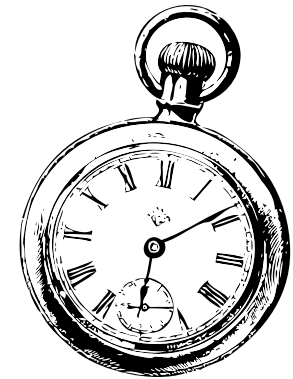
2. Use cases: Schema complexity

- **Simple schemas**
 - Session and user management
 - Lookup tables
 - *Examples: Web servers, logging, event storage*
- **Complex schemas**
 - Interconnected schemas from different systems
 - Wide tables to implement nested structures
 - *Examples: Accounting, trading or booking systems*
- **Migrations**



2. Use cases: Cardinal & Temporal complexity

- **Cardinal** complexity
 - Many simple rows
 - Fewer rows with many columns
 - Many rows with lots of columns
 - Scalability factors: **Data growth rate per month**
- **Temporal** complexity
 - Read-only or append-only (e.g. timeseries)
 - Data changes often
 - Requirement to run point-in-time queries



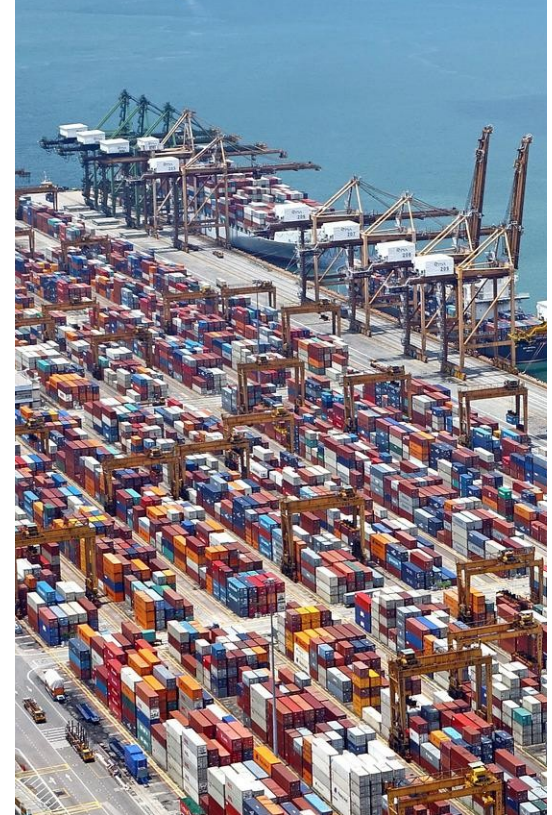
2. Use cases: Query complexity

- **Query** complexity
 - **Performance** requirements:
Transactions per second (TPS) for OLTP or analytics timing for OLAP
 - **Joins**:
Simple joins leading to small result sets or
Complex joins across multiple large tables
 - **Normalization**:
Fully normalized tables (DRY in databases) or
Denormalized tables (to avoid online joins)
 - Need for **Materialized Views, (special) Indexes**



2. Use cases: Deployment complexity

- **Compute**
 - Single server / cluster
 - On-premises hardware / Cloud
 - Dedicated hardware / VMs / containers / Kubernetes
- **Storage**
 - Data close to nodes (memory, NVMe, NFS)
 - Object storage (e.g. S3)
- **Resilience & Disaster Recovery (DR)**
 - Fail-over / High Availability Setups
 - Monitoring
 - Automation and self-healing



2. Use cases: Operational complexity

- **Managed / self-hosted**
 - Complex configuration
 - Self-optimizing
 - **Need for (elastic) scaling**
- Devops Integration
 - VMs and servers: Ansible / Salt / Terraform / etc.
 - Kubernetes: Operators / Helm Charts
 - Manual intervention: Shell access, UI tools
- **Upgrades**
 - Zero downtime
 - Performance degradation



3. Additional decision factors

- Is the database **mature enough** for my needs ?
- Is the ecosystem strong enough to **provide years of support** ?
- What are the **cost implications** ?
- Does the database have **good / supported Python interfaces** ?
- Does the database provide **interfaces to tools used in your stack** ?
 - e.g. Django, SQLAlchemy, DBT, Pandas, Dask, fugue, etc.

