
2024
数据风云
2021
数造未来
2022
分布式
2025
人工智能
2023
大数据
DTCC
第十七届中国数据库技术大会
DATABASE TECHNOLOGY CONFERENCE CHINA
融数聚智 创领未来
ITPUB168.com
ChinaUnix.net
ITPUB
北京·朗丽兹西山花园酒店 | 2026年8月20～22日

夯实数据基石
溯源 PostgreSQL 事务核心能力

肖玲峰

红石 PG 产品经理

我的经历

自 2011 年起从事 PostgreSQL 相关工作

从事过数据库和云服务方面的产品研发、解决方案和市场拓展工作

服务过全球范围的企业软件、运营商、金融保险、政企、医疗、传统制造等行业大客户

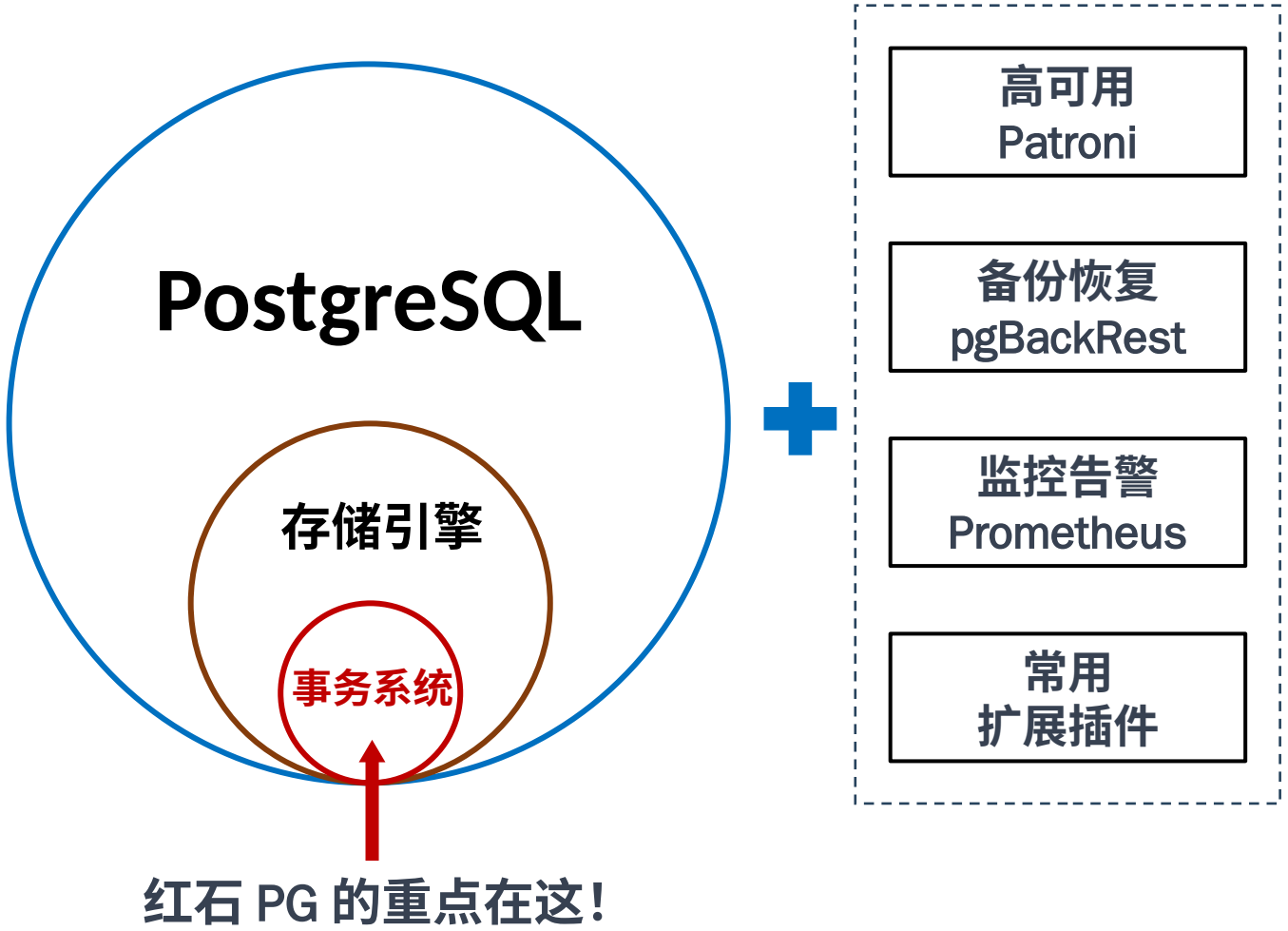
红石 PG 是做什么的?

聚焦 PostgreSQL 存储引擎的事务系统，专注 PostgreSQL 的质量和稳定性!

紧跟 PostgreSQL 开源社区主线发展，最新支持 PostgreSQL 18.6

整合 PostgreSQL 开源社区的技术和生态，提供开放的产品、解决方案和专业服务!

