Storage of information in the form of computerized databases started in early 1960s, when various governmental offices, businesses, libraries etc. saw the cost-effective option with the use of computers. The most popular data models currently were the network database model, the hierarchical database model, and the flat-file database model [3]. Many of these models had various disadvantages such as complex representation of database, lack of structural independence and no support for many to many relationships [4].

RDBMS overcame most of these limitations by providing simple database design, a language to access data easily, support for large amounts of data with multi-level relationships between data sets and many more. Dr. E.F. Codd, an IBM research scientist and mathematician, designed the relational model. Also, in this model, the database’s schema is disconnected from physical information storage, and this became the standard principle for database systems [3]. The main focal points of this model were to further reduce data redundancy and to improve data integrity within database systems [1]. In RDBMS, the data is stored in tables, or relations and these tables are divided into rows (records) and columns (fields). Relational databases are famous for handling multitudes of data and complex queries.

Both RDBMS and NoSQL databases provide various benefits of their own and aim to solve a single problem i.e. an efficient way to store information, but neither is a universal solution for every application. The developer must learn the advantages and disadvantages of each and choose the database based on the optimal set of tradeoffs that one can make based on their application. To explicate our analysis and to establish the differences between RDBMS and NoSQL, we choose PostgreSQL, a free and open-source RDBMS and MongoDB, a document-oriented NoSQL database. Both the databases are multi-platform software.

POSTGRESQL

PostgreSQL (postgres) is a powerful and object-relational database system that uses and extends the SQL language combined with many features that safely store and scale the most complicated data workloads. The origins of PostgreSQL date back to 1986 as part of the POSTGRES project at the University of California at Berkeley and has more than 30 years of active development on the core platform.

PostgreSQL has earned a strong reputation for its proven architecture, reliability, data integrity, robust feature set, extensibility, and the dedication of the open source community behind the software to consistently deliver performant and innovative solutions. PostgreSQL runs on all major operating systems, has been ACID-compliant since 2001, and has powerful add-ons such as the popular PostGIS geospatial database extender. It is no surprise that PostgreSQL has become the open source relational database of choice for many people and organisations.

Getting started with using PostgreSQL has never been easier - pick a project you want to build, and let PostgreSQL safely and robustly store your data.

PostgreSQL comes with [**many features**](https://www.postgresql.org/about/featurematrix/) aimed to help developers build applications, administrators to protect data integrity and build fault-tolerant environments, and help you manage your data no matter how big or small the dataset. In addition to being [**free and open source**](https://www.postgresql.org/about/license/), PostgreSQL is highly extensible. For example, you can define your own data types, build out custom functions, even write code from [**different programming languages**](https://www.postgresql.org/docs/current/xplang.html) without recompiling your database!

PostgreSQL tries to conform with the [**SQL standard**](https://www.postgresql.org/docs/current/features.html) where such conformance does not contradict traditional features or could lead to poor architectural decisions. Many of the features required by the SQL standard are supported, though sometimes with slightly differing syntax or function. Further moves towards conformance can be expected over time. As of the version 13 release in September 2020, PostgreSQL conforms to at least 170 of the 179 mandatory features for SQL:2016 Core conformance. As of this writing, no relational database meets full conformance with this standard.

Below is an inexhaustive list of various features found in PostgreSQL, with more being added in every [**major release**](https://www.postgresql.org/developer/roadmap/):

* **Data Types**
  + Primitives: Integer, Numeric, String, Boolean
  + Structured: Date/Time, Array, Range, UUID
  + Document: JSON/JSONB, XML, Key-value (Hstore)
  + Geometry: Point, Line, Circle, Polygon
  + Customizations: Composite, Custom Types
* **Data Integrity**
  + UNIQUE, NOT NULL
  + Primary Keys
  + Foreign Keys
  + Exclusion Constraints
  + Explicit Locks, Advisory Locks
* **Concurrency, Performance**
  + Indexing: B-tree, Multicolumn, Expressions, Partial
  + Advanced Indexing: GiST, SP-Gist, KNN Gist, GIN, BRIN, Covering indexes, Bloom filters
  + Sophisticated query planner / optimizer, index-only scans, multicolumn statistics
  + Transactions, Nested Transactions (via savepoints)
  + Multi-Version concurrency Control (MVCC)
  + Parallelization of read queries and building B-tree indexes
  + Table partitioning
  + All transaction isolation levels defined in the SQL standard, including Serializable
  + Just-in-time (JIT) compilation of expressions
* **Reliability, Disaster Recovery**
  + Write-ahead Logging (WAL)
  + Replication: Asynchronous, Synchronous, Logical
  + Point-in-time-recovery (PITR), active standbys
  + Tablespaces
* **Security**
  + Authentication: GSSAPI, SSPI, LDAP, SCRAM-SHA-256, Certificate, and more
  + Robust access-control system
  + Column and row-level security
  + Multi-factor authentication with certificates and an additional method
* **Extensibility**
  + Stored functions and procedures
  + Procedural Languages: PL/PGSQL, Perl, Python (and many more)
  + SQL/JSON path expressions
  + Foreign data wrappers: connect to other databases or streams with a standard SQL interface
  + Customizable storage interface for tables
  + Many extensions that provide additional functionality, including PostGIS
* **Internationalisation, Text Search**
  + Support for international character sets, e.g. through ICU collations
  + Case-insensitive and accent-insensitive collations
  + Full-text search

There are many more features that you can discover in the PostgreSQL [**documentation**](https://www.postgresql.org/docs/). Additionally, PostgreSQL is highly extensible: many features, such as indexes, have defined APIs so that you can build out with PostgreSQL to solve your challenges.

PostgreSQL has been proven to be highly scalable both in the sheer quantity of data it can manage and in the number of concurrent users it can accommodate. There are active PostgreSQL clusters in production environments that manage many terabytes of data, and specialized systems that manage petabytes.

[3] https://www.quickbase.com/articles/timeline-of-database-history#:~:text=1980s,network%20and%20hierarchical%20database%20models.

[4] <https://www.ukessays.com/essays/information-technology/hierarchical-data-model.php>