PostgreSQL

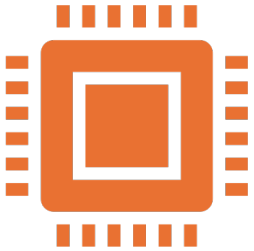
Advanced Postgres

Surendra Panpaliya

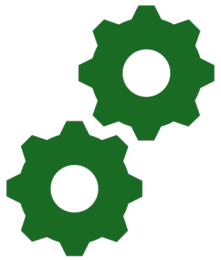
Agenda

- Introduction to PostgreSQL
- History and development of PostgreSQL
- Key features and benefits of PostgreSQL
- PostgreSQL vs. Oracle database systems

Agenda

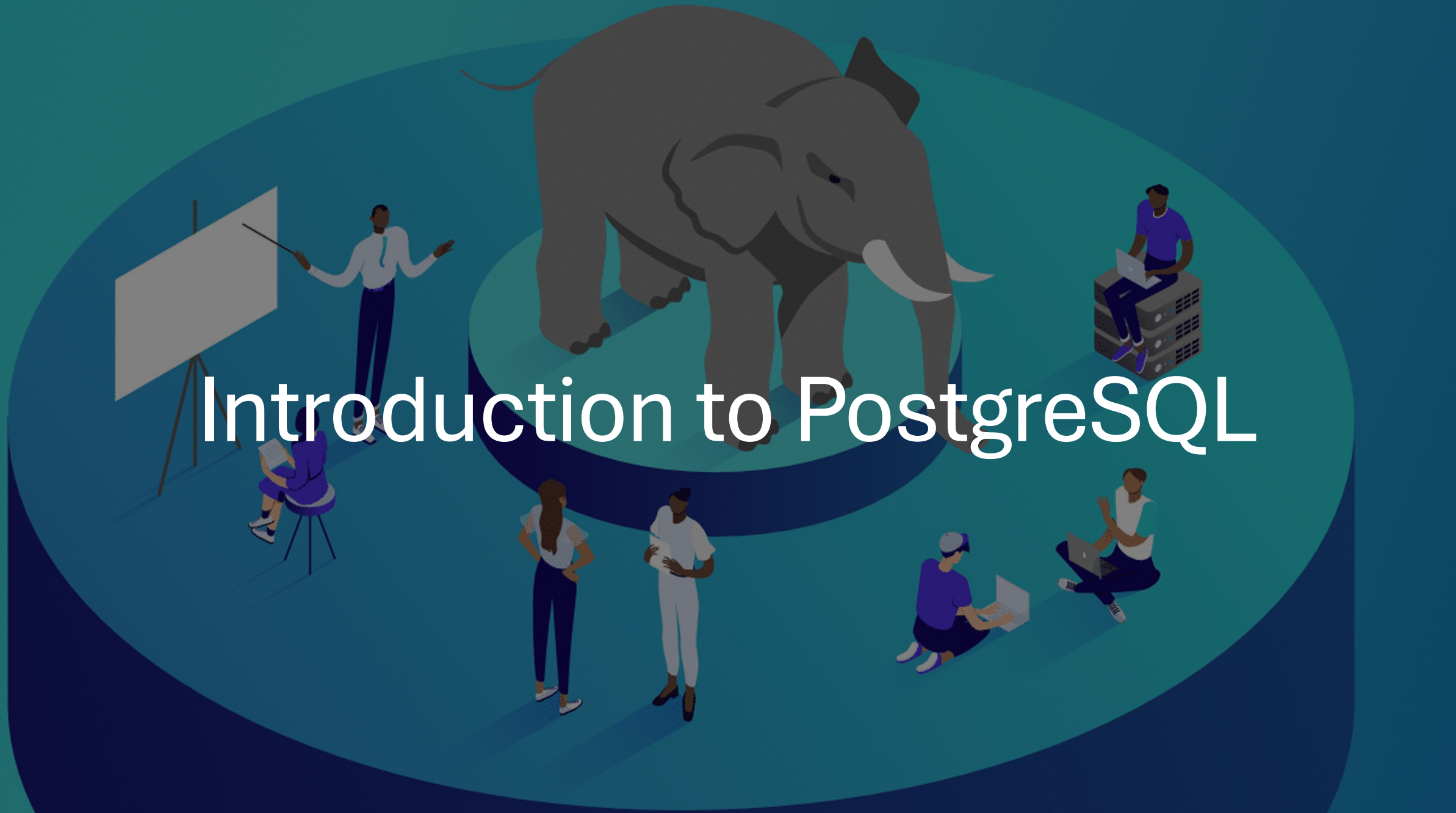


Installation (Windows, macOS, Linux)



Configuration basics

Connecting to the database



Introduction to PostgreSQL

What is PostgreSQL?

PostgreSQL (Postgres)

An advanced open-source

Relational Database Management System (RDBMS).

What is PostgreSQL?

- Known for
- Robustness
- Extensibility
- Standards compliance

What is PostgreSQL?

Designed to handle

Single-machine applications

Large-scale data warehousing

Web services

What is PostgreSQL?