4. Now, let's do some coffee database tasting...



Selection of databases for various use cases

- General purpose OLTP client-server databases
- In-memory / embedded databases
- Data Warehouse OLAP databases
- Data Lake OLAP databases
- Distributed OLAP databases
- Document databases
- Timeseries databases
- Machine learning and databases
- Other types of databases

General purpose OLTP client-server databases

- PostgreSQL
- MySQL / MariaDB
- Oracle, MS SQL Server, IBM DB2
- Cloud versions of most of the above
 - Amazon AWS RDS, Aurora
 - Microsoft Azure Database
 - Google Cloud SQL

Most of these are row-oriented.



PostgreSQL



MariaDB

ORACLE



Microsoft
SQL Server



In-memory / embedded databases

- **SQLite**
 - Embedded, on-disk and in-memory
 - Focus: OLTP
- **DuckDB**
 - Embedded, on-disk and in-memory
 - Focus: OLAP
- **Exasol**
 - Very fast, client-server, runs in memory only (persists to disk)
 - Focus: OLAP



● **DuckDB**

Exasol

Data Warehouse OLAP databases

- **Snowflake / Google BigQuery / Azure Synapse Analytics**
 - Fully managed cloud platforms
- **Amazon Redshift**
 - PostgreSQL compatible, fully managed cloud platform
- **Greenplum**
 - PostgreSQL fork which adds clustering and MPP
- **Teradata**
 - Used to be on-prem, now fully managed cloud platform

Most of these are column-oriented.



Data Lake OLAP databases

- **Amazon Athena**
 - Queries against data stored on S3 (similar to Apache Presto)
 - Fully managed cloud platform
- **Apache Presto / Trino**
 - Database engines only, no storage (separates compute/storage)
 - Allows querying across several data sources (S3, Cassandra, MongoDB, Kafka)
- **Delta Lake**
 - ACID compliant storage layer with several compute integrations, e.g. for Trino



Amazon
Athena



Most of these are column-oriented.

Distributed OLAP databases

- **YugabyteDB** and CockroachDB
 - PG compatible
- SingleStore
 - MySQL compatible
- ClickHouse
 - Fast OLAP engine, but lacks a few standard SQL features
 - Good for log analysis
- Exasol
 - Runs on RAM in a cluster, so very fast



Many of these can also be used for data warehouses / lakes.

Document databases

- **ElasticSearch / OpenSearch**
 - JSON document databases with focus on text queries
 - Part of the ELK Stack
- **Apache Cassandra**
 - High performance and highly scalable
 - SQL-like query language
- **MongoDB**
 - Full featured document database
- **Apache CouchDB**
 - Great for online / offline replicating data across different platforms



Timeseries databases

- **InfluxDB**
 - Fast intake and highly available, great for IoT and metrics
- **Apache Druid**
 - Fast intake and queries, real-time system for e.g. ad systems
- **CrateDB**
 - Combines Elasticsearch (document) and Presto/Trino (SQL)
- **TimescaleDB**
 - PostgreSQL extension



Timeseries databases (immutable)

- Apache Pino
 - Fast, append-only, immutable
- **ImmuDB**
 - Append-only, immutable, tamperproof
 - Great for auditing



Machine learning and databases

- Some database engines provide tooling to **run Python ML inside the database server**:
 - PostgreSQL and Greenplum:
via [MADlib](#)
 - Oracle, MS SQL Server, DB2:
via [Python UDFs](#)
- Other engines focus on **making ML available via SQL**:
 - [MindsDB](#): works with a large range of databases as a SQL proxy



Other types of databases

- Key value storages
- Databases running on GPUs
- Geo databases (for location based queries)
- Graph databases (for graph based queries)
- Vector databases (for ML models)
- Strongly typed databases

... and many more

(perhaps in some future talk)



