Types of Databases: Relational, NoSQL, Cloud, Vector

The main types of databases include relational databases for structured data, NoSQL databases for flexibility, cloud databases for remote access, and vector databases for machine learning applications.

May 22, 2024 · 15 min read

Moez Ali
Data Scientist, Founder & Creator of PyCaret

TOPICS

Data Science

SQL

In today's data-driven world, we face a significant challenge: how do we efficiently store, manage, and extract meaningful insights from data? Databases offer a solution, providing structured repositories for organizing and accessing information.

However, to address the unique requirements of various data structures and use cases, different types of databases have emerged.

In this article, we'll explore the four main types you'll encounter in the data science world: **relational databases**, **NoSQL databases**, **cloud databases**, and **vector databases**.

If you want to learn about database design, check out this course on Database Design.

The Importance of Databases

Databases are essential tools in the digital world. They are organized collections of data that facilitate the storage, retrieval, management, and manipulation of information.

At their core, databases are designed to **hold data in a structured format**, allowing users and applications to efficiently access and update the information as needed.

The importance of databases extends across nearly all fields but is particularly critical in data science. Data science projects often involve analyzing large volumes of data to derive insights, make predictions, or inform decision-making.

Without databases, managing this data—especially as it grows in size and complexity—would be cumbersome and error-prone. Databases provide a **systematic way to store data** and ensure its integrity, security, and accessibility.

Consider, for example, a retail company that tracks sales, customer interactions, inventory, and supplier information. A database serves as the backbone of the company's operations, enabling them to analyze trends, forecast demand, optimize inventory levels, and enhance customer experiences.

Without a database, the company would struggle to handle the vast amounts of data generated daily, let alone use this data to make informed business decisions.

Types of Databases: A Quick Overview

The different types of databases reflect the varied needs of use-cases and the complexities of the data they handle. Different types of databases are developed to optimize performance, enhance functionality, and cater to specific use cases.

This variety is not just a matter of technological abundance but also a necessity to address the unique challenges and requirements that arise in different use-cases. The need for different types of databases stems from the differences in data structures, access patterns, scalability demands, and consistency requirements.

For instance, traditional business applications often rely on structured data that fits well into tables with predefined schemas, making **relational databases** an ideal choice.

However, with the rise of big data, social networks, and real-time analytics, the limitations of relational databases in handling unstructured data, scaling horizontally, or managing highly connected data became evident.

This led to the emergence of **NoSQL databases**, designed to offer flexibility, scalability, and performance advantages for certain types of data that do not conform to the rigid structure of traditional databases. If you want to learn more how SQL and NoSQL databases compare, check out this tutorial on **SQL vs NoSQL Databases**.

Similarly, the advent of IoT and time-sensitive applications necessitated the development of **time-series databases** optimized for efficiently handling temporal data.

Cloud databases have also gained prominence, offering scalability and accessibility by hosting data on remote servers.

Additionally, **vector databases** have emerged to cater to the specific needs of machine learning applications, efficiently storing and querying high-dimensional vectors.

Popular Databases Management Systems

DB-Engines Ranking for May 2024 lists the top database management systems (DBMS) based on their popularity. This ranking is updated monthly and includes 420 systems. As of May 2024, the top four databases are all relational: Oracle, MySQL, Microsoft SQL, and PostgreSQL.

DB-Engines Ranking

The DB-Engines Ranking ranks database management systems according to their popularity. The ranking is updated monthly.

Read more about the method of calculating the scores.

trend chart

420 systems in ranking, May 2024

Rank May 2024	Rank Apr 2024	Rank May 2023	DBMS	Database Model	Score May 2024	Score Apr 2024	Score May 2023
1.	1.	1.	Oracle	Relational, Multi-model	1236.29	+2.02	+3.66
2.	2.	2.	MySQL	Relational, Multi-model	1083.74	-3.99	-88.72
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	824.29	-5.50	-95.80
4.	4.	4.	PostgreSQL	Relational, Multi-model	645.54	+0.49	+27.64
5.	5.	5.	MongoDB	Document, Multi-model	421.65	-2.31	-14.96
6.	6.	6.	Redis	Key-value, Multi-model	157.80	+1.36	-10.33
7.	7.	8.	Elasticsearch	Search engine, Multi-model	135.35	+0.57	-6.28
8.	8.	7.	IBM Db2	Relational, Multi-model	128.46	+0.97	-14.56
9.	9.	11.	Snowflake	Relational	121.33	-1.87	+9.61
10.	10.	9.	SQLite	Relational	114.32	-1.69	-19.54