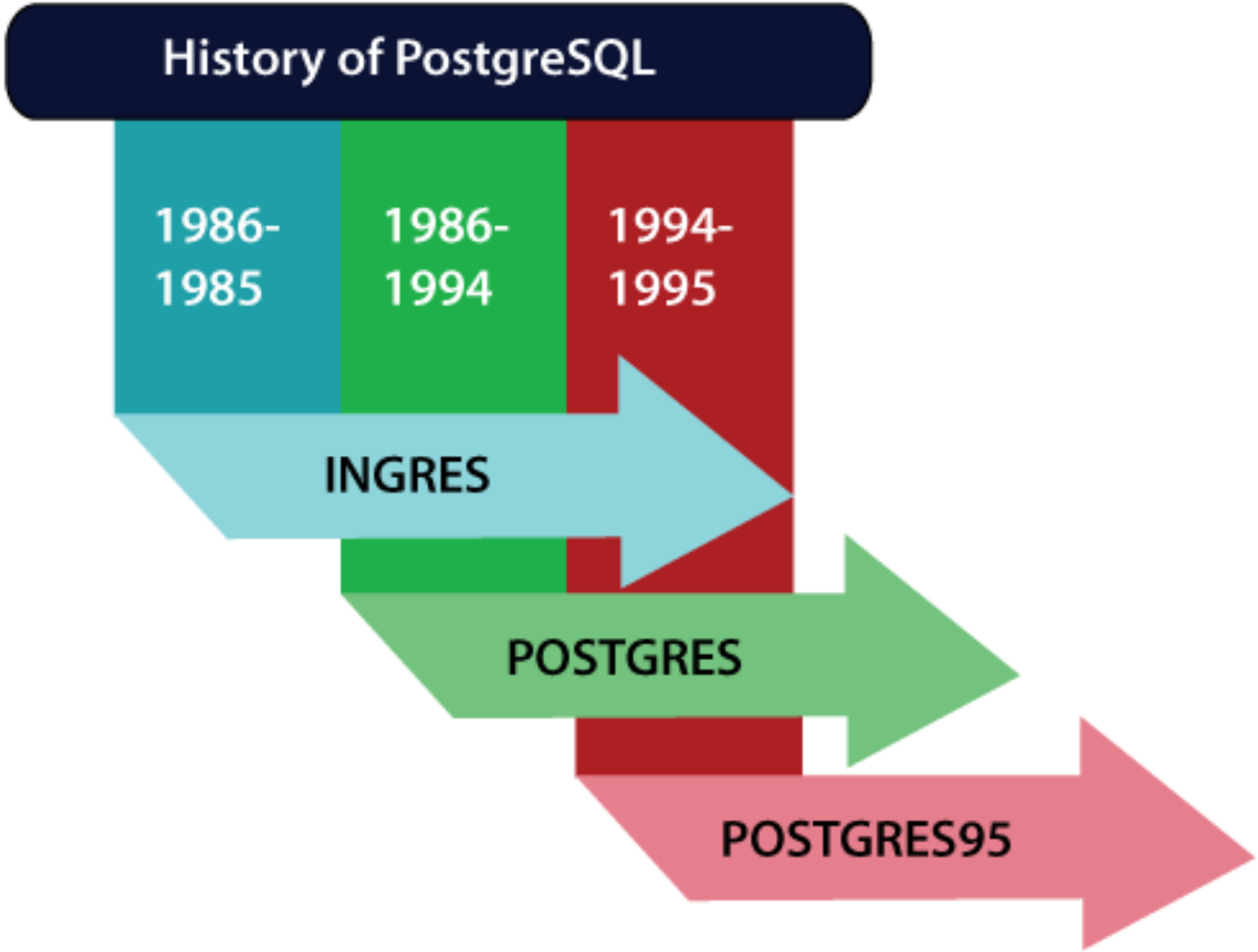
Used for

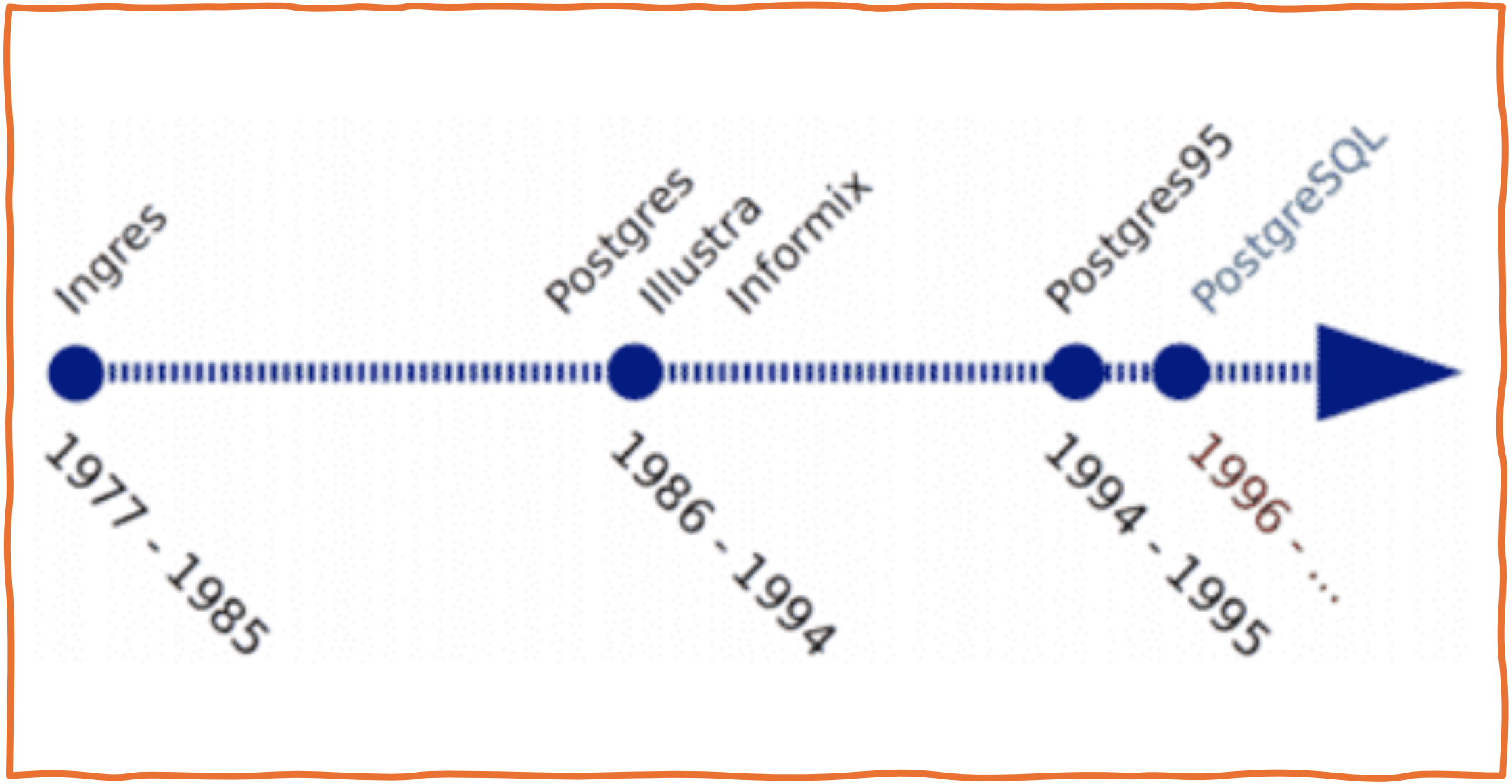
Web

Mobile

Geospatial

Analytics applications


History of PostgreSQL
1986-
1985
1986-
1994
1994-
1995
INGRES
POSTGRES
POSTGRES95


Ingres
1977 - 1985
Postgres
Illustra
Informix
1986 - 1994
Postgres95
1994 - 1995
PostgreSQL
1996 - ...

Origins of PostgreSQL

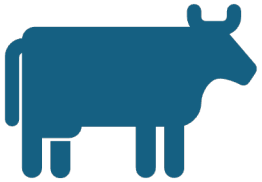
Ingres project

University of California, Berkeley

Professor Michael Stonebraker

Origins of PostgreSQL

Initiated in the 1970s



Laid the groundwork

Relational Database Systems

Postgres Project (1986-1995)

1986

Michael Stonebraker

Successor

To Ingres database

1986

Postgres = Post + Ingres

Designed to overcome

Limitations of Ingres system

1989

- Released first version of Postgres
- Introducing Object-Relational databases
- Allowing users to
- Define types
- Functions, and operators

1994

Postgres95, an enhanced version of Postgres

Replaced the PostQUEL query language with SQL

Standard database query language

Offered improved performance and usability

PostgreSQL (1996-Present)