红石 PG
Redrock Postgres
PostgreSQL 企业级发行版



01

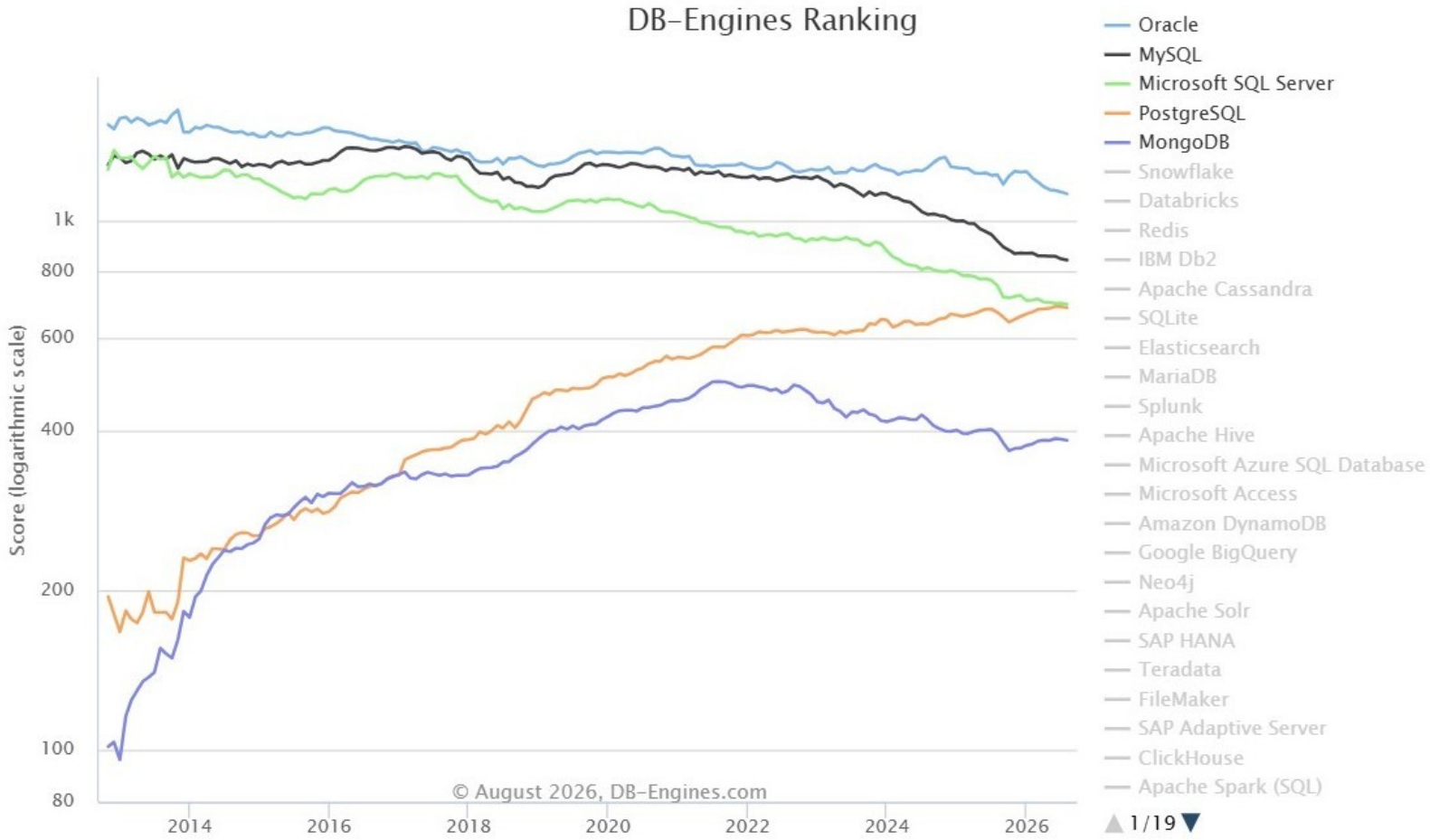
市场与社区： PostgreSQL 的黄金时代

头部数据库格局：三强下滑，一骑绝尘！

DB-Engines 流行度趋势稳步上升

三大主流数据库均呈现明显颓势

PostgreSQL 与第三名的差距已缩小至个位数，上升势头持续强劲！



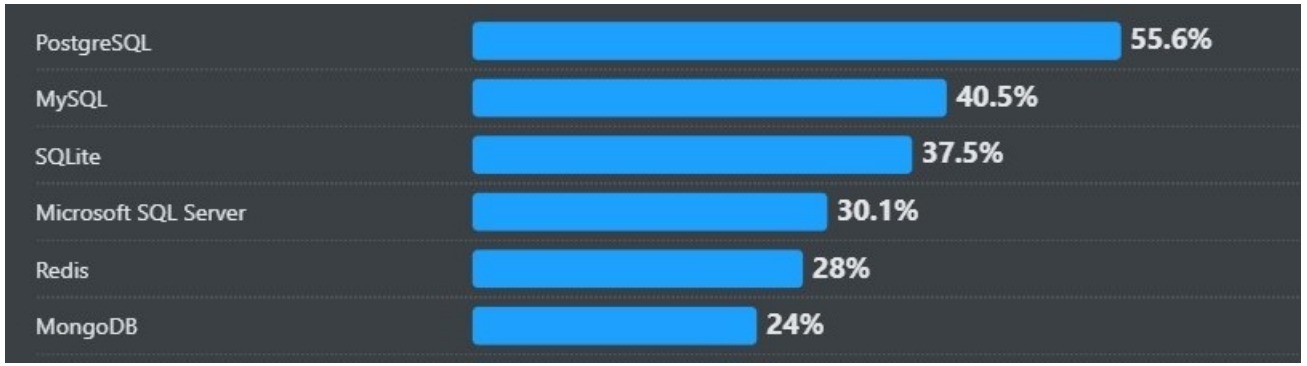
全球开发者的首选数据库

PostgreSQL 全栈式数据解决方案：替代种类繁多的专用数据库

需求	专用数据库	Postgres 特性能力 / 扩展
文档存储	MongoDB	JSONB 、 FerretDB 、 DocumentDB
向量数据 / AI 场景	Pinecone	pgvector 、 pgvectorscale
时序数据	InfluxDB	TimescaleDB 、 pg_partman
图数据库	Neo4j	SQL/PGQ 、递归 CTE 、 Apache AGE
全文搜索	Elasticsearch	tsvector 、 pg_trgm 、 ParadeDB
数据分析 / OLAP	Snowflake 、 ClickHouse	pg_duckdb 、 pg_clickhouse

