

# 开发者PG TOP 18问

Digoal

# 我是: 德哥



digoal's 微信

江湖人称: 德哥

帮会: PostgreSQL社区

帮会职务: 校长

正经职务: 阿里云数据库产品经理

擅长武器: PostgreSQL

必杀技: 删库跑路

愿景: 没有PG解决不了的问题,如果有,就多买几台

格言: 公益是一辈子的事

业余爱好: 写博、写专利赚钱、毁人不倦

github: <https://github.com/digoal>

# 继续庆祝PG 13 于2020-09-24正式发布

- PG 13 它牛逼在哪?
- PG 13 它贴心在哪?
- PG 13 它烧脑在哪?
- PG 13 它奔放在哪?
- "PostgreSQL 13展示了**全球社区**的**协作**和**奉献**精神。", "  
每个发行版带来的**创新**,以及PG在**可靠**性和**稳定**性方面的声  
誉, 是越来越多的企业选择使用PostgreSQL的原因"。

打死也不说:

[https://github.com/digoal/blog/blob/master/202009/20200926\\_01.md](https://github.com/digoal/blog/blob/master/202009/20200926_01.md)



# 目录

- 开发者 PG TOP 18问

# TOP 18问有没有代表性?

- 问题来源?
  - 全球最大的共享出行服务商HelloBike
- 回答够不够权威?
  - <PG大学>校长



# pg 可以使用hint指定走某个索引吗？

- 为什么要问这个问题?
  - 优化器没有选择合适索引
    - 为什么优化器没有选择合适索引
      - 统计信息维度不够、统计信息不准确、算子参数不准确
      - 为什么统计信息维度不够
        - 为什么不增加统计信息维度
        - 为什么统计信息不准确
        - 为什么没有触发统计
          - 为什么算子参数不准确
          - 还想要用HINT吗?
  - 有哪些优化器? 分别解决什么问题?
    - 自动: cbo, geqo, aqo
    - 人为: srplan, hint
    - 自动化趋势不可逆转, 未来一定是自动化的天下.
  - 答案
    - 废话真多, 用 pg\_hint\_plan 插件

pg 针对分表要新增一个字段， 需要对每个分表编辑吗？

- 为什么要问这个问题?
  - 废话, 躺着赚钱不香吗
- 答案
  - 不需要, 直接操作主表, PS: 所有DDL都有排他锁, 注意防止雪崩(后面会讲).

# pg 在保持性能的前提下，最多能存多少条数据呀？

- 为什么要问这个问题?
  - 一定是没有遇到过性能瓶颈，否则不就知道了么，还有毛线疑问
- 答案
  - 摩尔定律+安迪比尔定律
- 52C 384GB 16TB SSD
  - PG 12 TPCB 100亿
  - RO: 100万量级qps
  - RW: 50万量级qps

# pg 建议在多少条记录时分库/分表？

- 为什么要问这个问题?
  - 寻址上限, ctid(pageid, offset)->pageid 32位->pagesize(2k-32k) ->最大128TB每分区
  - 存储上限, 表空间->目录->文件系统->卷->块设备
  - 性能上限,
    - 垃圾回收, table/process, index/multiprocess, (更新模型, 瓶颈与 IO|CPU 匹配)
    - 建索引, 支持并行, 10亿记录, 创建索引252秒
    - freeze, 32bit xid, 只写(一次性freeze)、 更改(版本变化后需要freeze)
    - 逻辑备份, 快照, 垃圾回收oldest位点, DDL互斥
    - rewrite table, 某些情况需要重写表(vacuum full, 更改字段类型导致内部存储值发生变化, 老版本加字段默认值)
- 答案
  - SSD或者10万级+IOPS
  - 以下都是废话:
  - 更新多的表
    - 16GB or 1亿 / 分区
  - 更新少的表
    - 64GB / 分区
  - 分库/建只读实例
    - 热数据>内存/2 , 写入瓶颈, 查询瓶颈