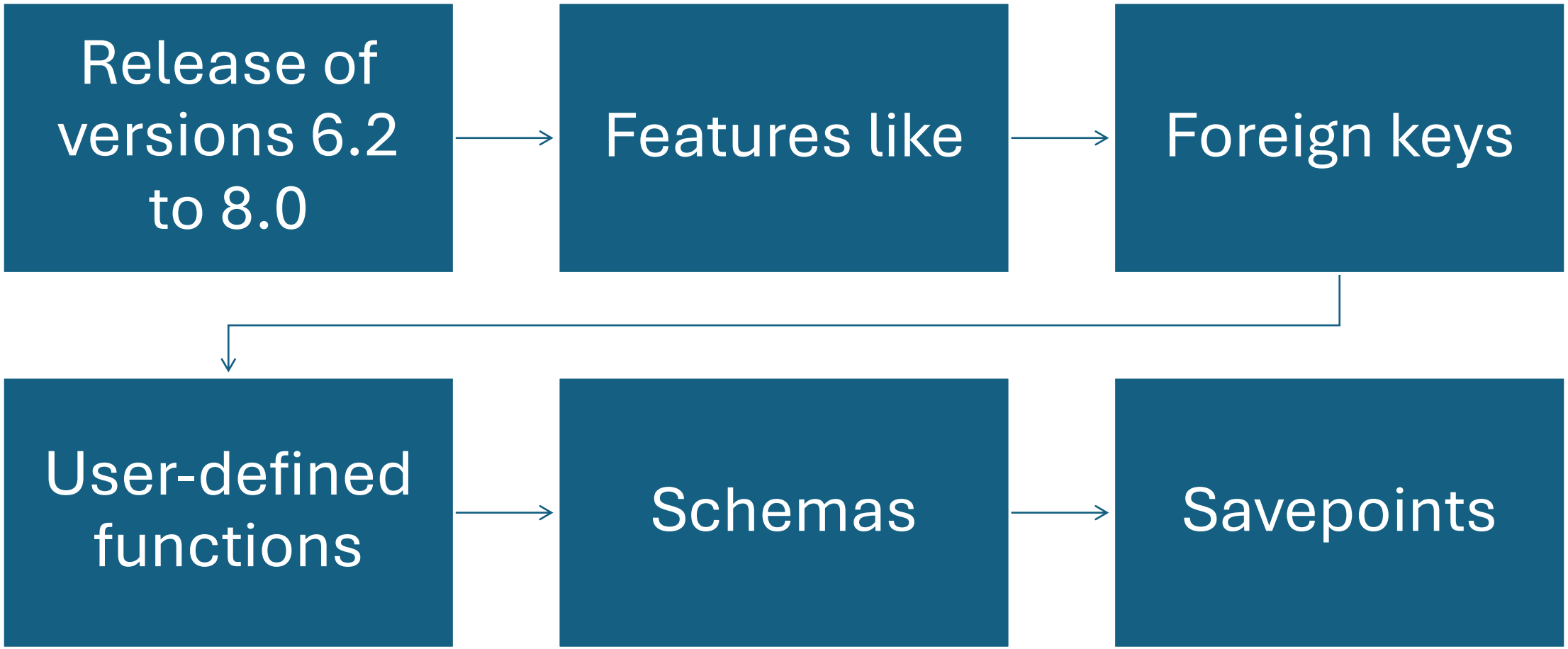
1996

- Postgres95 officially
- Renamed to PostgreSQL
- Reflect its SQL compliance
- First version PostgreSQL 6.0
- Released in January 1997

1997

- PostgreSQL 6.1 introduced
- Multi-Version Concurrency Control (MVCC)
- Major advancement
- Allows for high concurrency
- Without significant locking

1998-2005

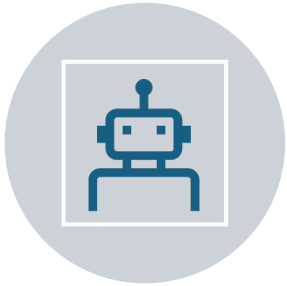

Release of versions 6.2 to 8.0
Features like
Foreign keys
User-defined functions
Schemas
Savepoints

1998-2005

Version 8.0

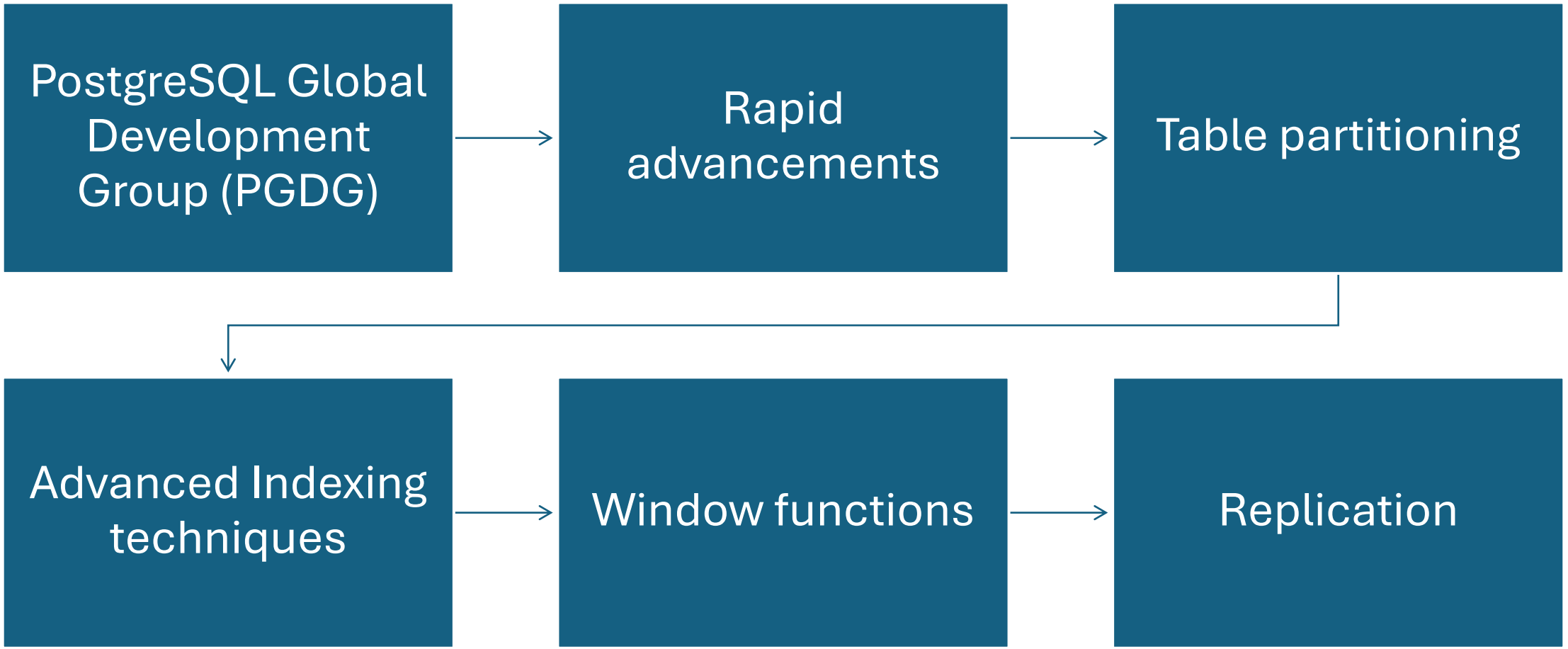
Released in 2005

Native Windows support



Expanding its usability

2006-2015


PostgreSQL Global Development Group (PGDG)
Rapid advancements
Table partitioning
Advanced Indexing techniques
Window functions
Replication