开发者首选，采用率 55.6%

Stack Overflow 2025 年度调查显示， PostgreSQL 大幅领先 MySQL ，是当今全球开发者最青睐的数据库技术选型。



AI 时代底层的数据基石！

supabase

Agent 主流一站式后端基建

依托 PostgreSQL + pgvector 承载 Agent 记忆、向量检索、权限与存储，是 Vibe Coding 平台常用默认底座，面向独立开发者与初创团队，在 Agent 数据持久层占据重要地位。

Lovable

AI 时代的应用开发平台

互联网时代，经典的 LAMP 组合大放异彩。而 Lovable 与 Supabase / PostgreSQL 的结合，正在重新定义 AI 时代的全栈开发效率。

OpenAI

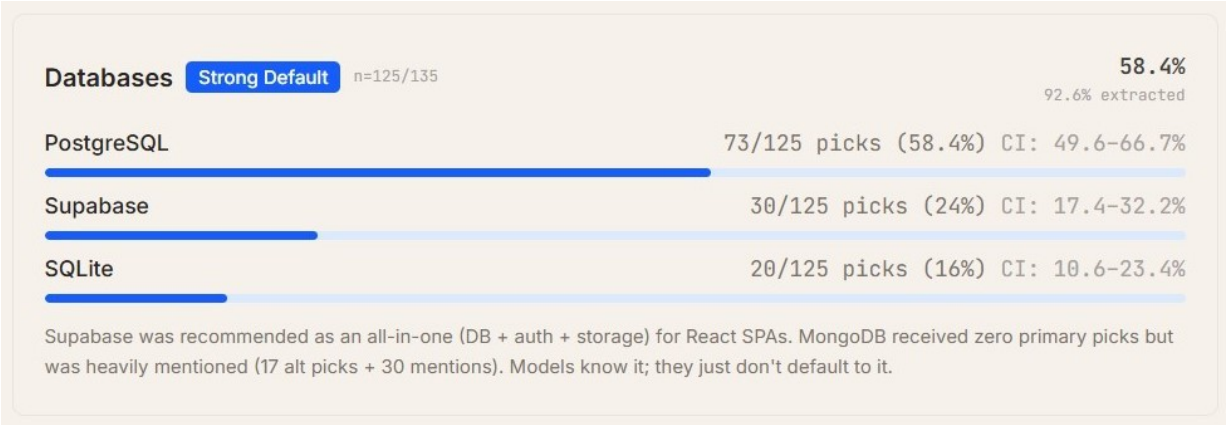
AI 时代的算力底座

深度支撑 ChatGPT 等大规模生成式 AI 服务，凭借对海量结构化数据的高效管理与高并发读写能力，成为人工智能时代最坚实的数据库基础设施。

Claude Code 强默认选择

PostgreSQL 以 58.4% 的选择率成为无可争议的“强默认”选项！

远超第二名 Supabase （24%）和第三名 SQLite （16%）



接下来讲什么?

PostgreSQL 在事务层面，有哪些核心挑战?

02

核心挑战一：空间膨胀与 VACUUM 风暴

深入解析高并发场景下数据库的存储黑洞与性能震荡根源

空间膨胀的根源： MVCC 的“仅追加”模式

Page			
xmin	xmax	200204	‘xxx’
…			

UPDATE

Page			
xmin	xmax	200204	‘xxx’
xmin	xmax	200205	‘xxx’
xmin	xmax	200205	‘yyy’
…			

图示：更新操作产生新行版本，旧版本被标记为“死亡”但暂不释放空间。

核心机制：“仅追加”的版本管理

PostgreSQL 的 UPDATE 并非原地覆盖，而是插入全新行版本并标记旧版本为“死亡”； DELETE 仅做逻辑标记，无物理删除。这种机制保障了高并发读写，但也埋下了空间膨胀的隐患。

