The future of the database

Twiddling thumbs between sessions at a database conference, **Mayank Sharma** ponders on the impact of the web on databases...

This is the information era, where almost everything is data. The scale at which this data is being collected and used is growing at an exponential rate. The speed at which organisations are ingesting, storing and processing data is hard to fathom; it's very common these days for even small-scale companies to process terabytes and even petabytes of data. In order for this data to be crunched into meaningful information, it first needs to be housed in a database.

Databases have become a part of your daily routine in more ways than you can imagine. These days you don't even have to be on a computer or use your smartphone to interact with a database. For instance, when you purchase items at the local supermarket, chances are there's an inventory database that automatically updates itself while you checkout. The same thing happens when you borrow a book from the library, withdraw cash from the ATM or buy a movie ticket – the list is virtually endless. In fact, it's safe to say that a majority of your daily chores will involve some sort of interaction with a database.

In addition to these traditional uses of a database that are fulfilled by what is known as a relational database, the internet has had a great influence on databases in both form and function. The explosion of social media platforms such as Twitter, Facebook, Instagram and the like have ushered in a new generation of databases that are designed to overcome the scalability limitations of the earlier varieties. These hold vast quantities of different kinds of data that just cannot be stored and processed, at least not efficiently, by traditional relational database systems.

Taking a look at the evolution of the mechanisms for data storage, retrieval and processing will help us better appreciate the challenges of an often unappreciated and unglamorous branch of computer science.

We've been using databases to help us organise information since time immemorial. Archaeologists have found stone tablets in digs dating to 4000 BC that were used to index various kinds of information. Before the advent of the computer, we were cataloguing information manually. The first computer databases were just digital versions of this manual system: a flat file of a consecutive list of records. While filing information was straightforward, search and retrieval was a slow, time-consuming process. In the mid-1960s IBM started using a hierarchical data model for its information management system, called IMS. It featured a parent node that pointed to several child nodes. IMS was famously used by NASA to help with the design of the Lunar Lander.

In 1969, the Committee on Data Systems Languages (CODASYL) consortium, which was a group of scientists and researchers working with the COBOL programming language, got together and came up with a standard interface for how COBOL programs should access and share databases. The lead proponent of the CODASYL group was Charles Bachmann, who tweaked the hierarchical model and made it more flexible by establishing what's known as the network data model, enabling child nodes to have multiple parents.

The CODASYL approach was a very complicated system to execute and required substantial training. Edgar 'Ted' Codd was a mathematician at IBM who saw programmers wasting time rewriting programs every time there were any changes to the layout of the database. He proposed a database abstraction approach that separated the logical and the physical structure of the database.

His relational database model, first proposed in 1970, organised a body of data into simple tables of related information. Instead of a freeform list of linked records, Codd proposed data to be stored in tables with fixed-length records. This, along with several other changes, made it easier to access, append and modify data. His words resonated with C. J. Date, an instructor at IBM, and together the duo authored several papers on relational databases.

Missed opportunity

Despite both Codd and Date being IBM employees, the company wasn't prepared to support their idea since it already had a successful database product in IMS. But that didn't stop others from jumping on the relational database model. In 1973, two researchers at UC Berkeley, Michael Stonebraker and Eugene Wong, built on Codd's idea to create the Ingres (Interactive Graphics and Retrieval System) database. Ingres worked with a query language known as QUEL.

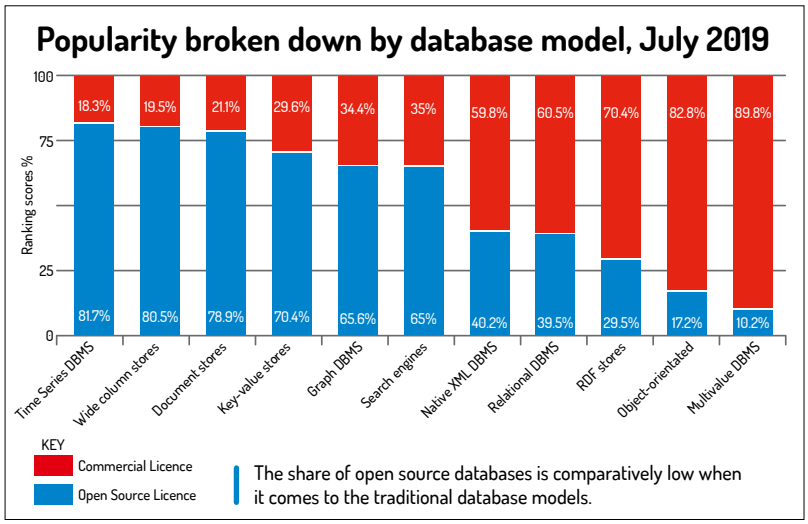
Several companies used Ingres as the basis for successful commercial products, although it took until 1975 for IBM to produce an experimental relational database called System R. It used a structured query language (SQL), developed by IBM's Don Chamberlin and Raymond Boyce, to search and modify data. SQL quickly replaced QUEL as a more functional query language and became an ANSI standard towards the end of the 1980s.

