Engineer

PostgreSQL

🐘 databasen som det mesta av det här arbetet byggdes på

Vad det är

PostgreSQL är en relationsdatabas med fri och öppen källkod. Den körs som en server, lagrar data i tabeller, besvarar SQL-frågor och gör varje transaktion odelbar: antingen genomförs hela transaktionen eller ingenting av den. De flesta som arbetar med den säger Postgres, och projektet räknar båda namnen som sina egna. Ställ frågan ”Vad är PostgreSQL?” till en erfaren utvecklare och svaret blir ofta kort: det är den allmänna databas som ett nytt system börjar på när ingen har ett särskilt skäl att välja en annan.

Den är äldre än det mesta av den mjukvara som i dag ligger ovanpå den. Projektet POSTGRES startade vid University of California i Berkeley under professor Michael Stonebraker, och implementationen inleddes 1986. År 1994 lade Andrew Yu och Jolly Chen till en SQL-tolk och släppte resultatet som Postgres95. Två år senare bytte det namn till PostgreSQL, och versionsnumren började om på 6.0 för att fortsätta den serie som Berkeley hade påbörjat. Projektet beskriver själv systemet som objektrelationellt: utöver tabeller och joinar kan en användare lägga till egna datatyper, funktioner, operatorer, aggregatfunktioner, indexmetoder och procedurspråk, och servern behandlar dem som sina egna.

Arkitekturen är klient och server. En serverprocess äger datafilerna, och varje klientanslutning får en egen process som forkas just för den anslutningen. Applikationer skickar SQL över anslutningen via en drivrutin för det språk de är skrivna i. Samtidigheten vilar på MVCC (multiversion concurrency control): en uppdatering skriver en ny version av raden i stället för att skriva över den gamla. Varje sats läser därför en konsekvent ögonblicksbild, läsare väntar aldrig på skrivare och skrivare väntar aldrig på läsare.

Var hamnar den bland alternativen? Bredvid MySQL och MariaDB som de andra relationsdatabaserna med öppen källkod, bredvid Oracle och Microsoft SQL Server som de kommersiella, bredvid SQLite som den inbäddade databasen som lever inne i ett enda program, och mot dokumentdatabaser som MongoDB, som lagrar poster utan ett fast schema.

Använd begreppet i ett beslut

Använd ”Vad är PostgreSQL?” i ett konkret beslut. Vi kan knyta begreppet till systemet ni bygger eller köper. Hittar databasfrågorna och de saknade indexen bakom ett system som har blivit tyngre för varje år, och åtgärdar dem som spelar roll.

Se våra tjänster Hör av er →

Granska antagandena bakom ordet

Använd förklaringen och det länkade arbetet för att formulera frågor till projektet. Vi kan undersöka antagandena tillsammans innan ett verktyg väljs. 30 dagars eldata på 180 ms — 180 ms för att aggregera en månads mätaravläsningar

30 dagars eldata på 180 ms Hör av er →

Varför det spelar roll

Valet av databas är ett beslut som ett team lever med i åratal, eftersom datan överlever koden som skrivs mot den. PostgreSQL är det förstaval man sällan ångrar, av tre skäl som en köpare kan kontrollera själv.

Det första är kostnad och inlåsning. PostgreSQL släpps under PostgreSQL License, en kort och tillåtande licens i samma familj som BSD och MIT. Det finns alltså ingen avgift per processorkärna, ingen licensrevision och ingen leverantör som kan ändra villkoren. Amazon, Microsoft och Google säljer den alla tre som hanterad tjänst, i AWS, Azure och Google Cloud, och samma databas körs på en bärbar dator. Att byta driftmiljö blir därför en flytt av data snarare än en omskrivning av applikationen.

Det andra är bredden. En och samma server täcker relationstabeller, JSON-dokument i `jsonb` -kolumner, fritextsökning, geospatiala databasfrågor genom tillägget PostGIS och schemalagda jobb genom `pg_cron` . Varje sådan del är en separat produkt som teamet slipper driva, säkra, ta säkerhetskopior av och rekrytera kompetens för. För de flesta produkter avgörs valet mellan PostgreSQL och MySQL just här: för vanliga läsningar och skrivningar fungerar båda bra, och skillnaden syns när arbetet kräver rikare typer, striktare villkor eller ett tillägg.

Det tredje är livslängden. Projektet släpper en huvudversion om året och stöder var och en i fem år, så en uppgradering kan planeras i stället för att tvingas fram av en nödsituation. PostgreSQL 18, som släpptes i september 2025, lade till

asynkron I/O och en inbyggd funktion för tidsordnade UUID, och SQL som skrevs för versioner tio år äldre fungerar i regel oförändrad på den.

Avvägningarna är verkliga. En process per anslutning gör anslutningar dyra, så en applikation med hög belastning behöver en anslutningspool. PostgreSQL skalar lättare upp, på en större maskin, än ut över flera. Den som behöver skalbarhet för skrivningar över många servrar får ta till partitionering, replikering eller ett tillägg, och vart och ett lägger till driftbörda. Och standardinställningarna är anpassade för en liten maskin, så en seriös installation behöver en genomtänkt konfiguration.

Så ser det ut i praktiken

Många av projekten på den här webbplatsen körs på PostgreSQL, och tre av dem visar tre olika sätt att luta sig mot den.