# pg delete记录后水位线会下降吗？

- 为什么要问这个问题?
  - 水位不下降会怎么样?
  - 水位下降有什么好处?
- 答案
  - 索引, 复用空间, 不回收, 不降水位.
  - HEAP, 复用空间, 无有效记录的空页为什么不能从文件系统回收空间?
    - 试想索引如何检索记录? 当记录在HEAP末尾页, 不在末尾页时有什么区别?
    - 末尾连续空页可以直接回收, 其他空页无法直接回收
  - 水位没降下来怎么办?
    - 表未来还要不要继续写入?
    - vacuum full, pg\_repack , (rewrite, 前提保留足够存储空间)

# pg 一般tps/qps能达到多少？

- 为什么要问这个问题?
  - 任何没有说明环境、场景、测试方法的性能指标都没有参考价值
- 答案
- 52C 384GB 16TB SSD
  - PG 12 TPCB 100亿
  - RO: 100万量级qps
  - RW: 50万量级qps

# pg 的统计信息收集是按照什么规则？

- 为什么要问这个问题?
  - 一定是遇到过统计信息不准、未开启, 引起的SQL执行计划不准的问题
- 答案
  - autovacuum\_analyze\_scale\_factor
  - autovacuum\_analyze\_threshold
  - table | global level set.
- 开启自动收集->计数器->监工发现需要统计的表->分配工人干活->更新统计信息

# pg 支持json/jsonb吗？

- 为什么要问这个问题?
  - 敏捷开发是把双刃剑
- 答案
  - PG: 无敌的sqljson功能
    - sql 2016的sql/json标准有15条， PG 支持14道标准
      - PG (12 14/15),
      - oracle(18c 11/15),
      - mysql(8.0.4 5/15),
      - sqlserver(2017 2/15)。
    - 索引支持
    - jsonpath搜索语法  
[https://github.com/digoal/blog/blob/master/202010/20201013\\_01.md](https://github.com/digoal/blog/blob/master/202010/20201013_01.md)

Table 8.25. jsonpath Accessors

Accessor Operator	Description
. <b>key</b> . "\$varname"	Member accessor that returns an object member with the specified key. If the key name matches some named variable starting with \$ or does not meet the JavaScript rules for an identifier, it must be enclosed in double quotes to make it a string literal.
.*	Wildcard member accessor that returns the values of all members located at the top level of the current object.
.**	Recursive wildcard member accessor that processes all levels of the JSON hierarchy of the current object and returns all the member values, regardless of their nesting level. This is a PostgreSQL extension of the SQL/JSON standard.
.**{ <b>level</b> } .** { <b>start_level</b> to <b>end_level</b> }	Like .**, but selects only the specified levels of the JSON hierarchy. Nesting levels are specified as integers. Level zero corresponds to the current object. To access the lowest nesting level, you can use the last keyword. This is a PostgreSQL extension of the SQL/JSON standard.
[ <b>subscript</b> , ...]	Array element accessor. <b>subscript</b> can be given in two forms: <b>index</b> or <b>start_index</b> to <b>end_index</b> . The first form returns a single array element by its index. The second form returns an array slice by the range of indexes, including the elements that correspond to the provided <b>start_index</b> and <b>end_index</b> .  The specified <b>index</b> can be an integer, as well as an expression returning a single numeric value, which is automatically cast to integer. Index zero corresponds to the first array element. You can also use the last keyword to denote the last array element, which is useful for handling arrays of unknown length.
[ * ]	Wildcard array element accessor that returns all array elements.

# pg 索引类型，除了btree还有哪些，可以建立btree-gist联合索引吗？

- 为什么要问这个问题?
  - 一定是遇到事了!!!
  - 《[直播]为什么饿了么网上订餐不会凉凉 & 牛顿发现万有引力有关?》
  - [https://github.com/digoal/blog/blob/master/202010/20201018\\_01.md](https://github.com/digoal/blog/blob/master/202010/20201018_01.md)
  - 《[直播]为什么打车和宇宙大爆炸有关?》
  - [https://github.com/digoal/blog/blob/master/202009/20200926\\_02.md](https://github.com/digoal/blog/blob/master/202009/20200926_02.md)
- 答案
  - 后面会讲到PG支持哪些索引
  - 可以, `create extension btree_gist;`
  - 性能提升的感觉
    - 就像男人看到美女后, 荷尔蒙爆表.

