

在数据库中跑全文检索、
模糊查询SQL会不会被开除？

Digoal

灵魂拷问又来啦, 我是谁?

周四晚上在群里瞎BB

《亿级用户量的实时推荐数据库到底要几毛钱?》
的那个德哥!!!



江湖名号: 德哥

帮会: PostgreSQL社区

帮会不正当职务: 校长

擅长武器: PostgreSQL

雕虫小技: 全文检索、模糊搜索、化学分析、
人脸识别、相似推荐、时空调度、向量搜索...

必杀技: 删库跑路

正经职务: 阿里云数据库产品经理

愿景: **没有MyBase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、分享、毁人不倦

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

联系方式:

目录

- 为什么要讨论这个问题?
- 到底会不会被开除?
- 问题在哪?
- 原因是什么?
- 怎么破?
- 搜来的方案靠谱吗?
- 什么才是经过思考的牛逼方案?
- 牛逼的方案就没漏洞吗?
- 更牛逼的方案是什么?
- 如何学会保护自己和公司业务?

为什么要讨论这个问题？

- 一切关系个人职业发展的问题都值得讨论

到底会不会被开除?

- select * from table where name like '%德哥';
 - 100万条记录(1.1GB): 300毫秒.
 - 1000万条记录(11GB): 3500毫秒.
 - 1亿条记录(110GB): 85秒.
- 32 Core 的数据库, 32个并发足以引起**数据库雪崩**.
 - 影响业务是必须的!!!**
 - 看老板心情、 资损!!!**

问题在哪？

- 没有创建索引!!!

原因是什么？

- DBA背锅，你咋不创建索引!!!
- DBA： 你见过哪个数据库的索引支持模糊查询呀!!!

怎么破?

- 土豪出没:
 - 多创建几个只读实例, 顶多把只读实例搞崩溃而已.
- 砍需求、定数据库规范.
 - 不允许在数据库中执行模糊查询、全文检索.
 - 砍产品经理就是砍客户.
- **砍客户就是把客户推向竞争对手的怀抱.**
- DBA: 需求领回家
 - “外事问x歌 内事问x度 糜事问x涯”



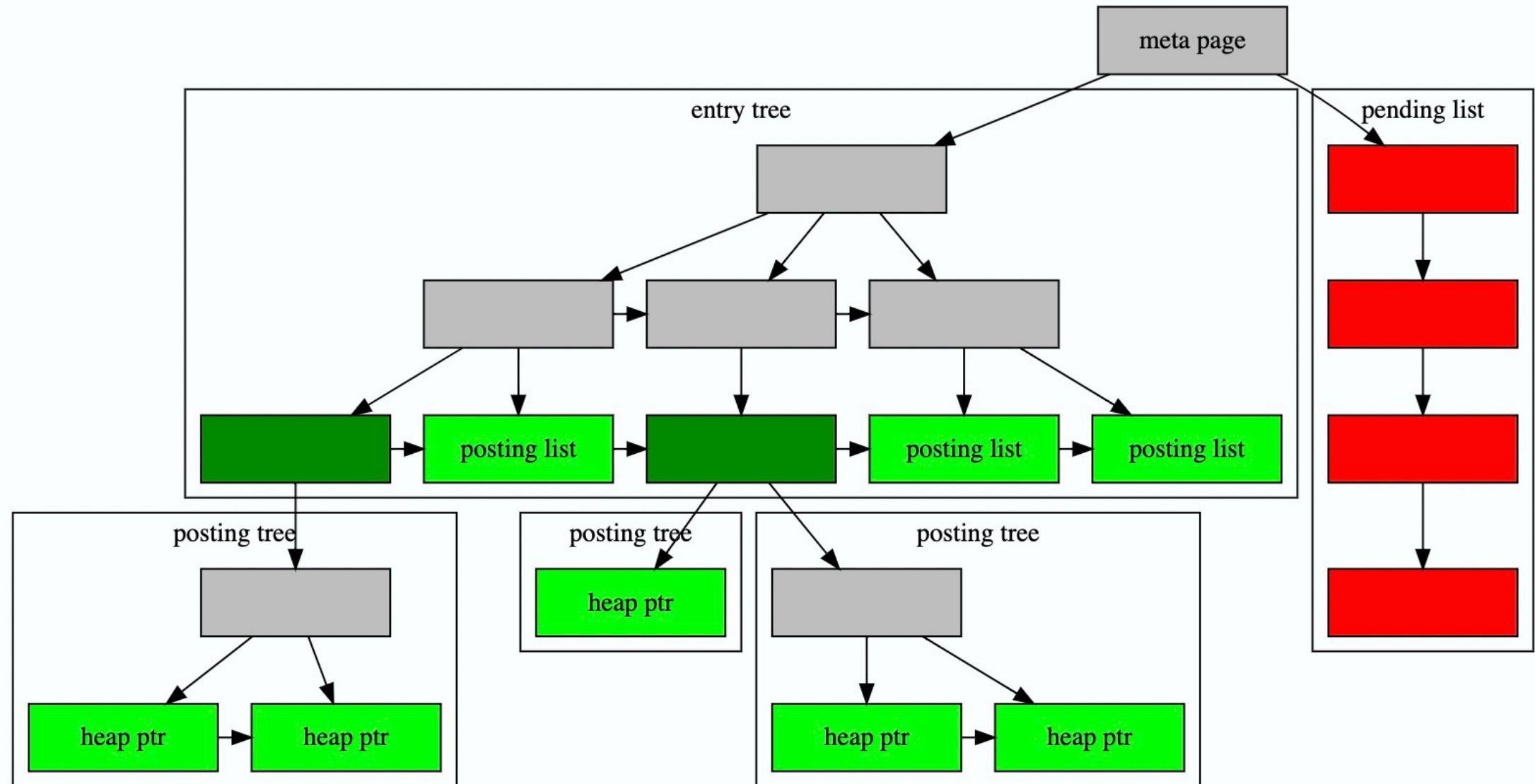
搜来的方案靠谱吗?