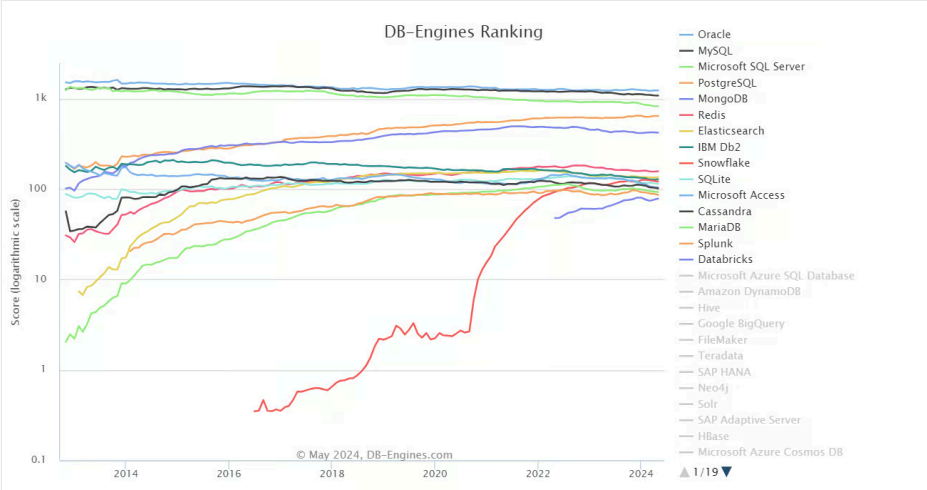
Source: **db-engines**

It's worth noting that NoSQL databases like MongoDB and Redis also hold strong positions in the ranking, reflecting the growing demand for flexible and scalable solutions capable of handling unstructured data and high-traffic applications. These NoSQL systems have seen significant year-over-year growth, indicating a shift towards more diverse database architectures.

The ranking also reveals the rising popularity of cloud-based databases like Snowflake, which offers a fully managed, scalable data warehouse solution. Elasticsearch, a powerful search engine and analytics platform, has also climbed in the rankings, underscoring the importance of search and analytics capabilities in modern data management.

Database Popularity Trends

Let's now look at the line graph below, which illustrates the dynamic landscape of database popularity from 2014 to 2024.



Source: **db-engines**

A key takeaway is the enduring dominance of relational database management systems (RDBMS) like Oracle, MySQL, Microsoft SQL Server, and PostgreSQL. These have consistently maintained their top positions throughout the decade, highlighting their importance in handling structured data and supporting complex queries across various applications.

However, the graph also reveals a notable shift in recent years. While RDBMS systems have seen a gradual decline in popularity, NoSQL databases like MongoDB and Redis have experienced significant growth. This upward trajectory reflects the increasing adoption of these flexible and scalable solutions for managing unstructured data and accommodating high-traffic applications.

Another interesting trend is the rise of cloud-based databases. Databricks, a cloud-based data engineering and machine learning platform, has skyrocketed in popularity, showcasing the growing demand for cloud-based solutions that offer scalability, ease of use, and powerful analytics capabilities.

Similarly, Snowflake, a fully managed **cloud data warehouse**, has seen significant growth, highlighting the appeal of its scalable and easy-to-use architecture.

Relational Databases

Relational databases store data in tables structured into rows and columns. Each row represents a unique record, and each column represents a specific attribute of that record.

Imagine them as meticulously organized spreadsheets, where data is stored in tables comprised of rows (records) and columns (attributes). Each row represents a distinct entity, like a customer or a product, while each column captures a specific characteristic, such as a name, address, or price.

The real power of relational databases lies in their ability to link these tables together using relationships. These relationships, established through foreign keys, allow us to connect data from different tables, creating a unified view of information.

For example, in a customer relationship management (CRM) system, a customer table might be linked to an orders table, enabling us to track a customer's purchase history.

Structured query language (SQL)

To interact with relational databases, we use Structured Query Language (SQL). This powerful language enables us to query, insert, update, and delete data, as well as perform complex operations like joining data from multiple tables. SQL's structured nature ensures data integrity and consistency through ACID properties:

- **Atomicity:** All operations within a transaction are treated as a single unit, ensuring that either all changes are committed or none are.
- **Consistency:** Data remains in a valid state throughout a transaction, adhering to predefined constraints and rules.

- **Isolation**: Transactions are executed independently as if they were the only operation happening on the database.
- **Durability**: Once a transaction is committed, its changes are permanent, even in the event of system failures.

If you want to learn more about SQL, check out this seven-course skill track on **SQL Fundamentals**.

When to use relational databases

Relational databases are great when we need:

- **Strong consistency**: Ensuring all users see the same data simultaneously.
- **Complex queries**: Joining data from multiple tables to gain insights.
- **ACID compliance**: Guaranteeing reliable transaction processing for critical applications.