The third major adaptation of Codd's idea came about in 1977, when Larry Ellison got together with Bob Miner and Ed Oats in order to commercialise the relational database. They shipped the Oracle database in 1979 – the first commercially available relational database. It became the dominant format and didn't leave much room for anyone else by the time IBM released a full-fledged commercial relational database in the form of DB2 in 1983.

The 1980s saw the development of various commercial relational databases as their benefits became more widely known. The first relational databases were quite slow, especially while accessing data records. However, their performance improved with the development of new storage and indexing techniques and better query processing and optimisation. Eventually, relational databases became the dominant type of database system for what are now referred to as traditional database applications.

Open sesame

One of the key ingredients that fuelled the success of the relational database model was open source code. It started with the Ingres database in 1974, which made its code available for a small fee under the BSD licence. The Ingres code spawned a number of popular and commercially successful databases such as Sybase



» THE DAWN OF SKYNET?

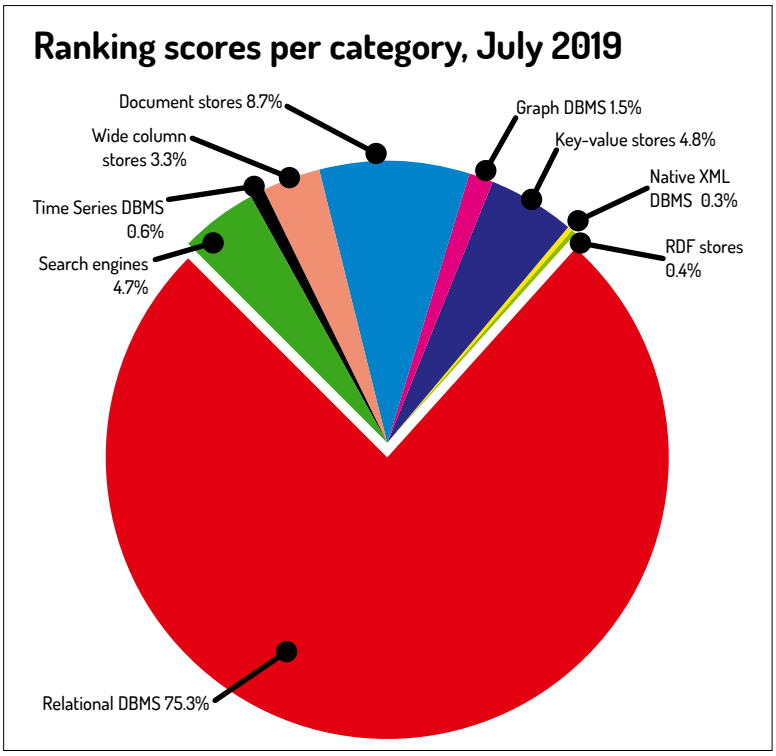
Artificial Intelligence (AI) and machine learning (ML) have entered the mainstream in the last couple of years. These technologies are now making their way into the next generation of databases as well. Administering large databases that operate complex workloads isn't a simple feat, considering the amount of configuration settings that need to be managed. An increasing number of database vendors are thus infusing AI and ML in their databases in order to relegate some regular monitoring and optimisation tasks to an autopilot that can then assist admins tune the databases for maximum performance.

"For decades, the primary time-sink for administrators has been routine maintenance and performance monitoring/tuning tasks that ensure each database system is optimised for its supported applications," says Robin Schmacher. "The promise of well-designed AI and ML functionality in databases is that it will remove this productivity drain from the IT staff and have the database maintain and tune itself – all the while learning internally what proper optimisation is for each ind ividual database." Thus databases in the very near future will be able to anticipate operational issues and take preventative actions automatically. (*destroy meat-bag infestation–ED*)

and Microsoft SQL Server, among others. After a brief hiatus, Michael Stonebraker returned to Berkeley in 1985 and began work on the next generation of the Ingres database to address some of the bottlenecks with the original design. His work, dubbed Postgres (Post Ingres), eventually evolved into PostgreSQL.

