MySQL

PostgreSQL

O PostgreSQL é um sistema gerenciador de banco de dados objeto-relacional (ORDBMS), open source, maduro e altamente confiável. Desenvolvido originalmente na Universidade da Califórnia, Berkeley, no projeto POSTGRES (iniciado em 1986 pelo pesquisador Michael Stonebraker), evoluiu para incluir suporte completo a SQL nos anos 1990, tornando-se o PostgreSQL que conhecemos hoje.

É mantido por uma comunidade global de desenvolvedores e é amplamente reconhecido como o banco de dados open source mais avançado do mercado, graças à sua fidelidade aos padrões SQL, robustez transacional e capacidade de extensão.

🔑 Principais Características

- Conformidade com padrões: Um dos SGBDs mais alinhados ao padrão SQL ANSI, garantindo portabilidade e boas práticas.
- Transações ACID completas: Ideal para sistemas que exigem integridade rigorosa dos dados, como financeiros, de saúde ou e-commerce.
- Tipos de dados avançados:
    - Suporte nativo a JSON/JSONB (com índices e consultas eficientes),
    - Arrays, hstore (pares chave-valor),
    - Geometria e geografia via extensão PostGIS (padrão em sistemas de mapeamento),

    - Tipos personalizados e domínios.
- Extensibilidade poderosa:
    - Crie suas próprias funções em PL/pgSQL, Python, JavaScript (PL/V8), R, entre outras.
    - Instale extensões oficiais ou da comunidade (ex: `uuid-ossp`, `pg_trgm`, `Citus` para escalabilidade horizontal).
- Alta disponibilidade e escalabilidade:
    - Replicação lógica e física nativa,
    - Particionamento de tabelas (por intervalo, lista ou hash),
    - Compatível com soluções como Patroni, pgBouncer e Citus (para clusters distribuídos).
- Segurança avançada:
    - Autenticação via SSL, SCRAM-SHA-256, LDAP, Kerberos,
    - Controle de acesso granular (RBAC),
    - Row-Level Security (RLS) para filtrar dados por usuário.

✅ Benefícios Práticos

- Custo zero de licenciamento, mesmo em ambientes corporativos.
- Desempenho consistente sob alta carga, com otimizador de consultas sofisticado.
- Fácil integração com frameworks modernos (Django, Rails, Spring, etc.) e ferramentas de BI (Metabase, Tableau, Power BI).
- Ideal para arquiteturas híbridas: combina o melhor do modelo relacional com flexibilidade de documentos (via JSONB).
- Excelente suporte a migrações, graças à compatibilidade com outros SGBDs e ferramentas como pg_dump, pg_restore e AWS DMS.

🌍 Casos de Uso Reais

- Shopify: processa milhões de transações diárias com PostgreSQL.
- Instagram: usa PostgreSQL como banco principal para metadados de usuários.
- GovTech e setor público: adotado por governos por ser open source, seguro e auditável.
- Startups e scale-ups: escolha natural por equilibrar custo, desempenho e recursos.

💡 Por que aprender PostgreSQL hoje?

Além de ser gratuito e poderoso, o PostgreSQL está em alta demanda no mercado de trabalho — especialmente em áreas como:

- Desenvolvimento backend,
- Engenharia de dados (Data Engineering),
- Análise de dados (Data Analytics),
- DevOps e SRE (com foco em observabilidade e automação de bancos).

Dominar PostgreSQL não só ensina SQL de verdade, mas também conceitos essenciais de arquitetura de dados, segurança e escalabilidade.

Diferença no código:

PostgreSQL	MySQL
`SERIAL` para auto incremento.	`AUTO_INCREMENT` para auto incremento.
`INSERT ... RETURNING id;` retorna valores direto.	`INSERT ...; SELECT LAST_INSERT_ID();` precisa de outra query.
Tipos extras: `JSON/JSONB`, `ARRAY`, `UUID`, `INET`, `HSTORE`.	Tem `JSON` (armazenado como texto), sem `ARRAY` nativo.
Índices: B-Tree, Hash, GIN, GiST, BRIN, SP-GiST.	Índices: B-Tree, Hash, Fulltext, R-Tree (engine-dependente).
`CHECK`, `EXCLUDE` constraints funcionam.	`CHECK` só funciona no MySQL 8+ (antes ignorado).
Expressões regulares: `~`, `~*`.	Expressões regulares: `REGEXP`.
`UPDATE ... RETURNING *;` devolve resultado atualizado.	`UPDATE ...;` não retorna nada.

Suporte a **Full Text Search** avançado (`to_tsvector`).	Full Text limitado (`FULLTEXT INDEX` em `TEXT`/`VARCHAR`).
Suporta **funções em várias linguagens** (PL/pgSQL, Python, C).	Funções/Procedures apenas em SQL.
Mais aderente ao **SQL ANSI padrão**.	Diverge em alguns pontos (ex.: `NULL`, `LIMIT`).
Transações com **MVCC robusto**.	Transações via InnoDB (menos flexível).

Comparação de um CRUD:

PostgreSQL	MySQL
CREATE `CREATE TABLE alunos (id SERIAL PRIMARY KEY, nome VARCHAR(100) NOT NULL, idade INT CHECK (idade >= 0));`	**CREATE** `CREATE TABLE alunos (id INT AUTO_INCREMENT PRIMARY KEY, nome VARCHAR(100) NOT NULL, idade INT);`
INSERT `INSERT INTO alunos (nome, idade) VALUES ('Maria', 20) RETURNING id;`	**INSERT** `INSERT INTO alunos (nome, idade) VALUES ('Maria', 20);SELECT LAST_INSERT_ID();`
READ `SELECT * FROM alunos WHERE idade > 18;`	**READ** `SELECT * FROM alunos WHERE idade > 18;`
UPDATE `UPDATE alunos SET idade = idade + 1 WHERE id = 1 RETURNING *;`	**UPDATE** `UPDATE alunos SET idade = idade + 1 WHERE id = 1;`
DELETE `DELETE FROM alunos WHERE id = 1 RETURNING *;`	**DELETE** `DELETE FROM alunos WHERE id = 1;`