2006-2015

Version 9.0

Released in 2010

Streaming Replication

Hot standby

2016-2020

PostgreSQL 9.6

Introduced

Parallel Query execution

2016-2020

Version 10.0

Released in 2017

Logical Replication

Declarative Table Partitioning

2016-2020

PostgreSQL 11 to 12

Focused on

Performance improvements

Partitioning

SQL compliance

2021-Present

POSTGRESQL
13 AND 14

ENHANCED
PARTITIONING

QUERY
PERFORMANCE

SECURITY
FEATURES

2021-Present

PostgreSQL 15, 16

Continues to build on these improvements

Enhanced Parallelism

Compression methods

JSON features

PostgreSQL 16

VARIETY OF
NEW FEATURES

IMPROVE
PERFORMANCE

USABILITY

SECURITY

Performance Improvements

Query Performance Enhancements

Partitioning Improvements

Parallelism Enhancements

Usability Enhancements

JSON and JSONB Enhancements

Enhanced Window Functions

Improved psql Command-Line Tool

Security Enhancements

Enhanced Authentication and Authorization

Advanced Data Encryption

Administrative Features

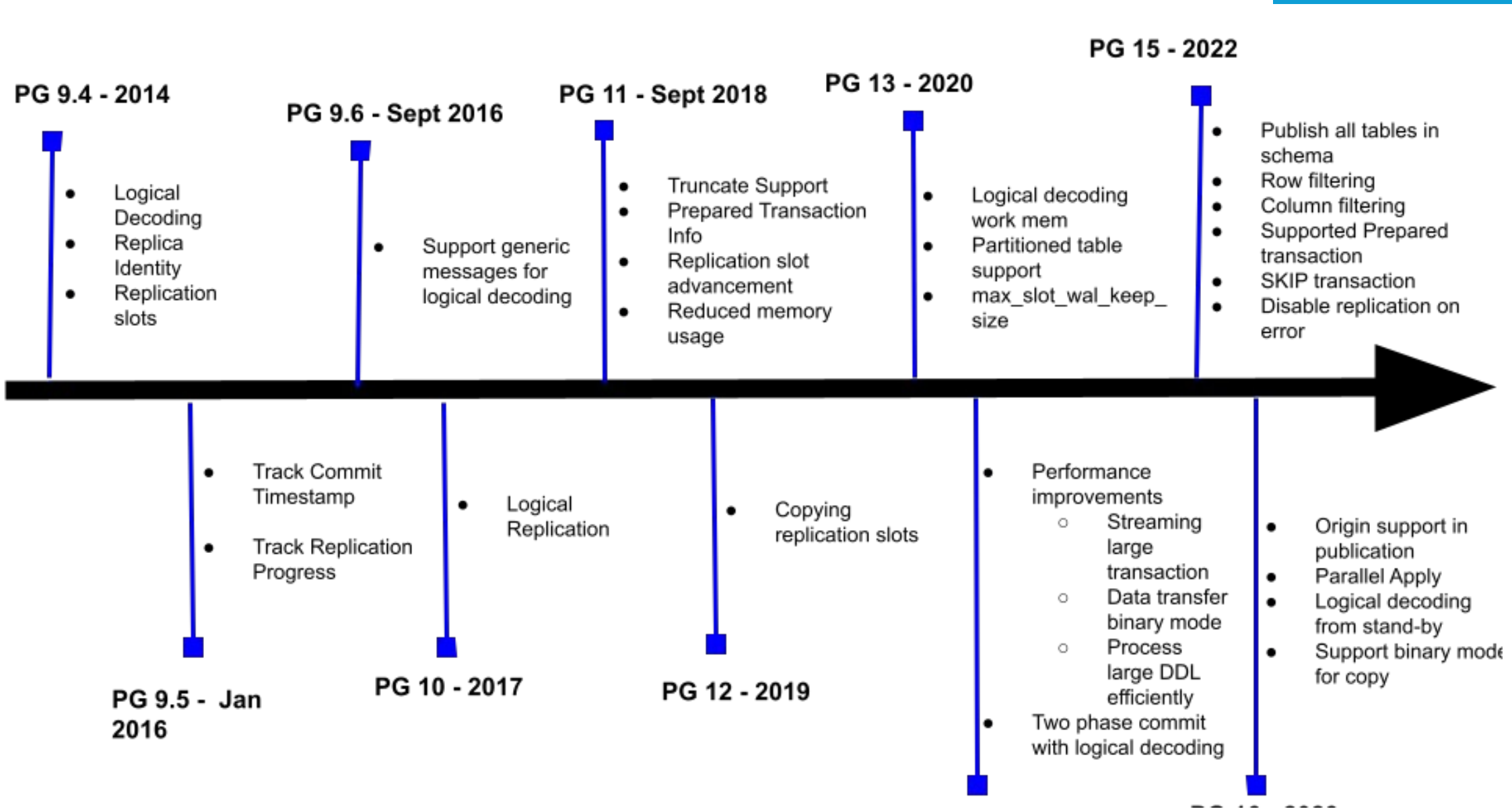
Enhanced Backup and Restore

Improved Monitoring and Logging

Developer Features

Enhanced PL/pgSQL

Advanced Indexing


PG 9.4 - 2014
Logical Decoding
Replica Identity
Replication slots
PG 9.6 - Sept 2016
Support generic messages for logical decoding
PG 11 - Sept 2018
Truncate Support
Prepared Transaction Info
Replication slot advancement
Reduced memory usage
PG 13 - 2020
Logical decoding work mem
Partitioned table support
max_slot_wal_keep_size
PG 15 - 2022
Publish all tables in schema
Row filtering
Column filtering
Supported Prepared transaction
SKIP transaction
Disable replication on error
PG 9.5 - Jan 2016
Track Commit Timestamp
Track Replication Progress
PG 10 - 2017
Logical Replication
PG 12 - 2019
Copying replication slots
Performance improvements
Streaming large transaction
Data transfer binary mode
Process large DDL efficiently
Two phase commit with logical decoding
Origin support in publication
Parallel Apply
Logical decoding from stand-by
Support binary mode for copy

Key Contributors

- **Michael Stonebraker**
- The primary Architect
- Original Postgres project
- Key figure in database research

PostgreSQL Global Development Group (PGDG)

A group of volunteers and Companies

Oversee the development

Maintenance of PostgreSQL

Community

Community-driven

Contributions from developers worldwide

Known for its collaborative spirit

Extensive documentation

Active mailing lists

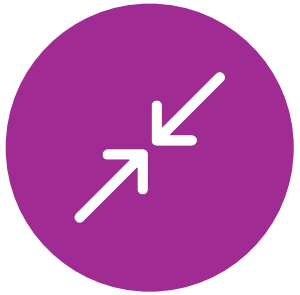
Design Philosophy

Standards Compliance

PostgreSQL adheres

To SQL standards

Ensuring compatibility



Ease of use

Extensibility

Designed to be highly extensible

Allowing users to define

New data types

Functions

Operators

Reliability and Stability

Focuses on delivering

Stable

Reliable database system

Suitable for

Mission-critical applications

Open Source

Released under the PostgreSQL License

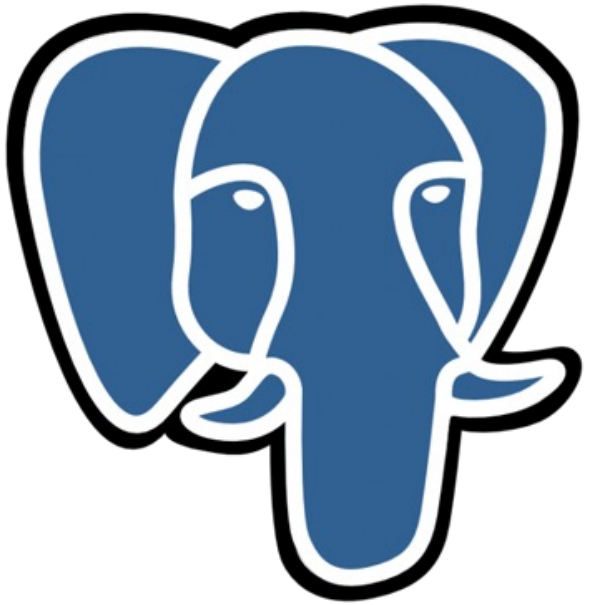
An open-source license

Allows for free use

Modification

Distribution

Major Milestones



PostgreSQL

MVCC (1997)

Multi-Version Concurrency Control

A crucial feature

for high Concurrency

Performance

SQL Compliance (1996)

Transition from

PostQUEL to SQL

Aligning with

Industry standards

Windows Support (2005)

Native support for

Windows

Broadening

PostgreSQL's user base

Streaming Replication (2010)

Hot standby

Improving

Data

Redundancy

Availability

Parallel Query Execution (2016)

Support for

Parallel Query Execution

Enhancing performance

for large queries.