The next milestone for open source databases came in the mid 1990s, when David Hughes wanted a database for his network monitoring and management system called Minerva. To meet his requirements, he developed mSQL, first as an SQL extension to Postgres, and then as a lightweight database with a limited subset of the SQL standard. When mSQL development began to stagnate in 1996, its open source code was adapted into MySQL. After the acquisition of MySQL AB by Oracle, the code was forked into MariaDB in 2010.

Another popular database that came about as a result of open source code was Firebird. It was forked from the open source code of InterBase 6.0 in 2000. The database introduced several performance improvements, and is the default database engine in LibreOffice.



Order in chaos

The first chinks in the armour of the relational databases began to appear with the dot-com boom in the 2000s. The proliferation of online applications and platforms such as social media websites, e-commerce vendors, cloud storage silos and others led to a huge surge in the amount of data that now had to be stored in large databases and massive servers. As the nature of the data changed, relational databases – which had mastered the art of storing and processing structured data – were incapable of efficiently handling the vast amounts of non-relational, schema-less unstructured data.

This called for a new type of database system that could provide fast search and retrieval, as well as reliable and safe storage of the unstructured, non-traditional data. NoSQL came about as a response to the need for faster speed and the processing of this new unordered and disorganised data. The term NoSQL is most often interpreted as Not Only SQL. It points to the fact that in these databases, some of the data is stored using SQL systems, whereas other data is stored using NoSQL, depending on the requirements of the application.

The highlight of the NoSQL data model is that it is non-relational and uses a distributed database system. It is fast, uses an ad hoc method of organising data, and is efficient at processing different kinds of data at high-volumes. The widespread use of NoSQL can be attributed to the services offered by popular social media platforms and cloud services such as Twitter, Facebook, LinkedIn, Instagram and Google. These online platforms store and process colossal amounts of unstructured data.

Just as open source helped fuel the evolution of relational databases, it essayed a similar role in helping

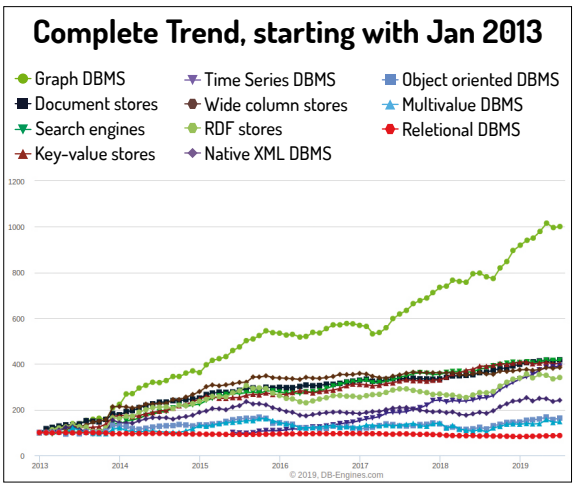


Image Credit: DB-Engines.com

As you can see, graph databases have been steadily rising, while the popularity of relational databases has plateaued in the last five years.

TYPES OF DATABASES

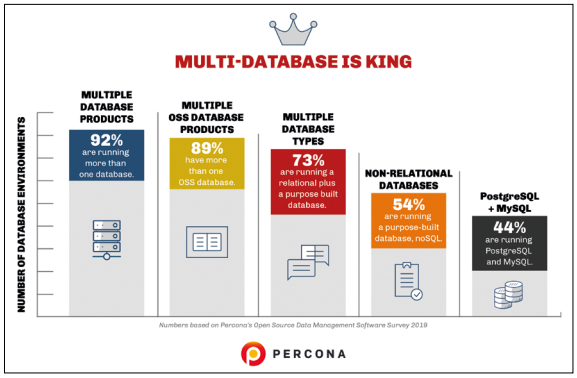
NoSQL databases provide the performance, scalability and stability that's required by the modern data-driven apps we interact with these days. But that is where the similarity between NoSQL systems end. In fact, it wouldn't be wrong to say that the only thing most NoSQL databases have in common is that they do not follow the traditional relational data model. Broadly speaking, NoSQL databases typically fall into one of four categories:

Key-value These function in heavy read environments. Key-value stores are the simplest form of NoSQL databases and won't be of much help when there are complex relationships between data elements. That's because all access to the database is done using a primary key. Redis and Memcached are frequently used solutions.

Columnar These are optimised for reading and writing columns of data as opposed to rows of data. Also known as wide-column store databases, they are well suited for analysing huge data sets. Apache Cassandra, HBase and Accumulo are the best-known ones.

Document These store, manage and retrieve data as semi-structured documents such as JSON and XML. These document-orientated databases store all information for a given object, and each object can be quite different from the others. MongoDB, Couchbase and CouchDB are the best known and most widely used.