直接后果：物理文件持续臃肿

随着更新 / 删除操作的频繁执行，表的物理大小会不断增长，即便逻辑数据量没有增加。这些未被清理的“死亡元组”会长期占用磁盘空间，拖慢全表扫描与 IO 性能。

实测警示：一次全表更新引发的膨胀

某业务表初始大小仅 35MB ，执行两次全表更新后，物理体积飙升至 104MB （增长近 2 倍）。若缺乏定期维护，膨胀会持续累积，严重影响数据库稳定性。

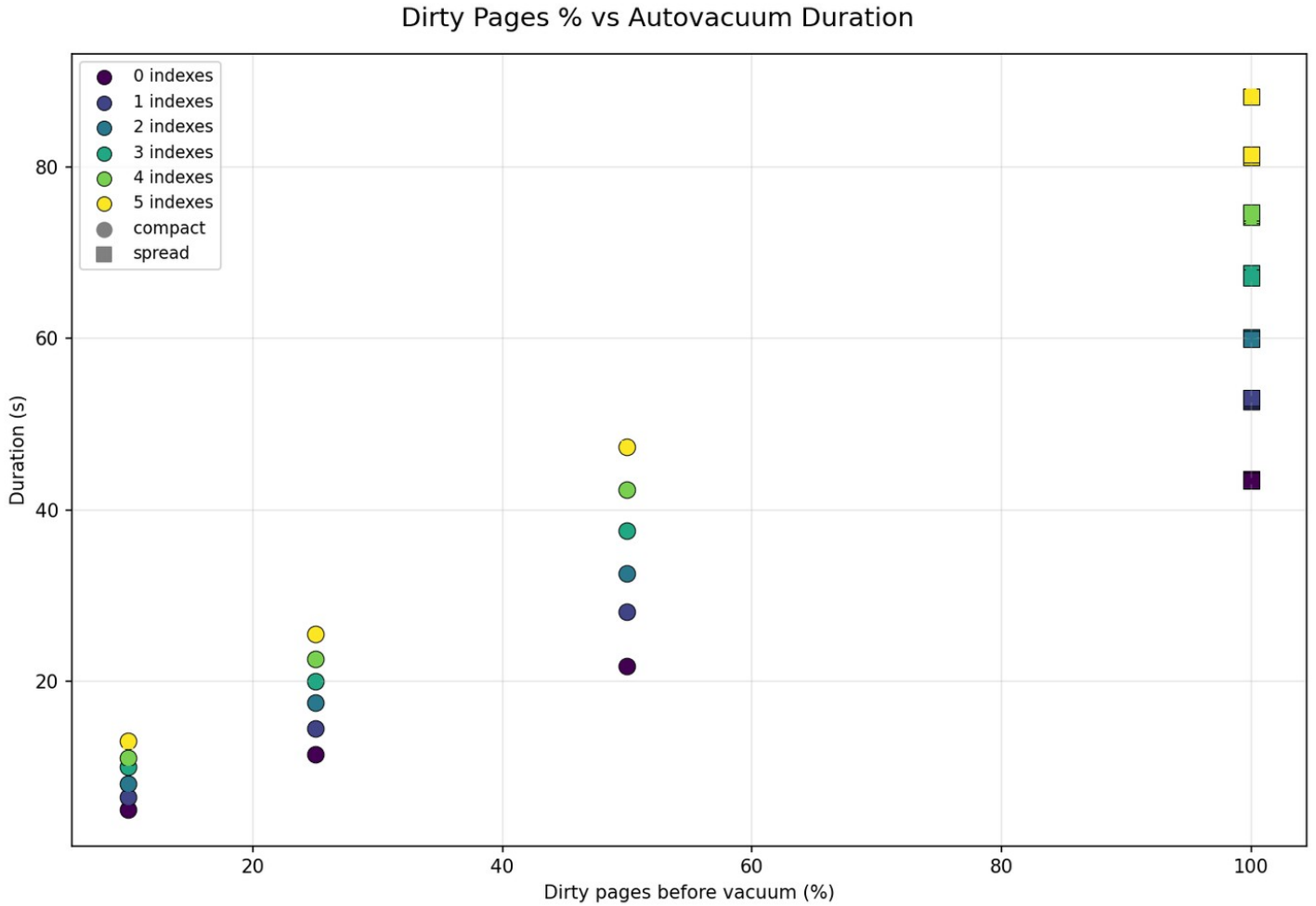
Autovacuum 的工作量

当表的死亡元组数量超过阈值时，就会触发自动清理：

pg_stat_all_tables.n_dead_tup >
autovacuum_vacuum_threshold +
autovacuum_vacuum_scale_factor *
pg_class.reltuples

测试结论

脏页数量是自动清理的主要代价驱动因素，其次是索引数量。相同数量的死亡元组，分布方式不同可能带来截然不同的自动清理开销。



注：图表数据来源于 Percona

Autovacuum 运维痛点

01 参数调优如同“黑魔法”

现有的阈值和系数计算方式非常粗糙，难以适用极端过大或过小的表。默认配置（表行总数的 20%）往往导致大表累积数亿死元组后才触发，一经运行便压垮系统。

02“死亡之角”难题

在繁忙的大型业务系统中，调高 Autovacuum 会挤压 IO 导致业务瘫痪，调低则会导致表急剧膨胀，很难找到完美的平衡点。在配置层面，用户无法直观限制其系统负载占比。

03 致命隐患：长事务阻塞

无法清理对活跃长事务可见的死亡元组，导致进程反复扫描无效，引发“VACUUM 风暴”，严重消耗 CPU 和 IO 资源，直接拖垮数据库性能。性能下降又会产生更多长事务，进一步阻塞自动清理，形成“恶性循环”。

总结： VACUUM 是 PostgreSQL 维持数据库健康的基础机制，但其被动式的清理逻辑使其难以应对高并发、长事务的复杂生产环境。

Autovacuum 运维痛点

04 物理复制的进退两难

在解决读库查询被取消的问题上，若调整延迟参数 max_standby_streaming_delay 以保护长查询，会导致主从延迟高。若开启 hot_standby_feedback ，虽能保护查询，但会导致主库由于无法清理而产生严重的表膨胀。这往往是一个无解的死循环。

