PostgreSQL Research Report

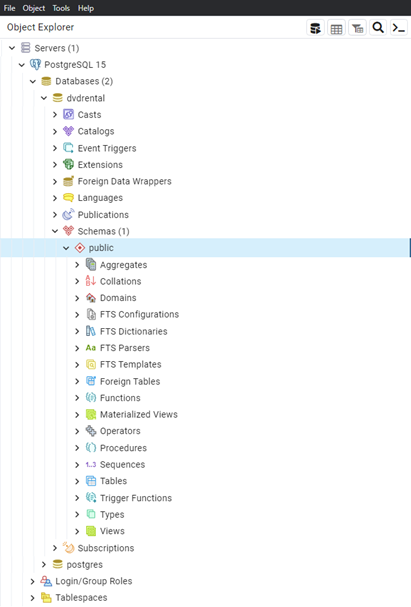
**Table of contain**

* Database Architecture of PostgreSQL
* SQL features
* Extensions and Ecosystem of PostgreSQL
* Server Administration
  + 1. [Connections and Authentication](https://www.postgresql.org/docs/current/runtime-config-connection.html)
    2. [Replication](https://www.postgresql.org/docs/current/runtime-config-replication.html)
    3. [Query Planning](https://www.postgresql.org/docs/current/runtime-config-query.html)
    4. Write Ahead Log
* Concurrency Control
* Performance
* Security / Authentication
* Backup – Restore
* Mysql vs PostgreSQL

* Quick overview PostgreSQL
  + PostgreSQL is an open-source project. As such, it depends on the user community for ongoing support.
  + PostgreSQL is used as a primary database
  + PostgreSQL with the [PostGIS extension](https://postgis.net/" \t "_blank) supports geospatial databases for geographic information systems.
  + **supports procedural language**. Procedural languages in PostgreSQL are programming languages that can be used to write stored procedures, functions, and triggers directly within the database. PostgreSQL supports multiple procedural languages, allowing developers to choose the language they are most comfortable with for writing database logic.
* PostgreSQL feature highlights
* User-defined types
* Table inheritance
* Sophisticated locking mechanism
* [Foreign key referential integrity](https://www.postgresqltutorial.com/postgresql-foreign-key/)
* [Views](https://www.postgresqltutorial.com/postgresql-views/), rules, [subquery](https://www.postgresqltutorial.com/postgresql-tutorial/postgresql-subquery/)
* Nested transactions (savepoints)
* Multi-version concurrency control (MVCC)
* Asynchronous replication
* ACID Compliance
* Geospatial Capabilities
* [Template structure](https://www.postgresql.org/docs/current/manage-ag-templatedbs.html) for database
* Object-oriented database features
* Define your own complex data types
* Overload functions to work with different argument data types
* Define inheritance relationships between tables

# 1. Database Architecture of PostgreSQL

* + PostgreSQL uses a client/server model.



* + **Database Cluster:**
    - A PostgreSQL database cluster is a collection of databases managed by a single PostgreSQL instance. It consists of one or more physical files, known as data files or tablespaces, that store the actual data.
  + **Database:**
    - A database in PostgreSQL is a logical container for storing related data. Each database within a cluster is isolated from others and can have its own set of tables, views, functions, and other database objects.
  + **Tables:**
    - Tables are the basic storage units in PostgreSQL, where data is organized and stored in rows and columns.
  + **Schemas**:
    - Schemas provide a way to organize database objects within a database. They act as containers for tables, views, functions, and other objects. Schemas can be used to logically group related objects and provide namespace isolation.
  + **Transactions:** 
    - PostgreSQL follows the ACID (Atomicity, Consistency, Isolation, Durability) properties to ensure data integrity and reliability. Transactions allow multiple database operations to be executed as a single atomic unit, ensuring consistency and isolation.
  + **Indexes:**
    - PostgreSQL supports various index types, such as B-tree, Hash, GIN (Generalized Inverted Index), GiST (Generalized Search Tree), and SP-GiST (Space-Partitioned Generalized Search Tree). Indexes improve query performance by providing faster data access and enabling efficient data retrieval based on specific criteria.
  + **Locking and Concurrency Control:**
    - PostgreSQL uses a multi-version concurrency control (MVCC) mechanism to handle concurrent transactions. MVCC allows multiple transactions to read and modify the database simultaneously without blocking each other. PostgreSQL uses row-level locks to manage concurrent access and ensure data consistency.