- 恩, 他们都是这样用的:
- 需要搜索的数据, 一份写入关系数据库, 再同步一份同步到搜索引擎
- 靠谱吗?
 - 同步延迟, 数据写入后无法实时被搜索到.
 - 跨产品同步引入的一致性问题, 每天刷一遍全量再继续增量同步.
- 一致性与查询时延要求高的场景用这个方案显然不行.

什么才是经过思考的牛逼方案？

- 如果要设计一款数据库索引来支持全文检索、模糊查询甚至正则表达式，应该如何设计？什么指标是重要的？
 - 索引构建的实时性，
 - 查询的实时性，
 - 查询性能，
 - 内容写入、变更性能，
 - 可以按相似度排序返回，
 - 全文检索：分词正确性，（全文检索不讨论，PG已经内置，安装各国语言插件可以实现索引加速，同时支持扩展字典。）
 - 全文检索：字典可自定义，
- **统统安排!!! 隆重推出真香数据库!!!**
- 有一款数据库叫PG，有个倒排索引接口GIN，有一个模块叫pg_trgm

Figure 66.1. GIN Internals



有了树，还需要词，把词填进树里。
怎么把字符串变成可以搜索到词？

```
postgres=# select show_trgm('hello, digoal');
            show_trgm
-----
 {" d"," h"," di"," he","al ",dig,ell,goa,hel,igo,llo,"lo,","o, ",oal}
(1 row)

postgres=# select show_bigm('hello, digoal');
            show_bigm
-----
 {" d"," h",""," ,al,di,el,go,he,ig,"l ",ll,lo,"o,","oa"
(1 row)
```

```
postgres=# select show_trgm('你好，这里是PG周末大讲堂');
            show_trgm
-----
 {0x827744,0xaf9d36,0xd6fac8,0xe435c6,0xf2970b,0xf98da8,0xfd571f,0x06aa92,0x0df9aa,0x359588,IgR,0x58007a,0x59de5d,0x64e84a,0x681e3f}
(1 row)

postgres=# select show_bigm('你好，这里是PG周末大讲堂');
            show_bigm
-----
 {"你好,周末,"堂 ",大讲,"好,",是P,末大,讲堂,这里,里是," 你"," 这",",", "G周,PG}
(1 row)
```