5. Pick the best fit... there are lots of great options available



Amazon
Athena



PostgreSQL



ORACLE®



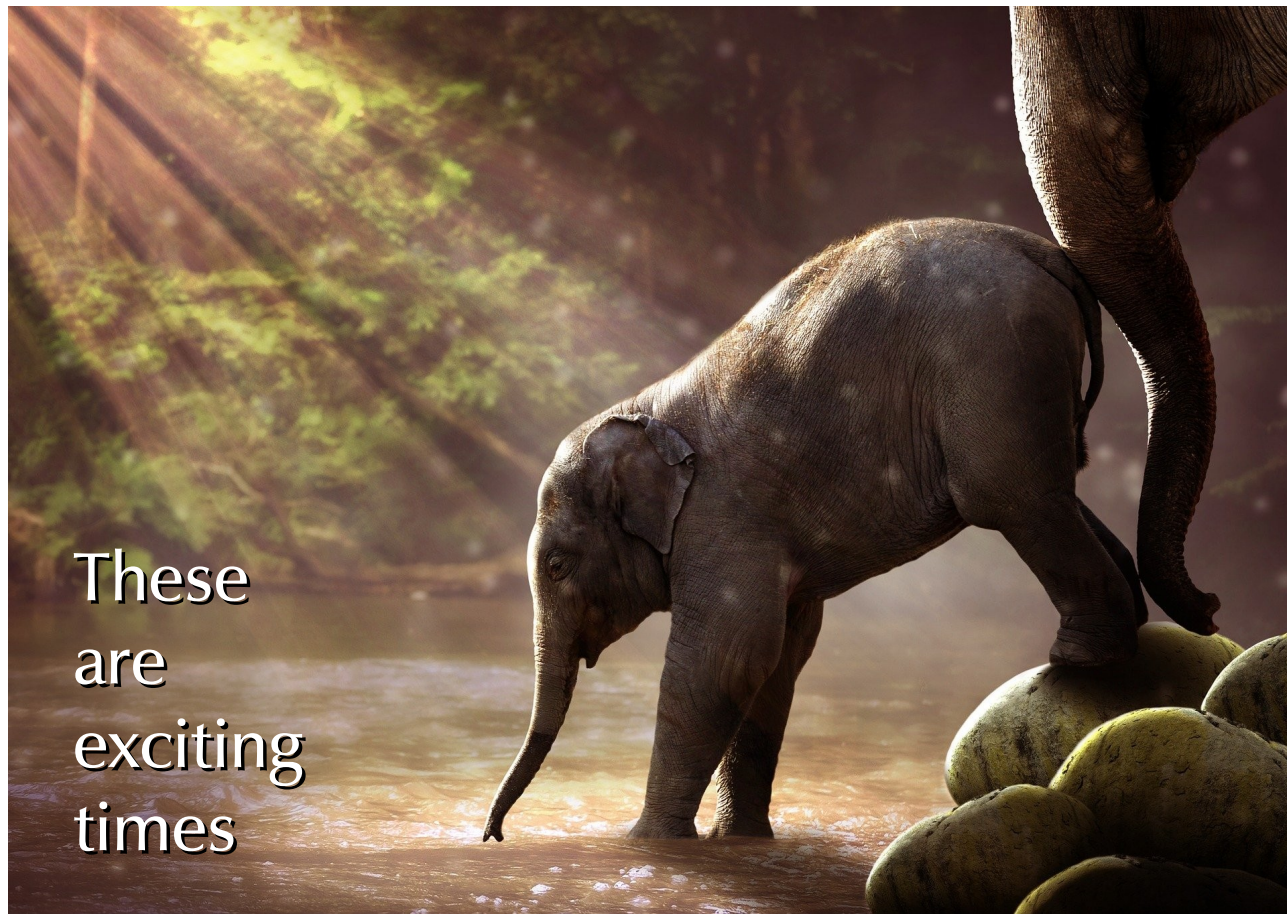
DELTA LAKE



DELTA LAKE



Main takeaway: Never stop to **learn** and **try out new things...**



These
are
exciting
times

Thank you for your attention !



Time for discussion

Contact

eGenix.com Software, Skills and Services GmbH

Marc-André Lemburg

Pastor-Löh-Str. 48

D-40764 Langenfeld

Germany

eMail: mal@egenix.com

Phone: +49 211 9304112

Fax: +49 211 3005250

Web: <https://www.egenix.com/>

LinkedIn:



References

- Several photos taken from Pixabay and Unsplash
- Some screenshots taken from the mentioned websites
- All other graphics and photos are (c) eGenix.com or used with permission
- Details are available on request
- Logos are trademarks of their respective trademark holders