# 2. SQL features

* **Data Types**
* Primitives: Integer, Numeric, String, Boolean
* Structured: Date/Time, Array, Range / Multirange, UUID(Universally Unique Identifier)
* 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
* Explicit Locks, Advisory Locks
* **Data Definition**
* Default values, generated columns, Constraints,
* Privileges
* Schemas
* Inheritance
* Table Partitioning
* Foreign Data
* **Inheritance, 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 Tcl. There are other languages available through extensions, e.g. Java, JavaScript (V8), R, Lua, and Rust
* Foreign data wrappers: connect to other databases or streams with a standard SQL interface
* Customizable storage interface for tables
* **Internationalisation, Text Search**
* Support for international character sets, e.g. through ICU collations
* Case-insensitive and accent-insensitive collations
* Full-text search
* [**Parallel Query**](https://www.postgresql.org/docs/current/parallel-query.html)
* ostgreSQL can devise query plans that can leverage multiple CPUs in order to answer queries faster. This feature is known as parallel query.
* [**User-Defined Functions**](https://www.postgresql.org/docs/current/xfunc.html) **& Procedures :**
* using different programming languages, including SQL, PL/pgSQL, PL/Python, PL/Perl, PL/Java, and more.
* They can be used in SQL queries, stored procedures, triggers, and other database operations
* **Function Overloading**
* [**User-Defined Aggregates**](https://www.postgresql.org/docs/current/xaggr.html)**:**
* User-Defined Aggregates extend the built-in aggregate functions in PostgreSQL to support custom aggregation operations.

# 3. Extensions and Ecosystem of PostgreSQL

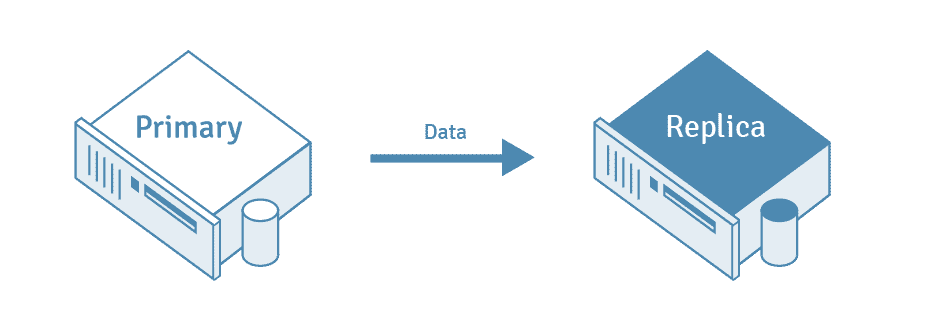
* **Extensions:**

PostgreSQL supports a wide range of extensions that provide additional features and capabilities beyond the core database functionality. Extensions can be developed by the PostgreSQL community or by third-party developers. Some popular extensions include:

* + **PostGIS:** Adds support for geographic objects and spatial queries, enabling the storage and analysis of geospatial data.
  + **pgcrypto:** Provides cryptographic functions for encrypting and decrypting data within the database.
  + **pgRouting**: Enables routing and network analysis capabilities, allowing you to perform pathfinding and navigation tasks.
  + **TimescaleDB:** Optimizes PostgreSQL for time-series data storage and analysis, offering performance improvements and advanced time-based queries.
* **Foreign** **Data Wrappers (FDW):**
  + PostgreSQL supports FDWs, which allow you to access and query remote data sources as if they were local tables. FDWs provide integration with external databases, web services, and other data sources, enabling you to fetch and combine data from different systems seamlessly.
* **Full-Text Search:**
  + PostgreSQL has built-in support for full-text search, but it also offers extensions like pg\_trgm and pg\_bigm, which provide enhanced text search capabilities. These extensions enable fuzzy matching, similarity ranking, and advanced indexing techniques for efficient searching.
* **Procedural Languages:**
  + PostgreSQL supports multiple procedural languages, such as PL/pgSQL, PL/Python, PL/Perl, PL/Java, and more. These languages allow you to write stored procedures, user-defined functions, and triggers in different programming languages, expanding the flexibility and expressiveness of your database logic.
* **Tools and Frameworks:** 
  + The PostgreSQL ecosystem includes a wide range of tools and frameworks that integrate with PostgreSQL and simplify database administration, development, and monitoring tasks. Some notable examples are:
    - **pgAdmin:** A popular graphical administration and development tool for PostgreSQL.
    - **psql:** The command-line interface for interacting with PostgreSQL databases.
    - **SQLAlchemy:** A Python SQL toolkit and Object-Relational Mapping (ORM) library that supports PostgreSQL and simplifies database operations.
    - **Django:** A popular Python web framework that has excellent support for PostgreSQL and provides high-level database abstractions.