Logical Replication (2017)

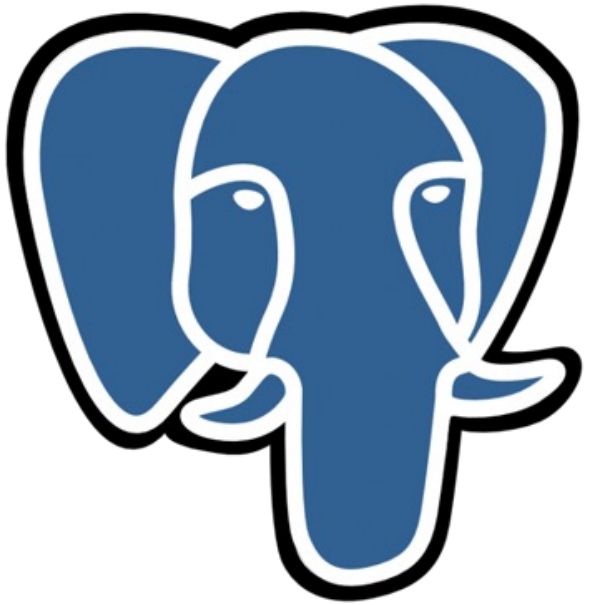
Introduction of

Logical replication,

Allowing selective

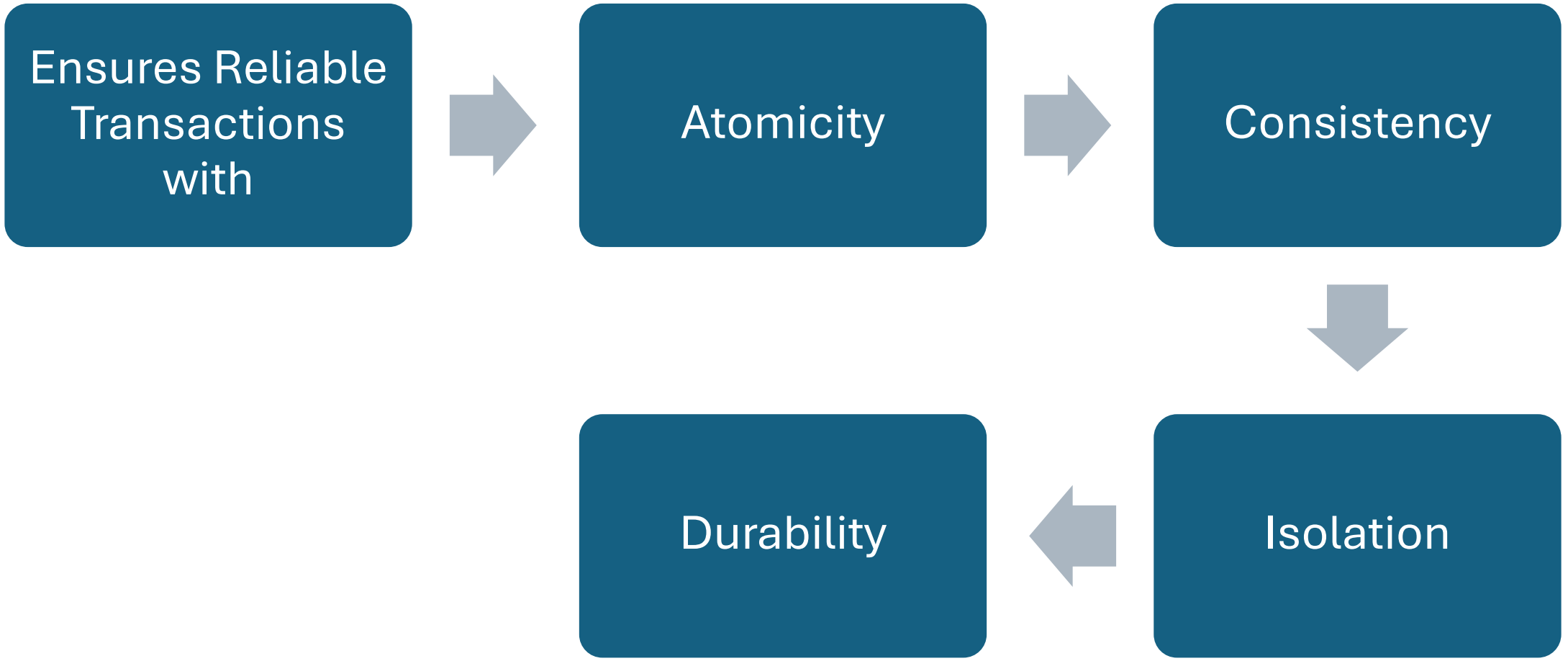
Replication of data

Key features and Benefits



PostgreSQL

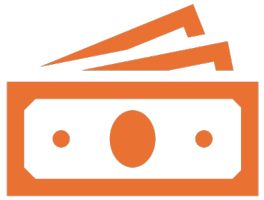
ACID Compliance



Atomicity

- Each transaction is treated
- Single ”Atomic" Unit
- Either completely succeeds or
- Completely fails.

Atomicity



If any part of the transaction fails

Entire transaction is rolled back

Database remains unchanged.

Example

If a transaction involves

transferring money from

Account A to Account B.

Example

Either both the debit from

Account A and

The credit to Account B

Happen or neither happens.

Example

If the debit succeeds

but the credit fails,

the transaction is rolled back.