However, they might not be the best fit for:

- **Unstructured data**: Handling data that doesn't fit neatly into a tabular format (e.g., social media posts, sensor data).
- **Massive scalability**: When your application needs to scale horizontally across numerous servers.

Popular relational databases

Some popular RDBMS options include:

- **MySQL**: Open-source and known for its ease of use, speed, and reliability, often used in web applications.
- **PostgreSQL**: Open-source and highly extensible, offering advanced features and strong compliance with SQL standards.
- **Oracle Database**: A comprehensive, enterprise-grade solution known for its performance, scalability, and security.
- **Microsoft SQL Server**: Tightly integrated with the Microsoft ecosystem, offering a wide range of tools for business intelligence and analytics.

If you want to learn how to use relational databases in Python, check out this free course on **Introduction to Databases in Python**.

NoSQL Databases

NoSQL databases, short for "not only SQL," have emerged as a powerful alternative to relational databases, particularly in scenarios where flexibility, scalability, and high performance are paramount.

Unlike their relational counterparts, NoSQL databases can handle unstructured or semi-structured data without the constraints of a fixed schema. This means we can store data in various formats, such as JSON documents, key-value pairs, or graph structures, without having to define a rigid structure upfront.

These databases often provide features to scale out across multiple servers and clusters, making them suitable for distributed data environments.

Querying NoSQL databases

Unlike relational databases, which use Structured Query Language (SQL), NoSQL databases don't have a universal query language. Instead, each type of NoSQL database typically has its unique query language or API tailored to its specific data model and structure.

While NoSQL databases prioritize flexibility and scalability, they often relax some of the ACID properties found in relational databases. For example, some NoSQL databases prioritize eventual consistency over immediate consistency, meaning that changes might not be reflected across all nodes instantly. This trade-off allows for better performance and scalability but requires careful consideration when designing applications that rely on strict data consistency.

If you want to learn how to query NoSQL databases, check out this **Introduction to NoSQL** course.

When to use NoSQL databases

NoSQL databases are particularly well-suited for scenarios where:

- **Agility is key**: Rapid development cycles and evolving data models.
- **Scale is a priority**: Applications with exponential data growth or high traffic.
- **Performance matters**: Real-time applications requiring fast read/write operations.
- **Variety is the norm**: Diverse data types (e.g., social media posts, sensor data).

Common use cases include:

- **Big data analytics**: Processing massive datasets.
- **Real-time applications**: Delivering up-to-the-minute information.
- **Content management systems**: Storing and managing diverse content.
- **Internet of Things (IoT)**: Handling continuous data streams.
- **Personalization engines**: Tailoring user experience.

While offering significant advantages, NoSQL databases might not be ideal for applications requiring strong transactional guarantees or complex relational queries. Many organizations adopt a hybrid approach, using both relational and NoSQL databases to leverage their respective strengths.

Popular NoSQL databases

Some of the most popular NoSQL databases include:

- **MongoDB**: A document-oriented database that is great for storing JSON-like documents with dynamic schemas.
- **Redis**: A key-value store often used for caching and as a fast in-memory datastore.
- **Cassandra**: A column-family store known for its scalability and fault tolerance.
- **Neo4j**: A graph database that excels in managing and querying highly connected data.

If you want to learn more about the four major NoSQL databases, check out this course on **NoSQL Concepts**.

Cloud Databases

Cloud databases have revolutionized data management by leveraging the vast resources and scalability of cloud computing platforms. These databases reside on remote servers and are accessed over the internet, eliminating the need for organizations to invest in and maintain their own hardware and infrastructure.

Cloud databases operate on a *pay-as-you-go* model, where we only pay for the resources we actually use. This eliminates the upfront costs and ongoing maintenance expenses associated with traditional on-premises databases. Cloud providers handle the underlying infrastructure, including servers, storage, and networking, while you focus on building and managing your applications.

If you want to learn about cloud computing, check out this course on **Understanding Cloud Computing**.

Querying cloud databases

Querying cloud databases typically involves using the same tools and languages we'd use with on-premises databases. For relational databases in the cloud, we'd use SQL to interact with the data. NoSQL databases in the cloud typically have their own query languages or APIs, similar to their on-premises counterparts.

Cloud providers often offer additional tools and services to simplify database management and querying. These might include web-based consoles, command-line interfaces, and SDKs for various programming languages.

When to use cloud databases

Cloud databases are an excellent choice when:

- **Scalability is crucial**: Easily adapt to changing demands.
- **Flexibility is a priority**: Wide range of database options available.
- **Global accessibility is important**: Low-latency access for users worldwide.
- **Cost-effectiveness is a concern**: Pay-as-you-go model and scalable resources.