# 4. Server [Administration](https://www.postgresql.org/docs/current/admin.html)

* PostgreSQL uses two primary configuration files: **postgresql.conf** and **pg\_hba.conf**.
  + the way clients connect
  + the role they are attempting to authenticate to
  + the database they are attempting to access
  + the client's IP address and network properties
* Users and user classes are defined within the system as roles.
* The methods of authenticating to a role are defined in the pg\_hba.conf file .
* The role's capabilities and level of access is defined by the **privileges** granted to them directly, through role membership, or through object ownership.
* Managing Databases
* **Backup and Restore:**
  + SQL dump
  + File system level backup
  + Continuous Archiving and Point-in-Time Recovery
* **Replication**



* + **Streaming Replication:**
    - Streaming replication allows a standby server to stay more up-to-date than is possible with file-based log shipping.
  + **Logical Replication:**
    - Logical Replication operates at the logical level rather than the physical level, allowing for more flexible and selective replication.
    - It enables replication of individual tables, columns, or even rows based on specific criteria.
  + **Bi-Directional Replication:**
    - This Replication is a configuration where two or more PostgreSQL servers act as both primary servers and standby servers simultaneously
  + **Cascading Replication**
    - The cascading replication feature allows a standby server to accept replication connections and stream WAL records to other standbys, acting as a relay. This can be used to reduce the number of direct connections to the primary servers.
* **Regression Tests**
  + used to verify that the existing functionality of the database remains intact after making changes or introducing new features.

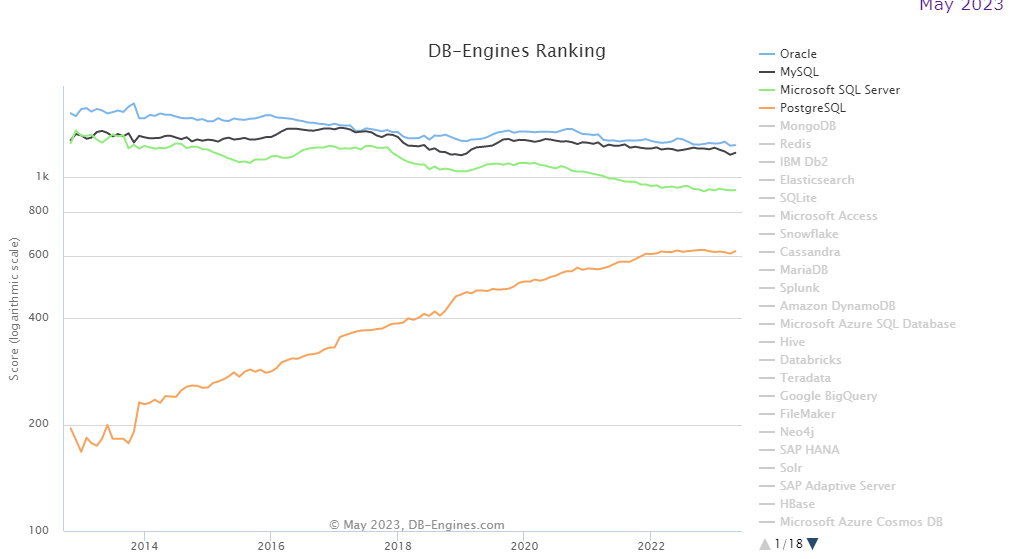
# [5. Server Programming](https://www.postgresql.org/docs/current/server-programming.html)

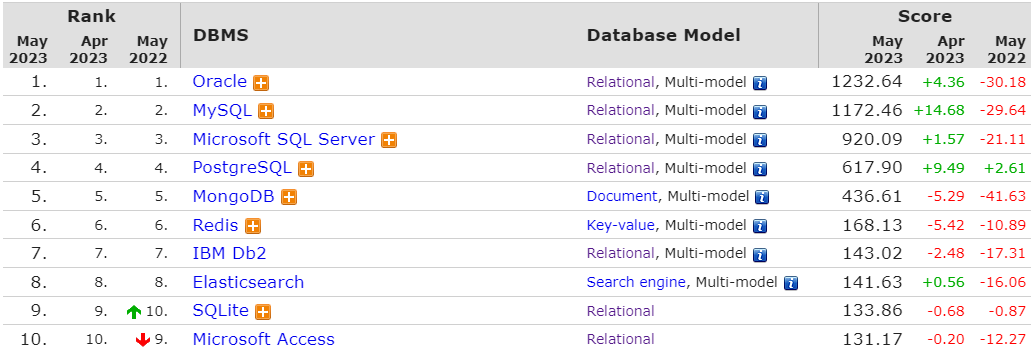
* **Triggers :**
  + we have main types of triggers:
* Row-level triggers
* Statement-level triggers.
  + The differences between the two kinds are how many times the trigger is invoked and at what time.
  + For example, if you issue an UPDATE statement that modifies 20 rows, the row-level trigger will be invoked 20 times, while the statement-level trigger will be invoked 1 time.
* **Trigger events**
  + Unlike regular triggers, which are attached to a single table and capture only DML events, event triggers are global to a particular database and are capable of capturing DDL events.