Account A's balance is restored

to its original state

ACID Compliance

Provides robust support

For concurrent transactions

Without compromising data integrity

Consistency

Ensures that a transaction

From one valid state

To another valid state,

Maintain all predefined rules

Constraints, and triggers

Consistency

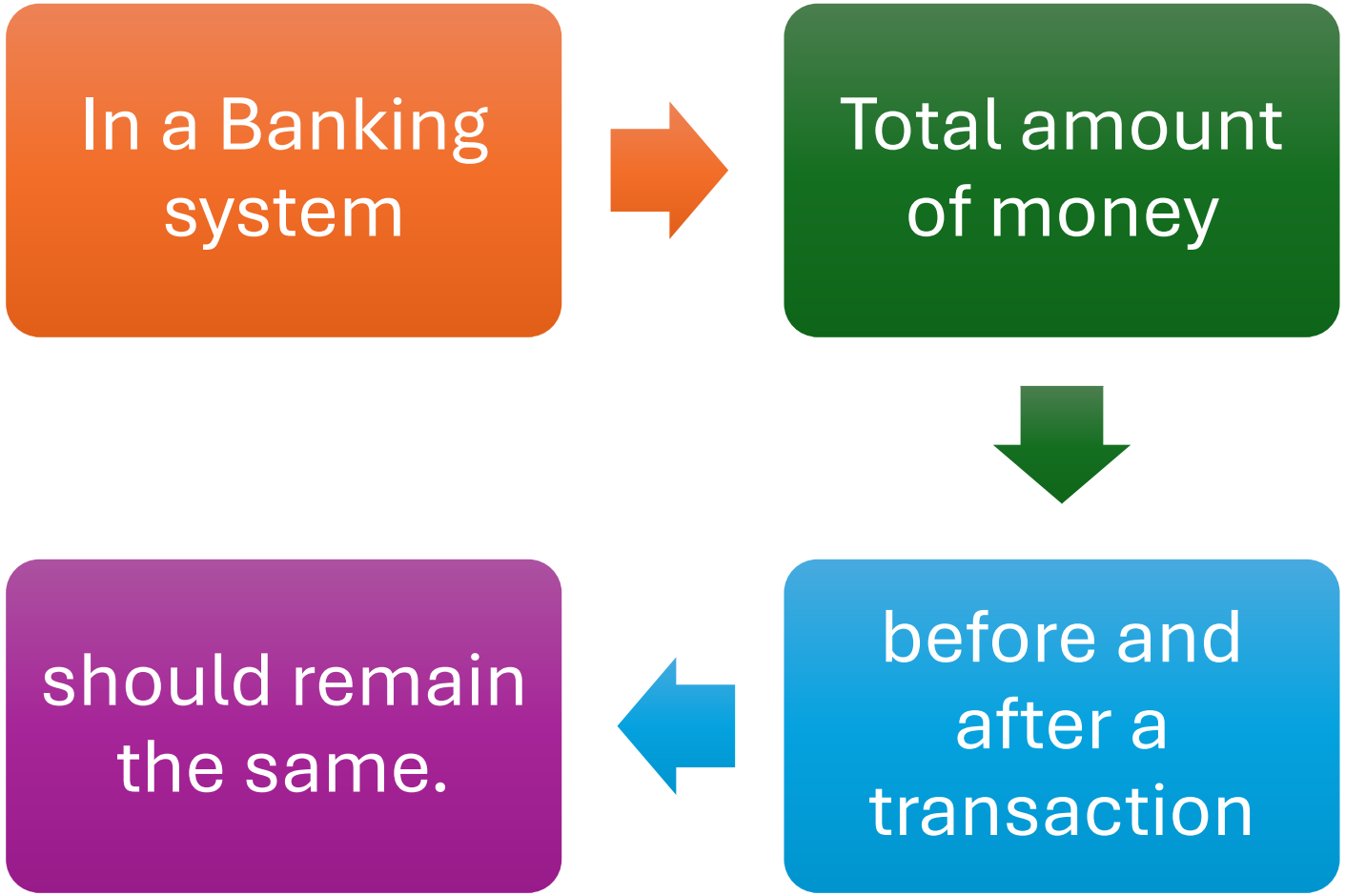
DATABASE MUST

REMAIN CONSISTENT

BEFORE AND AFTER

TRANSACTION

Consistency



Isolation

Transactions

Executed in

Isolation

From One another

Isolation

If two transactions

Running Concurrently

One transferring money from

Account A to Account B

Another from Account C to Account D

Isolation

Isolation ensures

Transactions

Do not see each other’s

Intermediate states

Durability

Once a transaction

Has been committed

Will remain

Even in the event of

a system failure

Durability

If a transaction to update

A bank balance is committed,

New balance is guaranteed

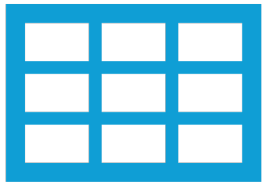
to be saved in the database.

Advanced Data Types

JSON/JSONB

Efficient storage

Querying of



JSON data.

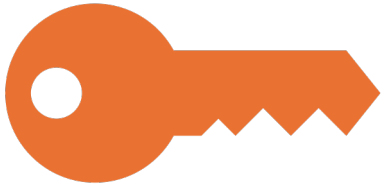
Arrays

Support for

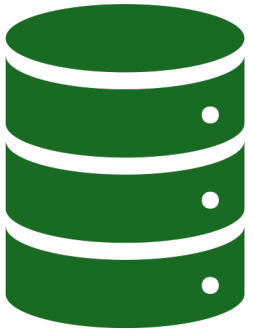
Multi-dimensional

Arrays

Hstore



Key-value pairs



Storage

Geometric Types

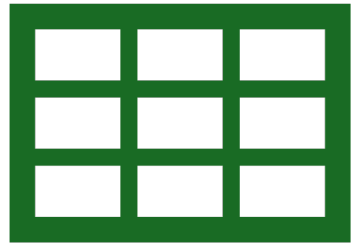
Points,

lines

polygons

Custom Types

Users can define



Own data types.

Extensibility

Users can create

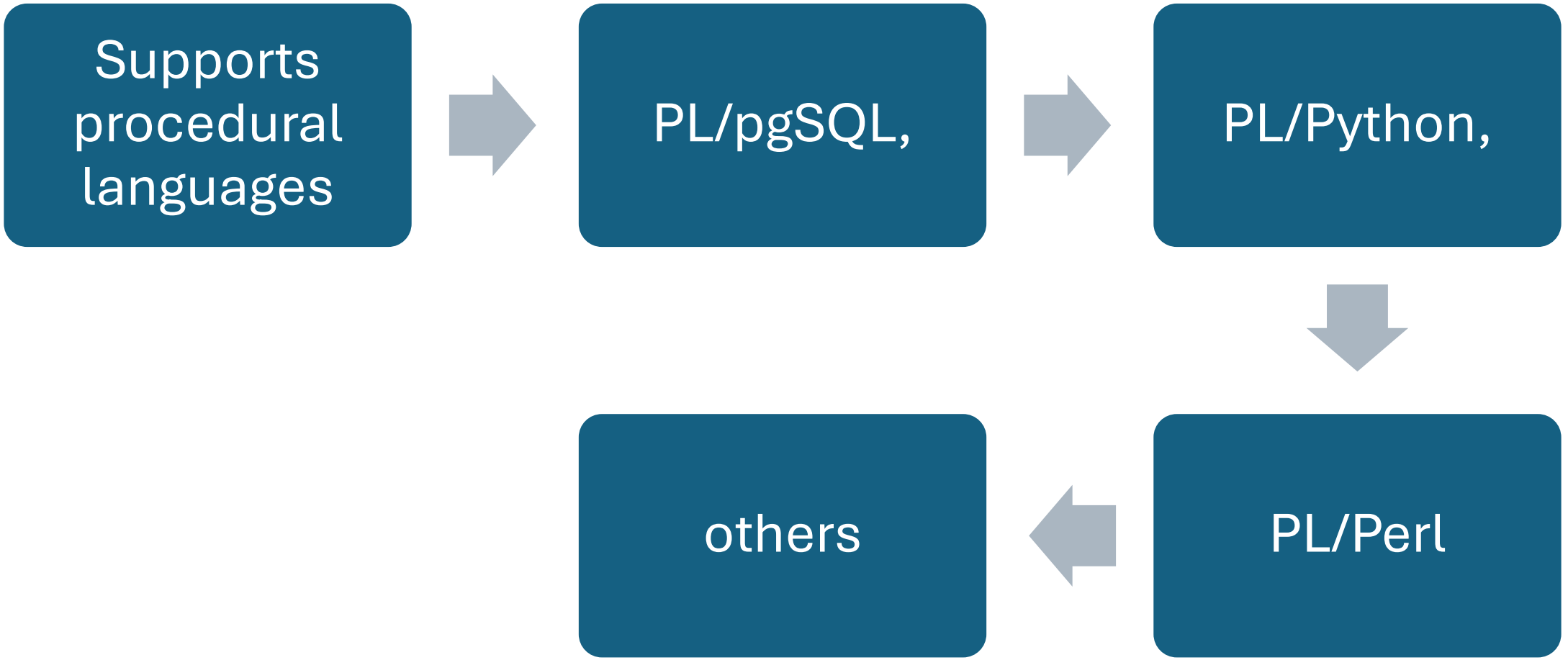
Custom functions

Operators,

Data types

index types.

Extensibility



Extensibility

Extensions like

PostGIS (for geospatial data)

Full-text search

Easily integrated

Multi-Version Concurrency Control (MVCC)

Allows multiple transactions

to occur simultaneously without locking.