05 空间无法有效释放

常规清理回收的空间仅能被同表重用，**不会返还给操作系统**。若使用 VACUUM FULL 强制释放，则会**加排它锁**，导致业务完全中断。

总结： VACUUM 是 PostgreSQL 维持数据库健康的基础机制，但其被动式的清理逻辑使其难以应对高并发、长事务的复杂生产环境。

核心挑战二：更新时的写放大

在高并发事务场景下，频繁的元组更新会触发大量 WAL 日志写入与数据页拷贝，
这不仅增加了 I/O 开销，更是制约数据库吞吐能力的关键性能瓶颈。

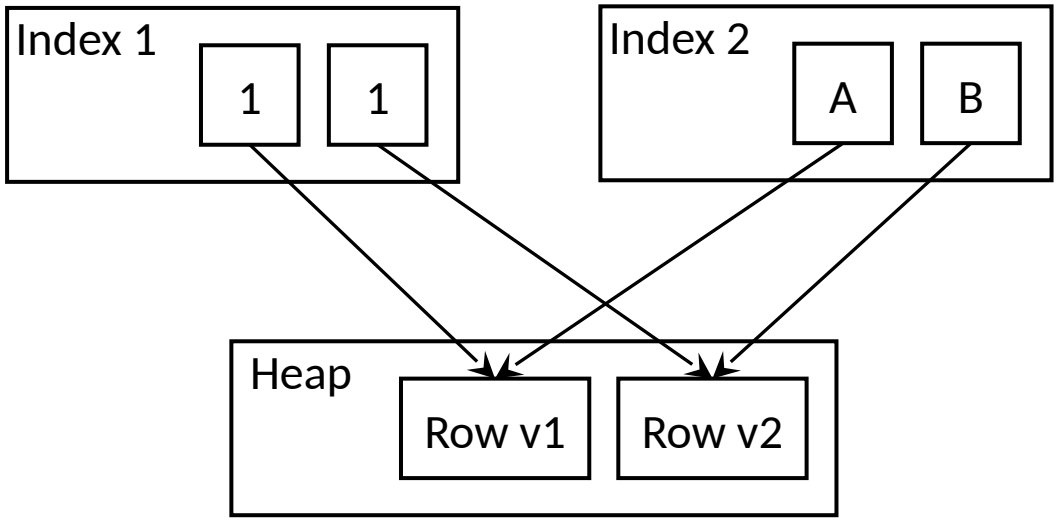
索引写放大：牵一发而动全身

PostgreSQL 的索引存储着指向数据行物理位置（CTID）的指针。当执行 **UPDATE** 操作时，新行版本会生成新的 CTID，这一微小变化会触发连锁反应——所有指向旧版本的索引都必须同步更新。

问题表现：非索引列也“躺枪”
即便更新的并非索引字段，所有关联索引仍需执行“删旧插新”操作，旧 CTID 被移除，新 CTID 被写入，无一幸免。

性能影响：I/O 开销指数级上升
表上索引越多，单次更新触发的索引重写就越多，磁盘 I/O 与 WAL 日志量激增，系统性能被大量冗余写操作严重拖累。

实测案例：更新 500 条非索引列数据，触发约 **1000 个索引页面**重写，写放大效应显著。



图示直观展示了 PostgreSQL MVCC 机制下的行版本与索引指针关系：每次更新都会产生新的行版本，旧版本不会立即删除，而所有索引都必须指向最新的行物理地址，这正是索引写放大的根源所在。

更新写放大与性能不稳定性

01. HOT 更新机制

当更新不会修改表的索引所引用的任何列，且在包含旧行的页面上有足够的空闲空间用于存放更新后的行时，可以不需要添加新的索引条目来定位到更新的行。

02. 更新性能的不可预测性

性能表现高度依赖 fillfactor 参数配置与 HOT 更新的严苛条件。一旦行版本链过长或数据页空间不足，性能会发生断崖式下跌，导致数据库响应时间剧烈波动，难以进行容量规划与性能调优。

实测对比：更新分散在不同页面的 500 条记录，出现的性能差异

PostgreSQL (追加式更新机制)

120.1 ms （更新性能下跌）

11.08 MB （日志量激增， IO 压力大）

Redrock Postgres (就地更新机制)

36.6 ms （更新性能稳定）

3.97 MB （仅为前者 36% ，开销极低）

查询写放大与性能不稳定性

01. 查询写放大根因

在 PostgreSQL 中，执行完 INSERT/UPDATE/DELETE 操作之后，后面在查询访问到这些元组时，还需要进行元组事务状态标记和页面清理，这些动作都会再次修改页面和写入 WAL 日志。

02. 查询性能的不可预测性

在查询访问修改过的表元组时，会更新元组的事务状态标记，这需要记录 WAL 日志。为保障数据一致性，检查点后首次修改脏页会触发“全页写（FPW）”，将整个数据页而非仅修改的部分写入 WAL 日志。