练一练： 1280万 模糊查询 要多久？

```
db1=# create or replace function gen_hanzi(int) returns text as $$  
db1$# declare  
db1$#   res text;  
db1$# begin  
db1$#   if $1 >=1 then  
db1$#     select string_agg(chr(19968+(random()*20901)::int), '') into res from generate_series(1,$1);  
db1$#     return res;  
db1$#   end if;  
db1$#   return null;  
db1$# end;  
db1$# $$ language plpgsql strict;  
CREATE FUNCTION  
db1=# create table test001(c1 text);  
CREATE TABLE  
db1=# insert into test001 select gen_hanzi(20) from generate_series(1,100000);  
INSERT 0 100000  
db1=# insert into test001 select * from test001;  
INSERT 0 100000  
db1=# insert into test001 select * from test001;  
INSERT 0 200000  
db1=# insert into test001 select * from test001;  
INSERT 0 400000  
db1=# insert into test001 select * from test001;  
INSERT 0 800000  
^[[A  
db1=# create index idx_test001_1 on test001 using gin (c1 gin_trgm_ops);  
CREATE INDEX  
db1=# explain (analyze,verbose,timing,costs,buffers) select * from test001 where c1 like '%德哥是PGer%';  
                                         QUERY PLAN  
-----  
Bitmap Heap Scan on public.test001  (cost=44.02..1450.58 rows=1280 width=61) (actual time=0.041..0.041 rows=0 loops=1)  
  Output: c1  
  Recheck Cond: (test001.c1 ~ '%德哥是PGer%':text)  
  Buffers: shared hit=21  
-> Bitmap Index Scan on idx_test001_1  (cost=0.00..43.70 rows=1280 width=0) (actual time=0.039..0.039 rows=0 loops=1)  
    Index Cond: (test001.c1 ~ '%德哥是PGer%':text)  
    Buffers: shared hit=21  
Planning Time: 0.507 ms  
  Buffers: shared hit=26 dirtied=5  
Execution Time: 0.059 ms  
(10 rows)
```

没错，性能提升59000倍

- 详细用法参考：
 - https://github.com/digoal/blog/blob/master/201704/20170426_01.md

苏格拉底说：怀疑一切!!!

牛逼的方案就没漏洞吗？

- pg_trgm真香方案能有啥问题?
 - pg_trgm采用3-grams切分粒度, 1个或2个字的模糊查询性能很差.
 - 当匹配结果非常非常多时, 即时LIMIT返回依旧有较大启动成本. bitmap scan造成.
 - 在开启fastupdate的情况下, 优先将数据写入pending list, autovacuum异步合并到gin树. 查询时需要查询pending list以及gin索引树, 会导致搜索性能降低.
 - 特别是在大量数据高并发写入后, 全文检索、模糊查询的性能都会下降. pending list 合并到gin树后性能恢复.
 - lc_ctype=C时无法切分wchar(有解,只是善意的提醒你别踩坑).
 - 例如中文模糊查询, 千万要注意, 别掉坑里.
- 详见
 - https://github.com/digoal/blog/blob/master/202009/20200912_01.md

更牛逼的方案是什么? – MyBase PG提前想到了

pg_bigm与pg_trgm对比

MyBase PG
pg_bigm

以上都不是问题

| 功能与特性 | pg_trgm | pg_bigm |
|-------------------------|--|--|
| 切词粒度 | 前加2后加1个空格, 每3个连续字 | 前加2后加1个空格, 每2个连续字 |
| 索引接口 | gin, gist | gin |
| 支持的语法 | LIKE (~~), ILIKE (~~*), ~, ~* | LIKE only |
| wchar支持情况 | lc_ctype <> C 或者 编译时注释 contrib/pg_trgm/trgm.h KEEPONLYALNUM macro定义, 否则不支持切分wchar的token | 任何时候都支持wchar |
| 1或2个字的模糊查询 | 慢(因为切词粒度为3个字, 所以无法匹配), 必须全表扫描, 或者full index scan+RECHECK | 快 |
| 相似查询 | 支持 | 支持 |
| like封装函数 | 不支持 | 支持 |
| 最大索引字段长度 | 228MB | 102MB |
| 高性能开关(关闭recheck, 不严谨查询) | 不支持 | 支持 pg_bigm.enable_recheck=off |
| 高性能开关(粗查, 不严谨查询) | 不支持 | 支持 pg_bigm.gin_key_limit 只查少量token (2-grams) |

如何学会保护自己和公司业务?

- 1、防止雪崩: 前端保护, 避免重复请求
 - 防止在后端响应慢时, 前端用户不断点击, 导致雪崩.
- 2、防止雪崩: 降级、疏导
 - 设置语句超时, 避免慢SQL击破数据库资源.
- 3、规范
 - 一切关系个人职业发展的问题都值得放到规范里面
- 4、以上够了吗? 你还需要找个强大的(背)后(锅)盾(侠).
 - 阿里云AliPG, 全兼容PostgreSQL, 2015年投入商用, 数万用户的坚强后盾.
 - **更重要的是: 拥有代码能力的后盾、不仅是PG内核代码, 还有插件代码~~~**
 - **MyBase PG已集成pg_trgm, pg_bigm等高级模块.**

憋废话了，到底哪里有 MyBase？

- 入钉钉群，咨询砖家
- **免费试用** 活动进行中!!!