Ensures data consistency

High performance

Indexing

Supports various index types

B-tree,

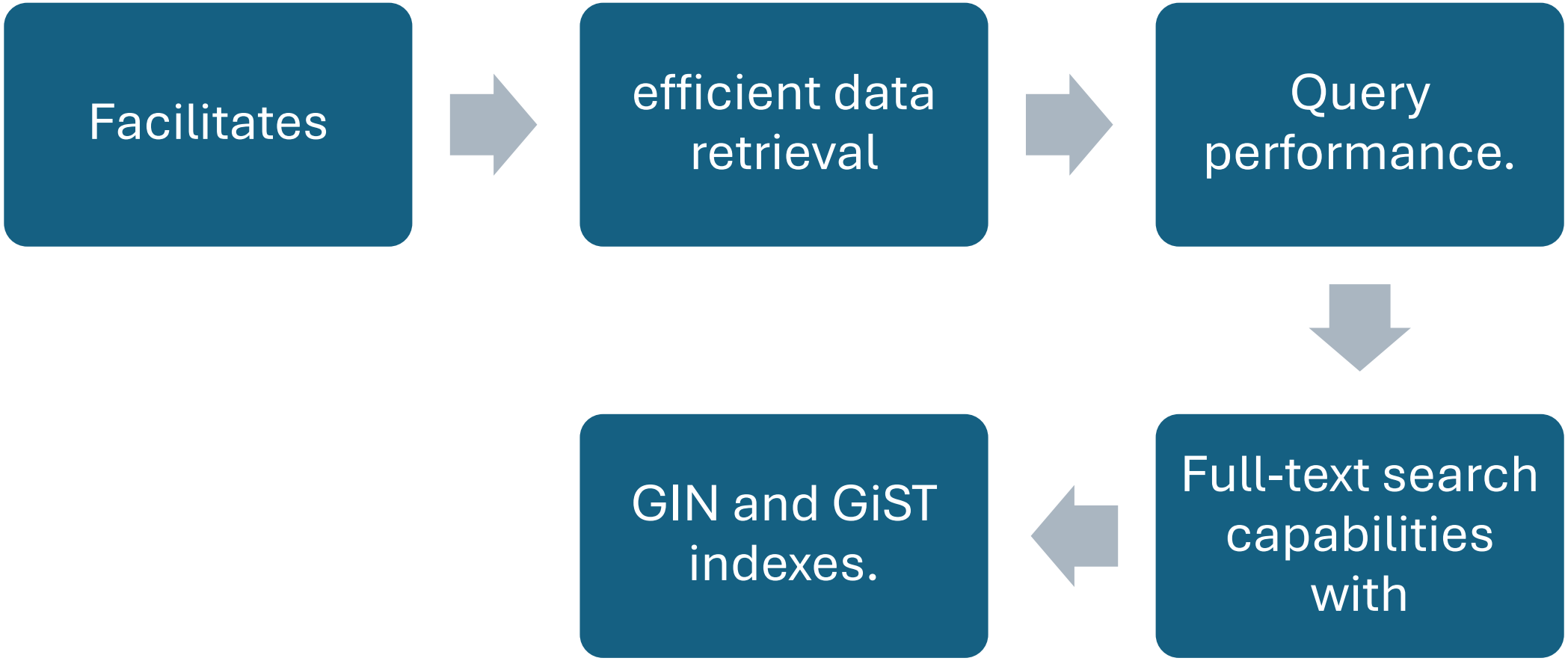
Hash,

GiST,

GIN

BRIN

Indexing


Facilitates
efficient data retrieval
Query performance.
Full-text search capabilities with
GIN and GiST indexes.

Foreign Data Wrappers (FDW)

Allows integration with

other databases and data sources.

Enables querying external data

as if it were a part of

the PostgreSQL database.

Full-Text Search

Built-in support for

full-text search capabilities.

Indexing and querying of

text data for fast search results

Replication and High Availability

- Supports both
- Synchronous
- Asynchronous
- Replication

Replication and High Availability

STREAMING REPLICATION

REAL-TIME DATA REDUNDANCY

LOGICAL REPLICATION

SELECTIVE DATA SYNCHRONIZATION

Partitioning

Declarative partitioning

for efficient management

of large tables.

Partitioning

Supports

Range,

List,

Hash

composite partitioning methods.

Security

Robust access control mechanisms

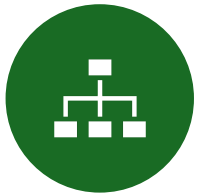
with roles and permissions.

SSL support for

secure client-server communication.

Security

Advanced features like



row-level security and

data encryption.

Performance Optimization

Parallel Query execution

for faster processing

of large datasets.

Performance Optimization

Query planner

Optimizer

for efficient

Query execution

Performance Optimization

CACHING MECHANISMS

TO IMPROVE PERFORMANCE.

Backup and Recovery

Comprehensive backup and

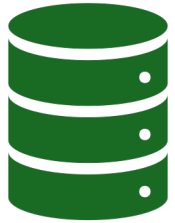
restore options with

tools like pg_dump and

pg_basebackup.

Backup and Recovery

Point-in-time recovery (PITR)



for restoring databases

to a specific state.

Benefits of PostgreSQL

Open Source and Free

PostgreSQL is open source,

allowing for free usage,

modification, and distribution.

No licensing costs,

making it cost-effective

for organizations

Flexibility

Highly extensible and

adaptable to various use cases.

Suitable for a wide range of applications,

from small-scale to

large-scale enterprise systems.

Community and Support

Strong, active community

providing extensive documentation,

support, and regular updates.

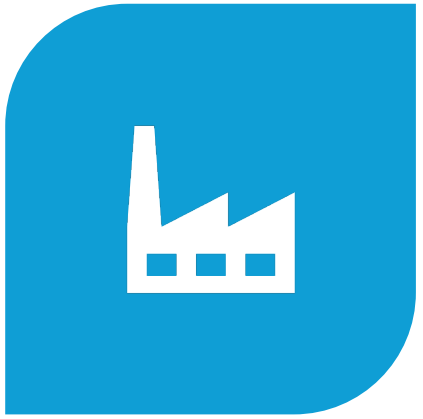
Numerous third-party tools and

extensions available.

Reliability and Stability

PROVEN TRACK RECORD OF

RELIABILITY AND STABILITY IN

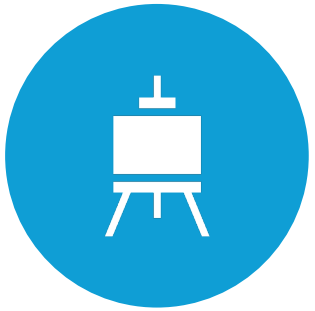


PRODUCTION ENVIRONMENTS.

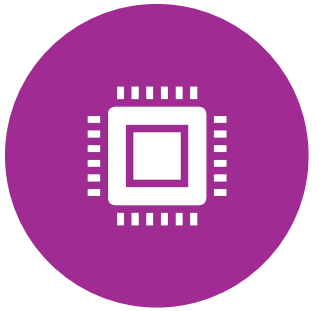
Standards Compliance

ADHERES TO SQL STANDARDS,

ENSURING COMPATIBILITY AND



EASE OF USE FOR DEVELOPERS



FAMILIAR WITH SQL.

Standards Compliance

REGULAR UPDATES

TO MAINTAIN COMPLIANCE

WITH EVOLVING STANDARDS.

Cross-Platform Compatibility

Runs on various operating systems

including Windows, macOS, and Linux.

Supports integration with

Various programming

languages and frameworks.

Performance

Advanced indexing and

query optimization techniques

ensure high performance.

Scalability to handle

large volumes of data and

high transaction loads.

Data Integrity

Strong support for data integrity

through constraints,

triggers, and foreign keys.

Ensures accurate and

consistent data storage and

retrieval.

Geospatial Capabilities

PostGIS extension

provides robust support

for geographic information systems (GIS).

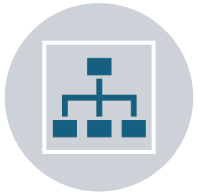
Ideal for applications

requiring spatial data management.

Enterprise Features

Comprehensive feature set

that meets the needs of



enterprise-level applications.

Support for complex queries,

transactions

data analytics.

Oracle vs PostgreSQL

Overview

Oracle

- A commercial, enterprise-level relational database management system.
- Developed by Oracle Corporation.
- Known for high performance, advanced features, and extensive support for complex, high-volume applications.
- Typically used in large-scale, mission-critical environments.

Overview

PostgreSQL

- An open-source relational database management system.
- Developed and maintained by a global community of developers.
- Known for its robustness, extensibility, and adherence to SQL standards.
- Suitable for a wide range of applications, from small projects to large enterprise systems.

Licensing and Cost

Oracle:

Proprietary software with licensing fees that can be quite high.

Offers various editions (Standard, Enterprise, etc.) with different feature sets and price points.

Licensing can include additional costs for support, features, and scalability options.

Licensing and Cost

PostgreSQL:

Open-source and free to use under the PostgreSQL License.

No licensing fees, making it cost-effective.

Community support is free; commercial support options are available but not mandatory.

Features

Oracle:

- Advanced features like Real Application Clusters (RAC), Data Guard, and Flashback Technology.
- Strong focus on security with features like Transparent Data Encryption (TDE) and Label Security.
- Comprehensive data warehousing and OLAP capabilities.
- Advanced indexing and partitioning options.
- Built-in support for PL/SQL for complex stored procedures and triggers.