实测对比：更改分散在不同页面的 20000 条记录后，进行查询出现的性能差异

PostgreSQL（延迟清理机制）

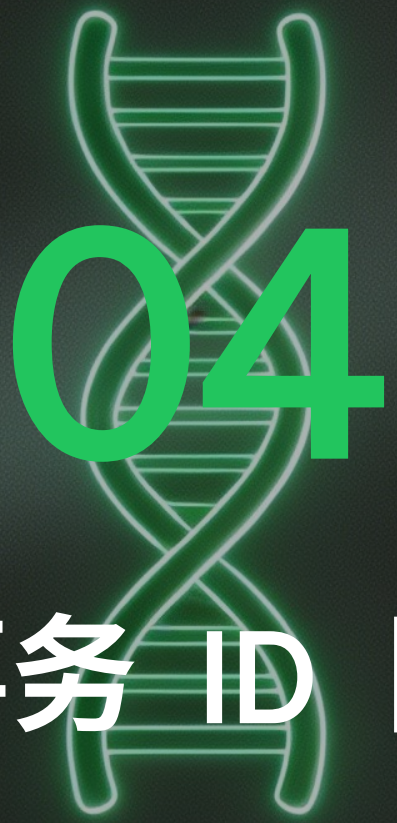
19592 （查询修改了大量页面，IO 压力大）

149 MB （WAL 日志量激增）

Redrock Postgres（提交清理机制）

3 （查询只修改了少量页面）

14.2 KB （WAL 日志开销极低）



04

核心挑战三：事务 ID 回卷的定时炸弹

事务 ID 循环复用引发的数据一致性风险，是数据库运行中隐蔽且极具破坏力的潜在危机，威胁系统稳定性基石。

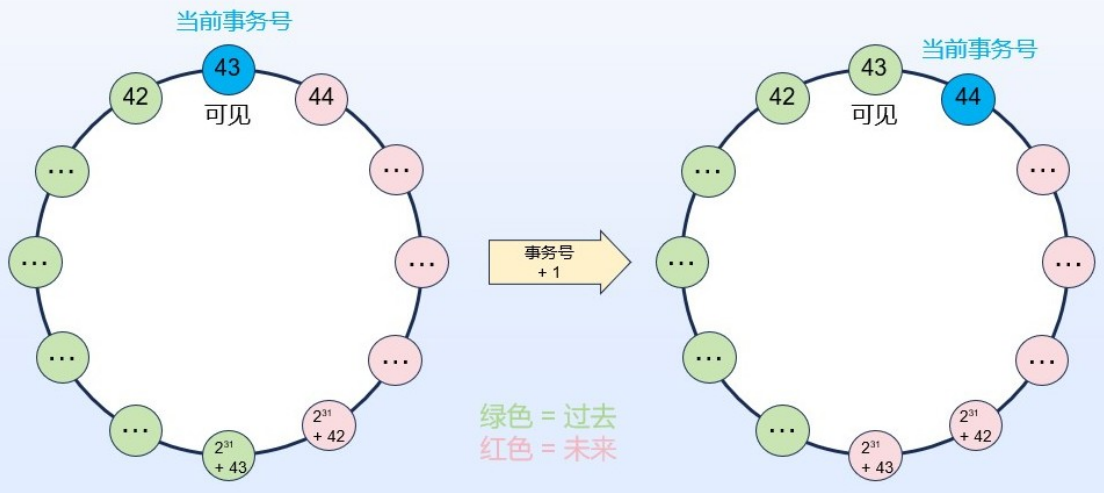
32 位 XID 的先天不足与 Freeze 机制

XID 循环的致命缺陷

基于 32 位无符号整数设计，最大值约为 20 亿。一旦耗尽将循环复用，导致数据库无法正确判断事务的新旧关系，引发可见性异常与数据损坏风险。

Freeze：事务的“青春永驻”

VACUUM 进程会将极旧的元组 XID 替换为永不回卷的 FrozenXID，并更新系统已完成冻结的 XID 值，从而重置 XID 可用空间，从根源上阻止回卷发生。



高并发下的“加速倒计时”

在高 TPS 场景中， XID 消耗极快， Freeze 需高频执行。若存在长事务阻塞清理进程，数据库将因“ XID 即将耗尽”触发保护机制，**强制停止所有写操作**，导致业务服务中断，这是 PostgreSQL 运维中需重点防范的风险点。

核心应对策略：建立 XID 水位监控体系，优化长事务代码逻辑，确保 VACUUM 进程稳定运行。

冻结频率与事务吞吐量 — 跑得越快，风险越大！

在 PostgreSQL 中，元组冻结操作（通过 VACUUM 命令执行）的频率与每秒修改数据的事务数（TPS，包括 INSERT、UPDATE、DELETE 操作）成正比。由于 PostgreSQL 采用 32 位事务 ID（XID），其事务 ID 会在累计约 20 亿次事务后发生回卷，因此高吞吐系统（高写入 TPS）需要更频繁地执行冻结操作，以防止因事务 ID 回卷造成数据丢失。

数据表达到默认的 autovacuum_freeze_max_age 阈值的耗时，会随事务 ID 消耗速率的提升而缩短。

事务 ID 消耗速率（每秒）	达到 20 亿事务阈值的时长	执行冻结操作的周期
1000	23 天	2.3 天
10000	2.3 天	5.5 小时
100000	5.5 小时	33 分钟
1000000	33 分钟	3.3 分钟

核心警示： PostgreSQL 使用了子事务来实现保存点的特性，PL/pgSQL 对每个包含 EXCEPTION 子句的块都会使用一个子事务。子事务的大量使用会加速事务 ID 的消耗速率。

事务 XID 防回卷的三层防护机制

01 常规 Autovacuum

可中断、带延迟节流：常规 Autovacuum 会在后台默默运行，持有 SHARE UPDATE EXCLUSIVE 锁。如果业务端发起了需要修改表结构的 DDL 语句，常规的清理进程会非常“识趣”地中断自己并释放锁，对业务 SLA 毫无影响。

02 紧急 Autovacuum

不可中断、无延迟节流：防事务 ID 回卷的 Autovacuum 肩负着防止数据丢失的生死任务， PostgreSQL 在内部将其设定为不可中断。这就意味着，哪怕全表扫描需要耗费好几个小时，它也会死死咬住手里的锁，绝不向任何冲突操作让步。