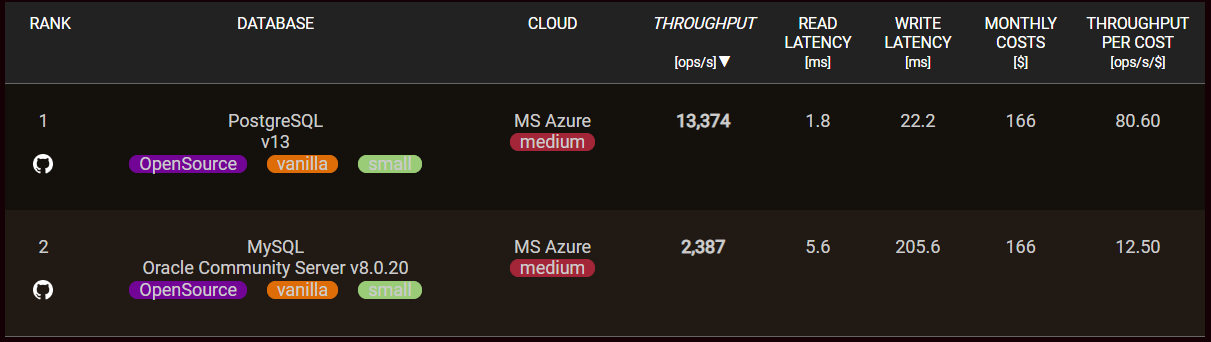
# 6. Concurrency Control

* Data consistency is maintained by using **MVCC ( Multi-version concurrency model ).**
* MVCC creates multiple versions of each row in the database, allowing each transaction to work with a snapshot of the data as it existed at the start of the transaction.
* This approach provides a high level of concurrency and eliminates the need for locking entire tables or blocking read operations.
* **Table- and row-level locking** facilities are also available in PostgreSQL
* **Transaction Isolation**
* Read Committed Isolation Level : SELECT only find target rows that were committed
* Repeatable Read Isolation Level : SELECT only sees data committed before the transaction began
* Serializable Isolation Level
* **Explicit Locking**
* **Table-Level Locks:**
  + Allow you to lock an entire table to prevent concurrent access by other transactions
  + Useful when you need to perform maintenance operations, such as schema changes or bulk data updates
* **Row-Level Locks**
  + allow you to lock specific rows within a table, uses a multi version concurrency control (MVCC) mechanism to manage row-level locks efficiently.
* **Page-Level Locks :**
* **Deadlocks :** automatically handle
* **Advisory Locks :**
  + allows applications to establish custom locks based on user-defined keys

**Database Ranking**





[](https://benchant.com/ranking/database-ranking)

[Difference](https://www.stitchdata.com/resources/postgresql-vs-mysql/)

|  |  |  |
| --- | --- | --- |
|  | PostgreSQL | Mysql |
| Architecture | Extensible object-relational database management system | Relational database management system; |
| Performance | High-performance in situations requiring high read and write speeds | High-performance in web-based applications and situations requiring high read speeds |
| Client Connection | * process-based Model * connection is associated with a dedicated operating system **process**. | * Thread-Based Model * connection is associated with a dedicated thread. |
| ACID compliance | ACID compliant | Only certain storage engines are ACID-compliant(InnoDB) |
| Programming language support | Written in C; supports C/C++, Python, JavaScript, R, Delphi, Java, Tcl, Go, Lisp, Erlang, and .Net | Written in C and C++; supports C/C++, R, Delphi, Java, Go, Lisp, Erlang, PHP, Perl, and Node.js |
| Replication | Asynchronous, cascade, and synchronous | Asynchronous, synchronous, and semisynchronous |
| Community support | Active, open source community with access to free resources | Vendor-backed community with access to free resources;  additional support offered with paid versions |