På en plattform blev databasen hela dataskiktet. Varje läsning och skrivning går genom någon av 1 275 lagrade funktioner i 34 scheman, och API:t, skrivet i Go, har ingen direkt åtkomst till någon tabell. Rollen som API:t loggar in som, `mariner`, har EXECUTE på funktionerna och USAGE på schemana och ingenting annat, ungefär 4 000 explicita behörigheter. Funktionerna körs som SECURITY DEFINER under en separat ägarroll utan inloggning och med en fast `search_path`. Åtkomstgränsen upprätthålls av själva Postgres. Priset är att ett nytt fält innebär en databasmigrering och en ändring i en funktion.

I en applikation med tenants, platser och dagliga mätvärden för el var uppgiften trimning snarare än design, och målet var 10 gånger snabbare utan att arkitekturen gjordes om. GiST-index för de spatiala data, sammansatta index på tenant-ID och datumintervall, intervallpartitionering av tabellen med eldata per dag och månad och SQL-frågor som skrevs om efter att deras planer lästs med `EXPLAIN ANALYZE` gjorde svarstiderna för databasfrågor 10 gånger snabbare och minskade resursanvändningen med 40 %.

Och på en plattform med långsamt bakgrundsarbete ligger jobbkön i samma databas. River håller sin kö i PostgreSQL, så ett jobb och de rader det rör committas eller rullas tillbaka tillsammans. Bakom den finns 15 worker-moduler, 8 schemalagda uppgifter och 20 `pg_cron`-jobb som rensar partitioner, roterar salt och uppdaterar aggregat. Sidan säger gränsen rakt ut: vid en viss volym slutar en kö i Postgres att vara rätt svar, och en rörlig del mindre var värd mer än marginal som ingen skulle använda.

Var det går fel

PostgreSQL tappar sällan data. Däremot överraskar den folk, och överraskningarna har några välbekanta former.

Tabellen växer medan antalet rader står still. En `UPDATE` eller `DELETE` lämnar kvar den gamla radversionen tills `VACUUM` tar bort den, eftersom en annan transaktion fortfarande kan läsa den. En transaktion som står öppen i timmar håller tillbaka den städningen för hela databasen, och standardinställningarna för autovacuum passar små tabeller. Stäng inaktiva transaktioner, trimma autovacuum per tabell för de stora och lägg data som löper ut i partitioner, så att en gammal månad kan tas bort i ett svep i stället för rad för rad. Vacuum gör också ett andra jobb. Transaktions-ID:n är 32 bitar, varje tabell måste gås igenom av vacuum minst en gång per två miljarder transaktioner, och en server som hamnar för långt efter vägrar nya skrivningar för att skydda datan.

Minnet tar slut vid första trafiktoppen. Varje anslutning är en serverprocess med eget minne, och standardgränsen är 100 anslutningar. Tjugo applikationsinstanser med en pool på femtio var ber om tusen processer. Sätt en anslutningspool som PgBouncer framför databasen och dimensionera varje applikations pool efter arbetet, vilket oftast blir en handfull anslutningar per instans.

En databasfråga som tog millisekunder börjar ta sekunder. Planeraren valde en annan plan när datan växte, ofta för att statistiken blev föråldrad eller för att indexen byggdes för förra årets databasfrågor. Läs planen innan du ändrar något, uppdatera statistiken med `ANALYZE` och indexera det villkor som SQL-frågan faktiskt filtrerar på:

```
EXPLAIN (ANALYZE, BUFFERS)
SELECT id, total FROM orders
WHERE customer_id = 42 AND placed_at >= '2026-01-01';
```

Infogningar går långsammare när tabellen blir stor. Slumpmässiga UUID som primärnycklar hamnar överallt i B-trädindexet, så varje infogning rör en ny sida och indexet fragmenteras. Tidsordnade identifierare undviker det: en plattform gjorde UUID v7 till standardnyckel genom `uuidv7()` i PostgreSQL 18, och nya rader sorteras in i indexet i stället för att strös ut över det.

Läs mer

- What Is PostgreSQL? — projektets egen definition, med de egenskaper och tilläggspunkter den räknar upp.
- A Brief History of PostgreSQL — Berkeley, Postgres95 och namnbytet, berättat av dem som var med.
- Concurrency Control: Introduction — hur ögonblicksbilder låter läsningar och skrivningar pågå utan att blockera varandra.
- Routine Vacuuming — döda radversioner, autovacuum och omslag av transaktions-ID (wraparound) i sin helhet.
- Using EXPLAIN — hur du läser en exekveringsplan, med genomarbetade exempel.
- Versioning Policy — den årliga huvudversionen och de fem års stöd som följer med varje.

Ordlista

Välj utifrån era faktiska förutsättningar

Ta med ett representativt dataset, frågorna det ska besvara och dagens flaskhals. Vi kan avtala mått på kvalitet, genomströmning och återställning.

Se vad det kostar Hör av er →

Gör begreppen praktiska

Fråga hur ”Vad är PostgreSQL?” gäller i er organisation. Ett konkret exempel från ert nuvarande system räcker för att börja samtalet.

Se våra tjänster

welcome@engineer.company +45 50 10 61 84 Köpenhamn, Danmark

Ta med dig den här sidan

Visitkort

Omsorgsfullt skapad med kärlek av D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Upphovsrätt © 2025 – nu · Engineer ApS · Alla rättigheter förbehållna.

Utskrivet från https://engineer.company/sv/glossary/postgresql/