03 数据库拒绝服务

直接停止服务： 数据库因“ XID 即将耗尽”触发保护机制，强制停止所有写操作，导致业务服务中断，这是 PostgreSQL 运维中需重点防范的风险点。

低 风险系数 高

05

核心挑战四：多核并发下的性能杀手

解析数据库内核在高并发场景下的资源竞争与调度损耗，探寻性能优化的关键突破口

热点行更新的性能退化

01 版本链的恶性循环

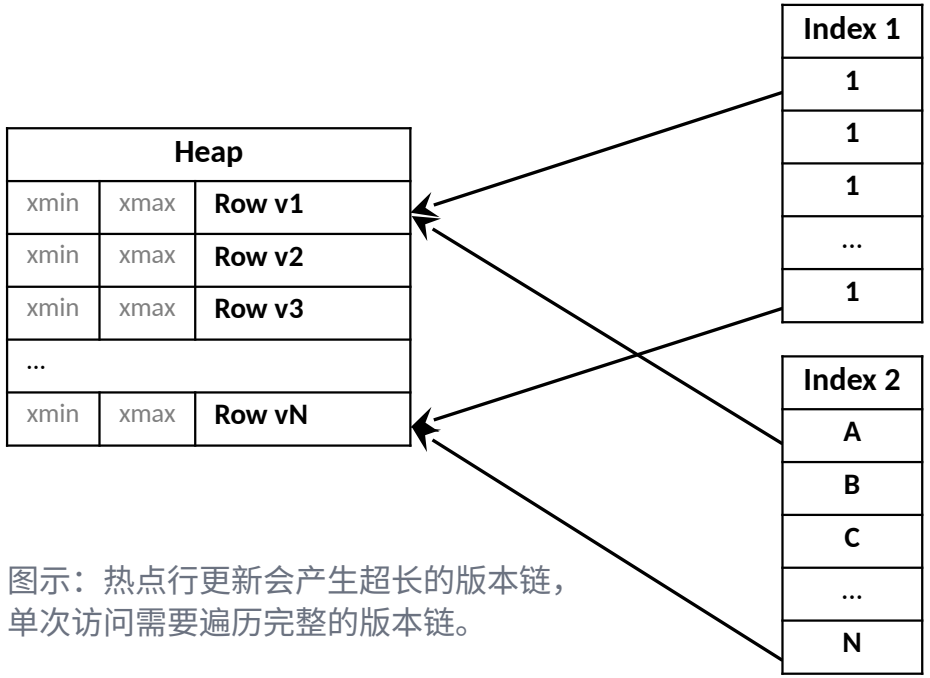
每次更新都会生成新的版本快照，形成“旧→新”的链式结构。后续更新必须从链头全量遍历才能定位修改，这为性能埋下了结构性隐患。

02 性能呈 $O(N^2)$ 坍塌

随着更新频次增加，版本链长度 N 线性增长，单次更新的遍历耗时同步上升，整体性能开销随 N^2 指数级恶化，系统响应急剧变慢。

03 长事务阻断清理机制

未结束的长事务会阻止 VACUUM 回收旧版本元组，导致版本链无限膨胀，最终使单条热点行的更新操作彻底阻塞，引发数据库雪崩。



图示：热点行更新会产生超长的版本链，单次访问需要遍历完整的版本链。

真实压测：长事务导致 TPS 断崖式下跌

14,519 TPS
正常场景下的吞吐量峰值

500 TPS
存在长事务时的吞吐量

性能下降幅度近 **30 倍**，这意味着系统在高并发热点更新场景下，会直接从高可用状态退化为无法支撑业务的不可用状态。

子事务的应用场景

PostgreSQL 使用了子事务来实现保存点的特性，下面的操作在同一个事务中，使用保存点创建了超过 64 个子事务：

01 使用保存点插入 70 行

```
CREATE TABLE IF NOT EXISTS subxid_test (id int);

BEGIN;
SAVEPOINT s1;
INSERT INTO subxid_test VALUES (1);
SAVEPOINT s2;
INSERT INTO subxid_test VALUES (2);
...
SAVEPOINT s70;
INSERT INTO subxid_test VALUES (70);
```