Popular cloud databases

The leading cloud providers offer a range of database services, each with its own strengths and specialties:

- **Amazon RDS**: Supports multiple database engines like MySQL, PostgreSQL, and Oracle, offering managed relational database services.
- **Google Cloud SQL**: A fully-managed service that allows running MySQL, PostgreSQL, and SQL Server databases in the cloud.
- **Azure SQL Database**: Provides scalable, intelligent, and fully-managed database services in the Microsoft Azure cloud.

You can learn more about cloud databases in this course on **AWS Cloud Technology and Services**.

Vector Database

Vector databases have emerged as a specialized tool for handling the unique demands of artificial intelligence and machine learning applications.

Vector databases are designed to store, index, and manage vector embeddings, which are high-dimensional data representations often used in machine learning models. This enables efficient similarity search, where the database can quickly identify vectors that are "close" to a given query vector based on distance metrics like cosine similarity or Euclidean distance.

These features make them suitable for applications like image recognition, recommendation systems, and natural language processing. They utilize indexing structures that optimize the retrieval of similar vectors based on distance metrics.

If you want to learn more about vector databases, you can read this article: **An Introduction to Vector Databases for Machine Learning**.

1. **Embedding the query**: The input query (e.g., an image, a piece of text) is converted into a vector embedding using an appropriate embedding model.
2. **Similarity search**: The vector database performs a similarity search to find the nearest neighbors of the query embedding in the vector space. This is often done using approximate nearest neighbor (ANN) algorithms to ensure efficiency at scale.
3. **Returning results**: The database returns the identified nearest neighbors along with their associated metadata or original data objects.

Different vector databases may offer various query options and parameters, such as specifying the number of nearest neighbors to return or setting a distance threshold. Some databases also support filtering based on metadata or combining vector search with traditional scalar filtering.

When to use vector databases

Vector databases are particularly well-suited for scenarios where:

- **Similarity search is critical**: Applications like image recognition or recommendation systems.
- **High-dimensional data is involved**: Inefficient for traditional databases.

- **Real-time performance is required:** AI applications like recommender systems.

Popular vector databases

- **Faiss:** Developed by Facebook AI Research, it provides efficient similarity search and clustering of dense vectors.
- **Milvus:** An open-source vector database that supports scalable similarity search and AI applications.
- **Pinecone:** A vector database service that simplifies the deployment and scaling of similarity search in production environments.

If you want to learn more about what the popular databases are, check out this article on **Top 5 Best Vector Databases**.

Other Types of Databases

While relational, NoSQL, cloud, and vector databases cover a wide range of use cases, several other database types exist, each tailored to specific data models and access patterns. Let's briefly explore some of these specialized solutions.

Time-series databases

Time-series databases are optimized for storing and analyzing time-stamped data, such as sensor readings, stock prices, or server logs. They excel at handling high-volume data ingestion and efficiently querying data points based on time ranges. Popular options include InfluxDB, TimescaleDB, and Prometheus.

Object-oriented databases

Object-oriented databases (OODBs) store data as objects, similar to object-oriented programming. This can simplify modeling complex data structures and relationships. However, OODBs have not gained widespread adoption due to challenges with standardization and query optimization. Popular options include ObjectDB and Versant Object Database.

Graph databases

Graph databases excel at representing and querying relationships between entities. They store data as nodes (entities) and edges (relationships), making them well-suited for social networks, recommendation engines, fraud detection systems, and knowledge graphs. Popular options include Neo4j, Amazon Neptune, and JanusGraph.

Hierarchical databases

Hierarchical databases organize data in a tree-like structure, with parent-child relationships between records. This structure is suitable for some specialized applications but can be inflexible for complex data models. While historically significant, hierarchical databases are less common in modern applications.

Network databases

Network databases are similar to hierarchical databases but allow for more complex relationships between records. While they offer flexibility, they can also be more challenging to manage and query. Network databases have largely been replaced by relational and graph databases in most applications.

Conclusion

In this overview, we've explored the diverse landscape of databases, each type tailored to address specific data challenges. From structured data in relational databases to the flexibility of NoSQL, the scalability of cloud solutions, and the specialized capabilities of vector databases, we've seen how these tools underpin modern data management.

Choosing the right database is a critical decision, one that depends on understanding the unique strengths and tradeoffs of each type. By carefully evaluating your specific needs and constraints, you can select the database that best empowers your data-driven applications and initiatives.

If you want to go into more detail about databases, you can try this four-course skill track on **SQL for Database Administrators**.