canceling statement due to conflict with recovery什么情况，怎么处理？

- 为什么要问这个问题?
  - 一定是用了只读实例, 一定是SQL被莫名其妙的KILL过.
    - [https://github.com/digoal/blog/blob/master/202005/20200518\\_01.md](https://github.com/digoal/blog/blob/master/202005/20200518_01.md)
- 答案
  - rw instance -> redo -> readonly instance -> startup process replay redo -> conflict with query -> recovery等query -> 等待是有限度的 -> cancel query
    - replay的redo里面包含什么信息时, 会和query冲突?
    - 最常见的情况: vacuum某些tuple version, 与query快照的xid相冲突
      - 从节点调大replay等待时长, 主节点设置延迟回收, 从节点设置query feedback
      - 可能导致主节点vacuum出现无用功, 或者膨胀
    - 其他: 删除表空间, 锁冲突, pinned buffer, 死锁等.

在频繁更新和删除的系统中，如何比较好地避免表和索引膨胀从而引起的sql效率降低？

- 为什么要问这个问题?
  - 掉坑里过?
  - [https://github.com/digoal/blog/blob/master/201906/20190621\\_01.md](https://github.com/digoal/blog/blob/master/201906/20190621_01.md)
- 答案
  - 开启自动垃圾回收, autovacuum
  - 配置足够多工人, autovacuum\_max\_workers
  - 配置足够频繁的监测周期, autovacuum\_naptime
  - 配置足够小的触发阈值, autovacuum\_vacuum\_scale\_factor, autovacuum\_vacuum\_threshold
  - 避免工人频繁休息, autovacuum\_vacuum\_cost\_delay, autovacuum\_vacuum\_cost\_limit
  - 避免工人做无用功, oldest xid snapshot, standby feedback, vacuum defer, old\_snapshot\_threshold, long query, long xact, long 2pc.
  - 避免超大单分区, 因为单个分区单个vacuum工人为之服务, 无法并行
  - 避免重复扫描索引, autovacuum\_work\_mem, 单个表分区的垃圾记录数\*17字节不要超出 autovacuum\_work\_mem

# pg 大表加字段及默认值会锁表吗？哪些版本能很好地解决这种情况？

- 为什么要问这个问题?
  - 一定是业务经常要半夜加字段？被DBA吐槽过？
- 答案
  - PG 11及以后的版本，加自动和默认值不需要rewrite table.
    - 其他版本加字段不含默认值不需要rewrite table, 包含默认值需要rewrite table.
  - 锁不锁关系不大，关键是：
  - 多大的锁，排他，与任何其他锁都会发生冲突
  - 锁影响多长时间
    - 取决于整个过程要多久
  - 未持有锁，等待中，会不会与其他会话发生冲突
    - 会，大多数人栽在这里。
  - 如何避免雪崩
    - 执行DDL前，设置锁请求超时，然后再执行DDL

# pg 大表更改字段类型会锁表吗？

- 为什么要问这个问题?
  - 是业务设计有问题? 该用数值错用字符串? 被DBA吐槽过?
- 答案
  - 一切DDL都会锁表
  - 影响有多大, 取决于锁时长
    - 数据内部存储未变化, 不需要rewrite table, 仅修改元数据
    - 数据内部存储发生变化, 需要rewrite table

# pg 想一次性对表（包括将来新建的表）赋权该如何操作？

- 为什么要问这个问题?
  - mysql用户的问题? <https://www.postgresql.org/docs/12/sql-alterdefaultprivileges.html>
  - <https://www.postgresql.org/docs/12/sql-grant.html>
- 答案
  - GRANT { { SELECT | INSERT | UPDATE | DELETE | TRUNCATE | REFERENCES | TRIGGER }  
[,...] | ALL [ PRIVILEGES ] }
  - ON { [ TABLE ] table\_name [, ...]  
| ALL TABLES IN SCHEMA schema\_name [, ...] }
  - TO role\_specification [, ...] [ WITH GRANT OPTION ]
  - ALTER DEFAULT PRIVILEGES  
[ FOR { ROLE | USER } target\_role [, ...] ]  
[ IN SCHEMA schema\_name [, ...] ]  
abbreviated\_grant\_or\_revoke
  - where abbreviated\_grant\_or\_revoke is one of:
    - GRANT { { SELECT | INSERT | UPDATE | DELETE | TRUNCATE | REFERENCES | TRIGGER }  
[,...] | ALL [ PRIVILEGES ] }
    - ON TABLES
    - TO { [ GROUP ] role\_name | PUBLIC } [, ...] [ WITH GRANT OPTION ]