Graph Focus on how data relates to other data points. These databases explore the relationships that link data. Data from graph databases stores directed link between data sets called edges. Neo4j, OrientDB and JanusGraph are popular examples.



Percona Live is a yearly conference dedicated to open source databases. It will be held in Amsterdam this year from 30 September to 2 October.

democratise NoSQL databases. The term NoSQL can be traced back to 1998 when Carlo Strozzi released his lightweight open source Strozzi NoSQL database, which was relational but didn't use SQL as the query language. 'NoSQL' as it is used today originated in 2009 when it was used to describe the emergence of new databases that do not view data as strictly defined tables of rows and columns. It sought to encapsulate the clones of Google's Bigtable and Amazon's DynamoDB that had started popping up around that time.

One of the developers of DynamoDB, Avinash Lakshman, helped develop a NoSQL database at Facebook that was eventually released as Cassandra under an open source licence in 2008. But it was MongoDB that made the industry stand up and take notice of open source non-relational databases as viable solutions. Unlike traditional databases, MongoDB is a document database that stores related data clumped together in chunks.

Another approach to catalogue the complex distributed data relationships in the social-media-dominated web came about in the form of graph databases. One of the oldest graph databases is the open source Neo4j. Yet another open source NoSQL database that's making heads turn is Redis. It's a simple database that employs what's known as a key-value store. Redis' unique selling point is its stellar performance, since it operates entirely from memory.

The popularity of open source databases can be gauged from the DB-Engines rankings (**https://db-engines.com/en/ranking**). It tracks 350 databases, of which 172 are open source, as compared to 178 commercial systems. Between them, all types of NoSQL databases are dominated by open source databases – see the table, right. When you drill down inside each category, you'll find an open source database at the top. Redis leads the key-value database ranking, MongoDB tops the document database ranking, Neo4j is the most popular graph database, InfluxDB outscores the other time-series databases and Cassandra is the top ranking wide column database.

This comes as no surprise to Matt Yonkovit, Chief Experience Officer at Percona. He explains that open source databases are popular since "the barrier for entry for open source is extremely low – there is no need for budget approval with free software, there are lots of people with the right skills available, and support for these systems is at hand for production applications. When you are going to start a new project, having access to enterprise-ready tools and databases to build not only a proof of concept but also the actual applications enables faster time to value."

Robin Schumacher, SVP and Chief Product Officer, DataStax, agrees: "One of the key benefits the open source development model offers is being able to draw on the innovative talent and efforts of everyone in the technical community, and have their contributions rapidly come together to form a meaningful technology that's open and available to anyone. Of course, the fact that such software is free to use helps adoption."

With the rise of widely distributed cloud apps generating copious amounts of unstructured data, it's the unprecedented workload that's fuelling the growth of NoSQL databases designed to operate and function efficiently at this level. But neither Robin nor Matt is writing off relational databases just yet. Robin believes that the relational database remains a "crucial component in today's infrastructure stack". He adds that relational databases will continue to be useful "for applications that are centralised in nature, are best handled by a relational data model, don't need to scale past singular hardware deployments, and don't have 100 per cent uptime requirements."

The advantages that open source databases offer over traditional closed source, proprietary ones are twofold. On the one hand they equip users with powerful solutions without raiding their pockets. On the other, they enable vendors to build a sustainable business model around open source software. It's a feedback loop, as the vendor then ploughs back some of its revenue to fund the development of the open source database. LXF

Position	Database	Type	Score
1	Oracle	Relational, Multi-model	1321.26
2	MySQL	Relational, Multi-model	1229.52
3	Microsoft SQL Server	Relational, Multi-model	1090.83
4	PostgreSQL	Relational, Multi-model	483.28
5	MongoDB	Document	409.93
6	IBM Db2	Relational, Multi-model	174.14
7	Elasticsearch	Search engine, Multi-model	148.81
8	Redis	Key-value, Multi-model	144.26
9	Microsoft Access	Relational	137.31
10	Cassandra	Wide column	127
11	SQLite	Relational	124.63
12	Splunk	Search engine	85.49
13	MariaDB	Relational, Multi-model	84.44
14	Hive	Relational	80.87
15	Teradata	Relational, Multi-model	77.83
16	Solr	Search engine	59.64
17	FileMaker	Relational	57.9
18	HBase	Wide column	57.54
19	SAP Adaptive Server	Relational	56.65
20	Amazon DynamoDB	Multi-model	56.42
21	SAP HANA	Relational, Multi-model	55.54
22	Neo4j	Graph	48.98

Data from: DB-Engines.com

All things considered, 12 out of the top 22 databases in use today are open source.