Features

PostgreSQL

- Extensible with support for custom data types, operators, and functions.
- Advanced data types including JSON/JSONB, arrays, and hstore.
- Multi-Version Concurrency Control (MVCC) for high concurrency.
- Support for various procedural languages (PL/pgSQL, PL/Python, etc.).
- Extensions like PostGIS for geospatial data and full-text search.

Performance and Scalability

- **Oracle**
- Known for high performance and scalability in large, complex environments.
- Supports clustering, parallel execution, and advanced optimization techniques.
- Can handle very large databases and high transaction volumes with ease.
- In-memory database capabilities for performance enhancement.

Performance and Scalability

PostgreSQL:

- Highly performant and scalable, particularly with recent improvements in parallel query execution and indexing.
- Suitable for medium to large-scale applications.
- Support for partitioning, sharding, and replication for scalability.
- Performance can be further enhanced through tuning and optimization

Security

Oracle:

Extensive security features including Advanced Security Option (ASO), Virtual Private Database (VPD), and fine-grained auditing.

Robust user and role management with detailed access controls.

Security

PostgreSQL:

Strong security features including roles and permissions, SSL support, and row-level security.

Supports data encryption, though some advanced features may require additional configuration or extensions.

Support and Community

Oracle:

Comprehensive commercial support options from Oracle Corporation.

Extensive documentation, training, and certification programs.

Large user community, but primarily focused on enterprise customers.

Support and Community

PostgreSQL:

Strong community support with active mailing lists, forums, and extensive documentation.

Commercial support options available through various companies.

Large and diverse user base ranging from small startups to large enterprises.

Ease of Use and Management

- **Oracle:**
- Advanced management tools like Oracle Enterprise Manager for database administration.
- Can be complex to set up and manage, particularly for advanced features.

Ease of Use and Management

- **PostgreSQL:**
- User-friendly with tools like pgAdmin and a variety of third-party management tools.
- Generally easier to set up and manage, with a straightforward configuration.

Integration and Compatibility

- **Oracle:**
  - Integrates well with other Oracle products and many third-party applications.
  - Extensive support for various programming languages and platforms.

Integration and Compatibility

PostgreSQL:

- Highly compatible with many programming languages, frameworks, and platforms.
- Foreign Data Wrappers (FDWs) for integrating with other databases and data sources.

Conclusion

- **Oracle** and **PostgreSQL** are both powerful database management systems, but they cater to different needs and budgets

Conclusion

Oracle is ideal for large enterprises requiring high performance, advanced features, and comprehensive support for mission-critical applications, albeit with higher costs.

Conclusion

PostgreSQL offers a robust, cost-effective solution suitable for a wide range of applications, from small projects to large-scale enterprise systems, with strong community support and extensibility.

Oracle vs PostgreSQL Queries

Oracle

```
CREATE TABLE employees (  employee_id NUMBER GENERATED BY DEFAULT AS IDENTITY,  first_name VARCHAR2(50),  last_name VARCHAR2(50),  email VARCHAR2(100),  hire_date DATE,  salary NUMBER(8, 2),  PRIMARY KEY (employee_id));
```

PostgreSQL

```
CREATE TABLE employees (  employee_id SERIAL PRIMARY KEY,
first_name VARCHAR(50),  last_name VARCHAR(50),  email
VARCHAR(100),  hire_date DATE,  salary NUMERIC(8, 2));
```

Inserting Data

Oracle:

```
INSERT INTO employees (first_name, last_name, email, hire_date, salary) VALUES ('John', 'Doe', 'john.doe@example.com', TO_DATE('2023-07-01', 'YYYY-MM-DD'), 50000);
```

Inserting Data

PostgreSQL

```
INSERT INTO employees (first_name, last_name, email, hire_date, salary)
VALUES ('John', 'Doe', 'john.doe@example.com', '2023-07-01', 50000);
```

Selecting Data

Oracle:

SELECT employee_id, first_name, last_name, email FROM employees WHERE salary > 40000;

PostgreSQL:

SELECT employee_id, first_name, last_name, email FROM employees WHERE salary > 40000;

Updating Data

Oracle:

```
UPDATE employees SET salary = salary * 1.10 WHERE hire_date < TO_DATE('2020-01-01', 'YYYY-MM-DD’);
```

PostgreSQL:

```
UPDATE employees SET salary = salary * 1.10 WHERE hire_date < '2020-01-01';
```

Deleting Data

Oracle:

DELETE FROM employees WHERE last_name = 'Doe';

PostgreSQL:

- DELETE FROM employees WHERE last_name = 'Doe';

Functions and Procedural Languages

- ***String Functions***

Oracle:

SELECT CONCAT(first_name, ' ', last_name) AS full_name FROM employees;

PostgreSQL:

- SELECT first_name || ' ' || last_name AS full_name FROM employees;

Date Functions

Oracle:

SELECT SYSDATE FROM dual;

PostgreSQL:

SELECT CURRENT_DATE;

Conditional Logic

Oracle:

```
SELECT  employee_id,  first_name,  last_name,  CASE    WHEN
  salary > 50000 THEN 'High'   WHEN salary BETWEEN 30000 AND
  50000 THEN 'Medium'   ELSE 'Low'  END AS salary_levelFROM
  employees;
```

Conditional Logic

PostgreSQL:

- SELECT employee_id, first_name, last_name, CASE WHEN salary > 50000 THEN 'High' WHEN salary BETWEEN 30000 AND 50000 THEN 'Medium' ELSE 'Low' END AS salary_levelFROM employees;

PL/SQL vs PL/pgSQL

Oracle (PL/SQL):

```
CREATE OR REPLACE PROCEDURE raise_salary (emp_id NUMBER)
  ASBEGIN  UPDATE employees  SET salary = salary * 1.05  WHERE
  employee_id = emp_id;END;
```

PL/SQL vs PL/pgSQL

PostgreSQL (PL/pgSQL):

```
CREATE OR REPLACE FUNCTION raise_salary(emp_id INT) RETURNS VOID AS $$BEGIN  UPDATE
    employees  SET salary = salary * 1.05  WHERE employee_id = emp_id;END;$$ LANGUAGE plpgsql;
```

JSON Support

Oracle:

SELECT json_value(employee_data, '$.firstName') AS first_nameFROM employees_json;

PostgreSQL:

SELECT employee_data->>'firstName' AS first_nameFROM employees_json;

Recursive Queries

Oracle:

```
WITH employee_hierarchy AS (  SELECT employee_id,
 manager_id, first_name, last_name  FROM employees
 WHERE manager_id IS NULL  UNION ALL  SELECT
e.employee_id, e.manager_id, e.first_name,
e.last_name  FROM employees e
JOIN employee_hierarchy eh ON
e.manager_id = eh.employee_id)
SELECT * FROM employee_hierarchy;
```

Recursive Queries

PostgreSQL:

```
WITH RECURSIVE employee_hierarchy AS
(  SELECT employee_id, manager_id, first_name, last_name
FROM employees  WHERE manager_id IS NULL  UNION ALL
SELECT e.employee_id, e.manager_id, e.first_name, e.last_name
 FROM employees e  JOIN employee_hierarchy eh
ON e.manager_id = eh.employee_id)
SELECT * FROM employee_hierarchy;
```

Summary

Syntax Similarities:

Both Oracle and PostgreSQL share
SQL syntax for basic operations,
making it relatively easy
to switch between the two for common tasks.

Summary

Syntax Similarities: Both Oracle and PostgreSQL share SQL syntax for basic operations, making it relatively easy to switch between the two for common tasks.

Differences in Functions: Some functions and operators differ between the two systems, particularly with string concatenation and date functions.

Summary

Procedural Languages: Both databases support procedural languages, but the syntax and available features can vary.

JSON Support: PostgreSQL offers more straightforward syntax for JSON operations, while Oracle requires specific functions.

Licensing and Cost: PostgreSQL is open-source and free, whereas Oracle requires licensing fees, which can be significant.