在数据库中跑全文检索、模糊查询SQL会不会被开除?

Digoal

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 糗事问x涯”

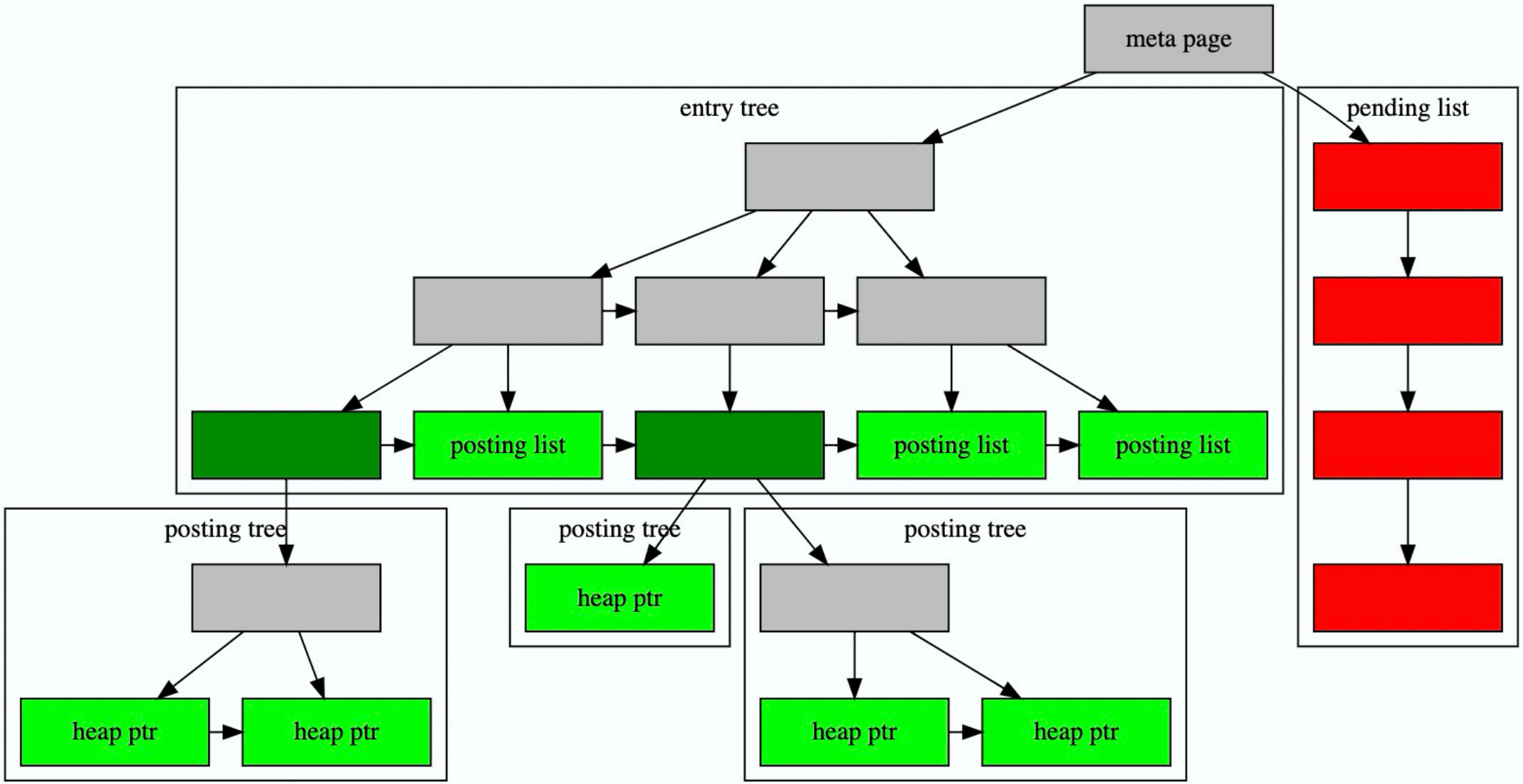
搜来的方案靠谱吗?

- 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!!! 真香!!!**
- **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)
```

牛逼的方案就没漏洞吗?

- pg_trgm真香方案会有什么问题?
  - 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pengding 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_ctyp=C时无法切分wchar.
    - 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已集成

- pg_bigm

以上都不是问题

pg_bigm与pg_trgm对比

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, 别让一种慢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、不仅仅是PG内核代码, 还有插件代码~~~**
    - **MyBase PG已集成pg_trgm, pg_bigm等模块.**

憋废话了，到底哪里有MyBase?

- 详细方案入钉钉群，咨询砖家