ERROR: index row requires 8600 bytes,  
maximum size is 8191, 怎么办？

- 为什么要问这个问题?
  - 设计问题? 未限制字段长度. 然后报错让DBA去解决?
- 答案
  - hash index
    - hash value长度固定
  - function index
    - func(col), 查询时也使用func(col)
  - partial index
    - index on tbl (col) where col\_length < 8192
    - index on tbl (1) where col\_length >= 8192
    - select x on tbl where ... and col\_length < 8192 union all ... and col\_length >= 8192

current transaction is aborted, commands ignored until end of transaction block, 这是啥，怎么办？

- 为什么要问这个问题？

- 开启事务后, 哪种处理方法更适合你?
  - 1、数据库出错了不告诉你, 还让你继续执行
  - 2、数据库出错了, 告诉你, 并且需要你handle

- 答案

- begin;...错误; end; 结束事务后再发起新事务
- 使用自动事务提交
- 或者看看你用的驱动有没有相关参数、相关模块可以自动结束事务.

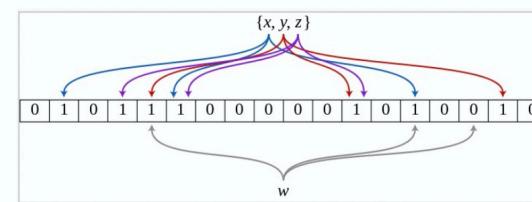
# mysql不支持位图存储、PG支持位图存储吗？

- 为什么要问这个问题?
  - 什么情况适合用位图索引?
    - 数据量大, 唯一值集合小, 数据变更少, 按单值、组合值查询count?
- 答案
  - 8.2支持过, 后来从代码中去掉了, 取而代之的:
    - GIN
    - Bloom
  - 更通用.

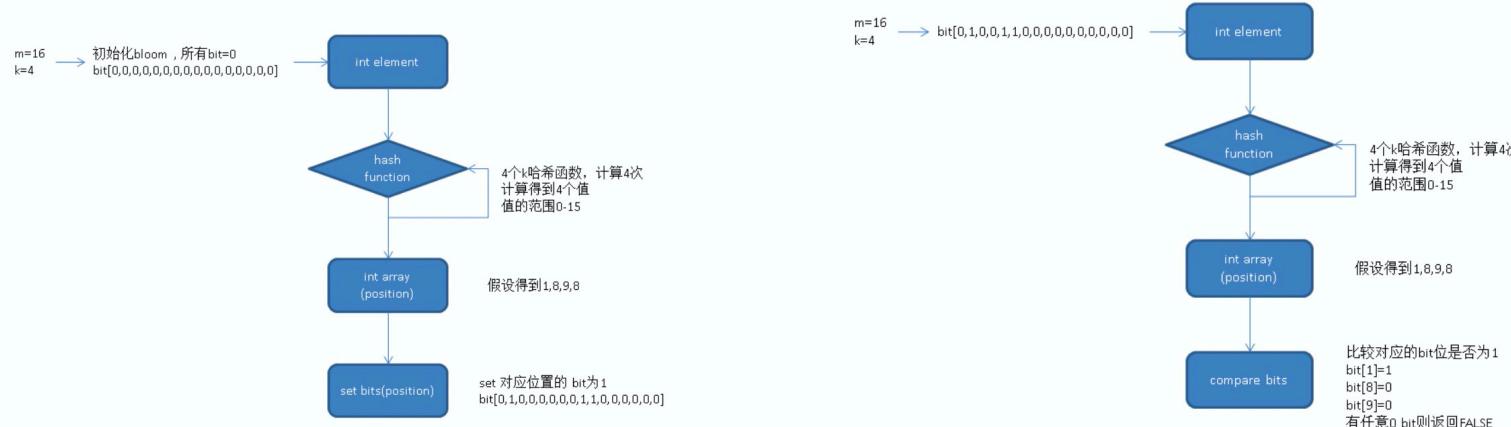
# mysql不支持位图存储、PG支持位图存储吗？

## bloom索引结构

- bloom
  - <https://www.postgresql.org/docs-devel/static/bloom.html>
  - [https://github.com/digoal/blog/blob/master/201605/20160523\\_01.md](https://github.com/digoal/blog/blob/master/201605/20160523_01.md)
  - [https://en.wikipedia.org/wiki/Bloom\\_filter](https://en.wikipedia.org/wiki/Bloom_filter)



An example of a Bloom filter, representing the set  $\{x, y, z\}$ .  
The colored arrows show the positions in the bit array that each set element is mapped to. The element  $w$  is not in the set  $\{x, y, z\}$ , because it hashes to one bit-array position containing 0. For this figure,  $m = 18$  and  $k = 3$ .