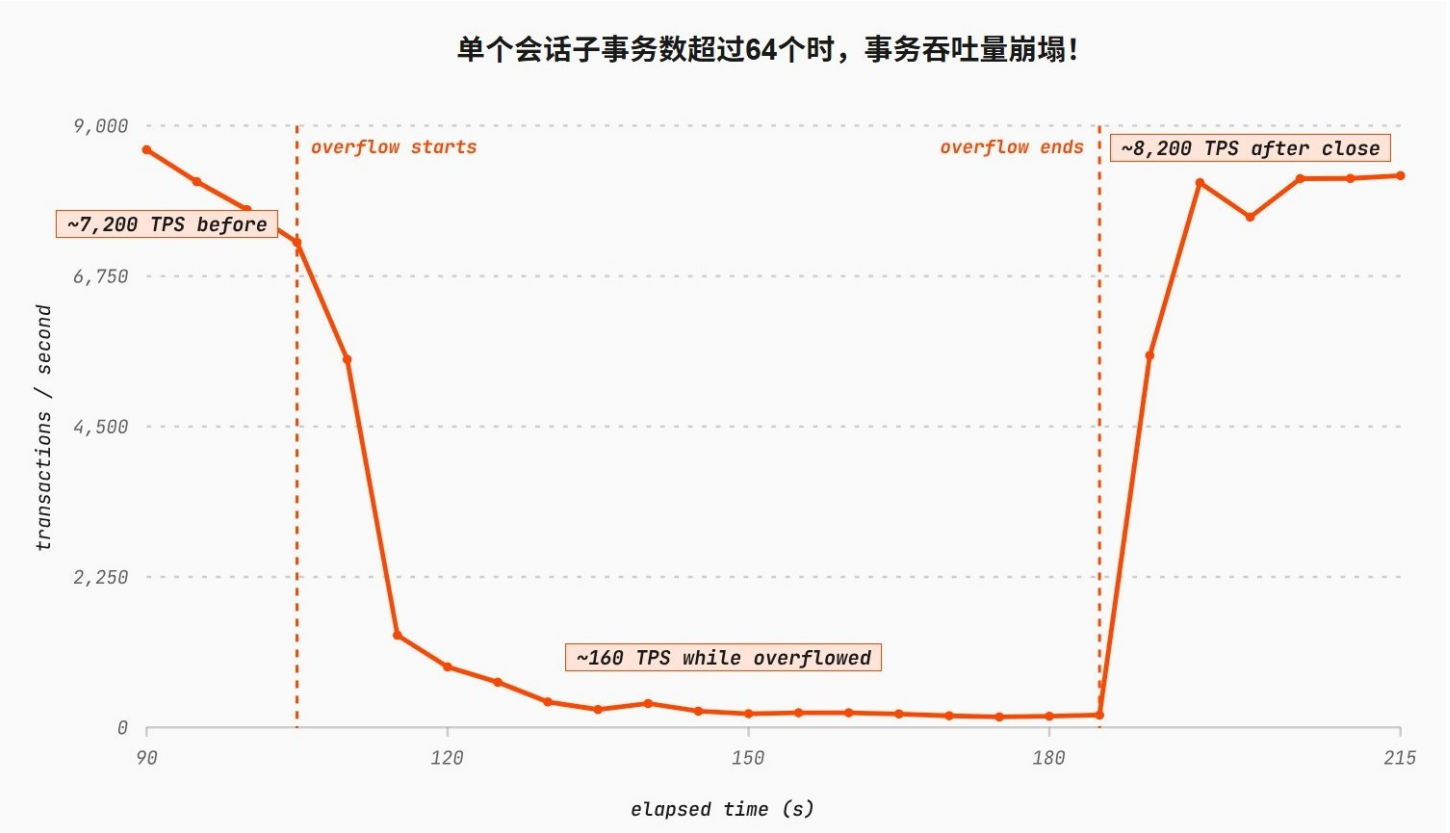
PL/pgSQL 对每个包含 EXCEPTION 子句的块都会使用一个子事务。下面的循环在没有直接发出 SAVEPOINT 的情况下创建了超过 64 个子事务：

02 使用异常代码块插入 70 行

```
DO $$
BEGIN
    FOR i IN 1..70 LOOP
        BEGIN
            INSERT INTO subxid_test VALUES (i);
        EXCEPTION WHEN OTHERS THEN
            RAISE;
        END;
    END LOOP;
END $$;
```

子事务与锁争抢雪崩

单个事务使用了大量子事务 ID 后，可能会显著降低整个 PostgreSQL 集群的吞吐量。
即使副本在持续回放 WAL 日志，它也可能无法接受新的查询请求。



核心警示：单个事务如果积累了超过 64 个已分配且未中止的子事务 ID，可能会导致整个 PostgreSQL 集群的查询变慢，包括那些自己没有使用子事务的会话。

06

综合评估：各业务场景下的表现

深度解析 · 场景适配 · 性能基准

OLTP / OLAP / HTAP 场景综合评分

01 OLTP 事务处理

高并发 · 短事务 · 频繁更新

优势：完善的 ACID 事务保障，生态工具链成熟。
短板：更新写放大、垃圾清理与 XID 冻结难于控制，高并发下性能受限。

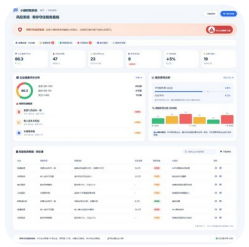
综合评分：★★★☆☆ （中等）

02 OLAP 数据分析

海量数据 · 复杂查询 · 读密集型

优势：强大的查询优化器，支持复杂 SQL 与并行计算。
短板：批量加载速度较慢，不支持列存和向量化执行引擎。

综合评分：★★★★☆ （优秀）



03 HTAP 混合负载

事务 + 分析 · 实时性高 · 资源竞争

优势：避免数据迁移，理论上兼顾两类需求。
短板：大型分析查询会阻塞垃圾清理与 XID 冻结，难以做到性能与稳定性的平衡。

综合评分：★★☆☆☆ （受限）

总结： PostgreSQL 是功能完备的开源数据库，在中等规模 OLTP 和复杂 OLAP 场景下表现卓越，生态极其成熟。但在超大规模高并发写入或严苛的 HTAP 混合负载场景中，其传统架构和 MVCC 机制会成为瓶颈，通常需要结合读写分离、分库分表或引入专用分析引擎来弥补不足。

核心数据汇总

01 市场与社区表现

开发者采用率： 55.6%
Claude 采用率： 58.4%
数据来源： Stack Overflow / Amplifying

02 空间膨胀实测

全表更新： 35MB →104MB
删插测试：体积膨胀 2.67 倍
数据来源： Redrock 实验室测试

03 写放大与日志开销

非索引列更新：
单表 WAL 日志量达 11.08MB
远超常规预期，显著增加 IO 压力

04 稳定性与并发挑战

事务 ID 回卷：
10 万 TPS 下仅需 5.5 小时即触达 Freeze 阈值，维护成本高。

热点行并发：
长事务下 TPS 从 1.4 万降至 500，高并发场景性能骤降。

05 索引扫描性能衰减

在频繁更新与高负载场景下，索引扫描延迟翻倍：执行耗时从基线 55ms 恶化至 118ms，直接影响核心业务的查询响应速度与吞吐量。

Redrock Postgres

https://www.rockdata.net

欢迎随时交流与提问，共同探讨技术与创新的


2024
数据风云
2021
数造未来
2022
分布式
2025
人工智能
2023
大数据
2019
2016
2014
2015
THANKS