# bloom索引结构

- bloom

- <https://www.postgresql.org/docs-devel/static/bloom.html>
- [https://github.com/digoal/blog/blob/master/201605/20160523\\_01.md](https://github.com/digoal/blog/blob/master/201605/20160523_01.md)
- [https://en.wikipedia.org/wiki/Bloom\\_filter](https://en.wikipedia.org/wiki/Bloom_filter)

```
CREATE INDEX bloomidx ON tbloom USING bloom (i1,i2,i3)
WITH (length=80, col1=2, col2=2, col3=4);
```

length, m值, 即总共多少个bit位表示一个被索引的行。

Col1,col2,..., 该列用几个bit表示, 每个bit对应position是否设置为1由一个hash函数计算得到.  
最多需要m个不同的hash函数, hash函数返回的值范围是0 ~ m-1 (即bit position).

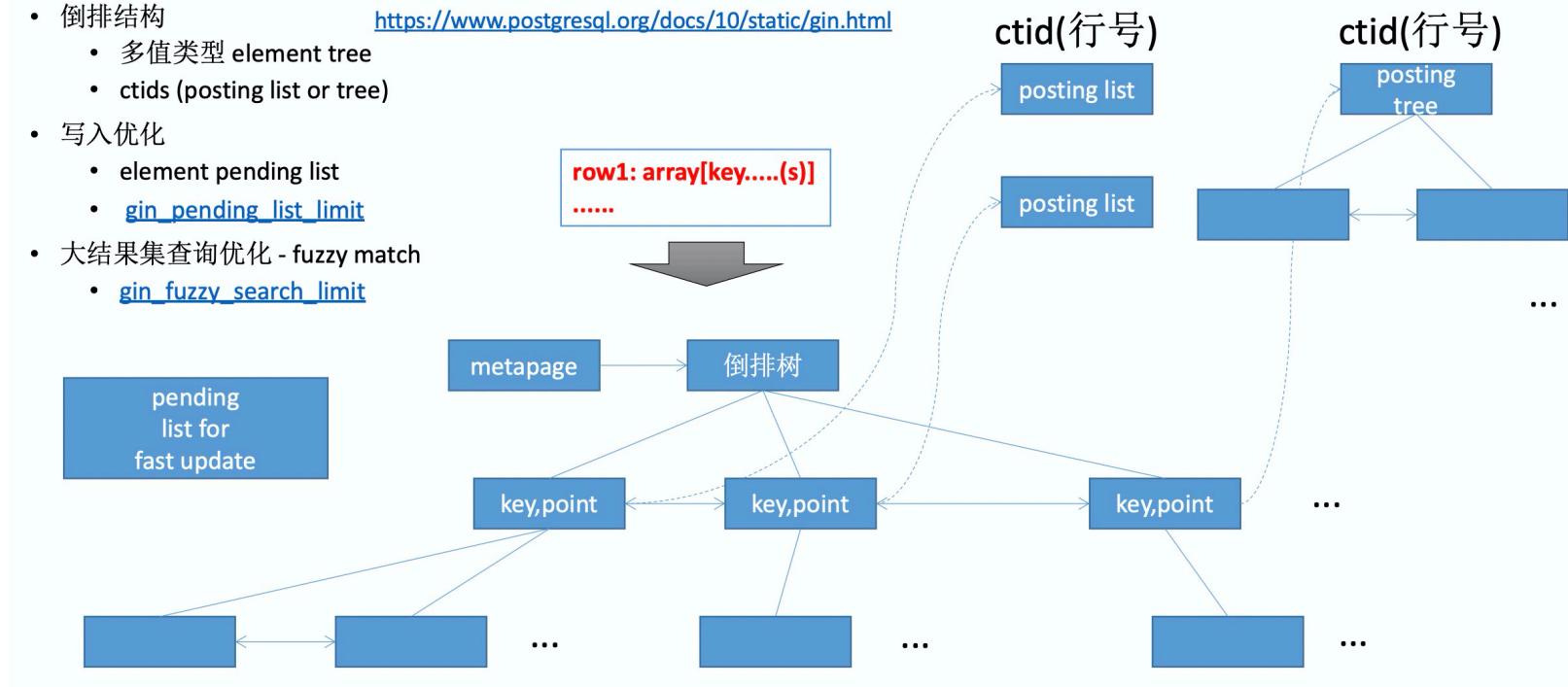
假设col1=3, 则索引的第一列需要3个hash函数来计算出3个bit position上的值是1或0.

假设m=10

i1='abc', hash1('abc')=3, hash2('abc')=5, hash3('abc')=9, 那么m=0001010001

# gin索引结构

- src/backend/access/gin/README
- 倒排结构 <https://www.postgresql.org/docs/10/static/gin.html>
  - 多值类型 element tree
  - ctids (posting list or tree)
- 写入优化
  - element pending list
  - [gin\\_pending\\_list\\_limit](#)
- 大结果集查询优化 - fuzzy match
  - [gin\\_fuzzy\\_search\\_limit](#)



`gin_page_opaque_info` returns information about a GIN index opaque area, like t

```
test=# SELECT * FROM gin_page_opaque_info(get_raw_page('gin_index', 2));
 rightlink | maxoff |      flags
-----+-----+
      5 |      0 | {data, leaf, compressed}
(1 row)
```

`gin_leafpage_items` returns information about the data stored in a GIN leaf page. For example:

### 内窥GIN内容

<https://www.postgresql.org/docs/>

```
test=# SELECT first_tid, nbytes, tids[0:5] AS some_tids
        FROM gin_leafpage_items(get_raw_page('gin_test_idx', 2));
 first_tid | nbytes |          some_tids
-----+-----+
 (8, 41) |    244 | {"(8, 41)", "(8, 43)", "(8, 44)", "(8, 45)", "(8, 46)"}
 (10, 45) |    248 | {"(10, 45)", "(10, 46)", "(10, 47)", "(10, 48)", "(10, 49)"}
 (12, 52) |    248 | {"(12, 52)", "(12, 53)", "(12, 54)", "(12, 55)", "(12, 56)"}
 (14, 59) |    320 | {"(14, 59)", "(14, 60)", "(14, 61)", "(14, 62)", "(14, 63)"}
 (167, 16) |   376 | {"(167, 16)", "(167, 17)", "(167, 18)", "(167, 19)", "(167, 20)"}
 (170, 30) |   376 | {"(170, 30)", "(170, 31)", "(170, 32)", "(170, 33)", "(170, 34)"}
 (173, 44) |   197 | {"(173, 44)", "(173, 45)", "(173, 46)", "(173, 47)", "(173, 48)"}
(7 rows)
```

# 极大丰富索引接口



连学霸都竖起大拇指

- btree: 等值、范围、排序、唯一约束
- hash: 等值
- gin, 数组包含、相交; 全文检索; 模糊查询; 正则匹配; JSON搜索; 相似查询; 任意字段组合等值搜索;
- gist: R tree, RD tree通用自定义平衡树; 地理信息搜索、距离排序; 全文检索; 多维向量距离排序; 排他约束
- spgist: 通用自定义非平衡树; quad tree, k-d tree, radix tree; 空间搜索; 排他约束;
- brin: 时序区间搜索; 线性相关存储数据搜索;
- bloom: 任意字段组合等值、不等过滤;
- rum: 全文检索; 文本相似; 数组相似;
- zombodb: ElasticSearch扩展引擎索引; (数据存PG, 索引在ES)
- pase: 阿里云PG专供: 多维向量距离排序; 图像识别; 相似圈选;

# 学霸配方在此!!!

- 提前收藏:
  - <https://github.com/digoal/blog>



# 总结一下

- PG学习门槛高?
  - 90%的人: 不高
    - 大多数业务根本用不到极大丰富的高级功能
- PG有内涵吗?
  - 如果你把它当成简单增删改查的数据存储, 和其他DB没什么两样.
  - 如果你把它当成数据工厂, 想让数据发挥价值, 它会给你无限惊喜.
    - 在应用中实现的逻辑都可以在PG中实现
- PG极大丰富的功能到底有啥好处?
  - 万众创新

# 我们正在经历的第四次数据库大战(2020s)

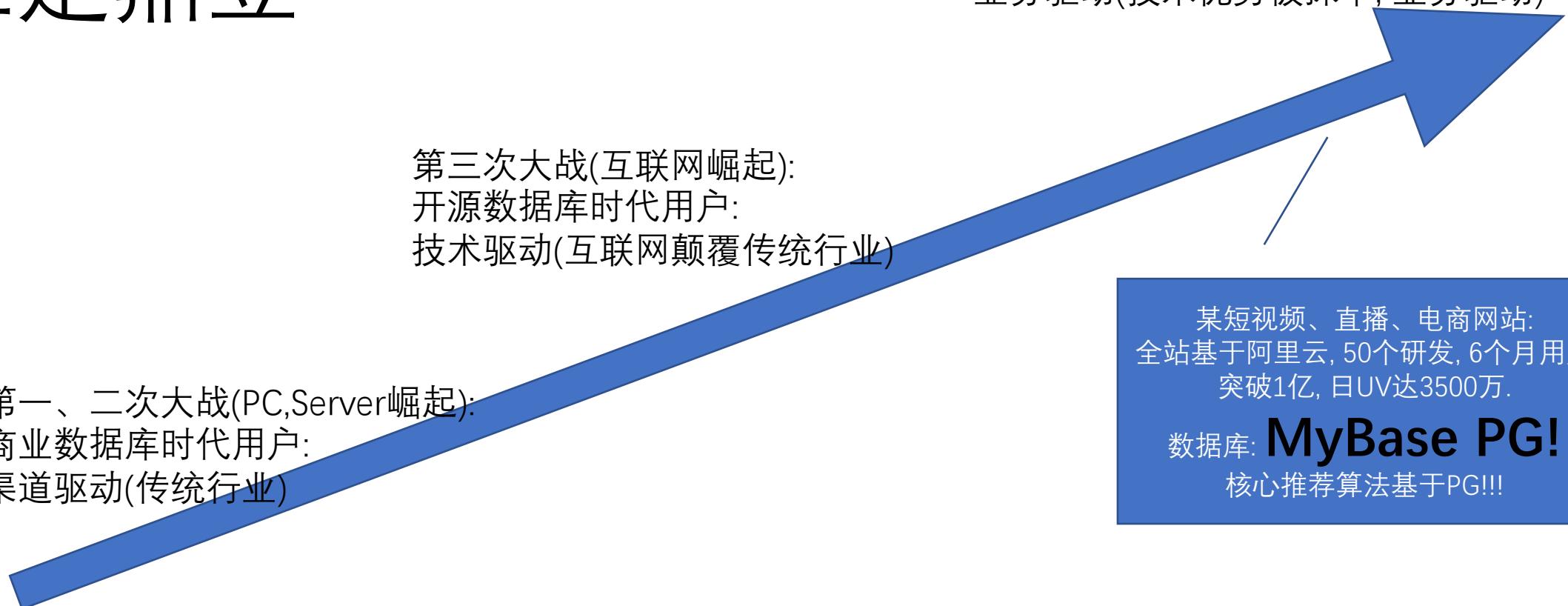
- 云 VS 开源 VS 商业

三足鼎立

第一、二次大战(PC, Server崛起):  
商业数据库时代用户:  
渠道驱动(传统行业)

第三次大战(互联网崛起):  
开源数据库时代用户:  
技术驱动(互联网颠覆传统行业)

第四次大战(万众创新):  
数据即服务时代用户:  
业务驱动(技术优势被抹平, 业务驱动)



某短视频、直播、电商网站:  
全站基于阿里云, 50个研发, 6个月用户  
突破1亿, 日UV达3500万.  
数据库: **MyBase PG!**  
核心推荐算法基于PG!!!

明天就来上班!



# 憋废话了，到底哪里有 MyBase？

- 入钉钉群，**咨询砖家**
- **免费** 活动进行中!!!
- **周周** 有直播和**红包雨**!!!

