

导购网站应用案例

PostgreSQL海量数组实时相似计算

digoal

阿里云

业务介绍

平板电脑购买指南_如何选择平板电脑



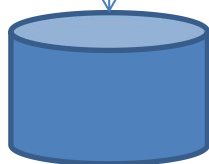
导购文1



导购文n



审核
去重(相似度)



数据库

[摘要] 平板电脑就是一款无须翻盖、没有键盘、小到可以放入场上就有各式各样的“平板大军”，而且高、中、低档俱全，电脑呢？我们一起来了解了解。

这几天华为P10、三星S8的发售，无疑在手机界引发了一就会发现，随着市场的饱和，近几年来，手机在技术方面色、拼外观、拼拍照.....一个小小的像素的提升，都已经开

苹果iPad推荐

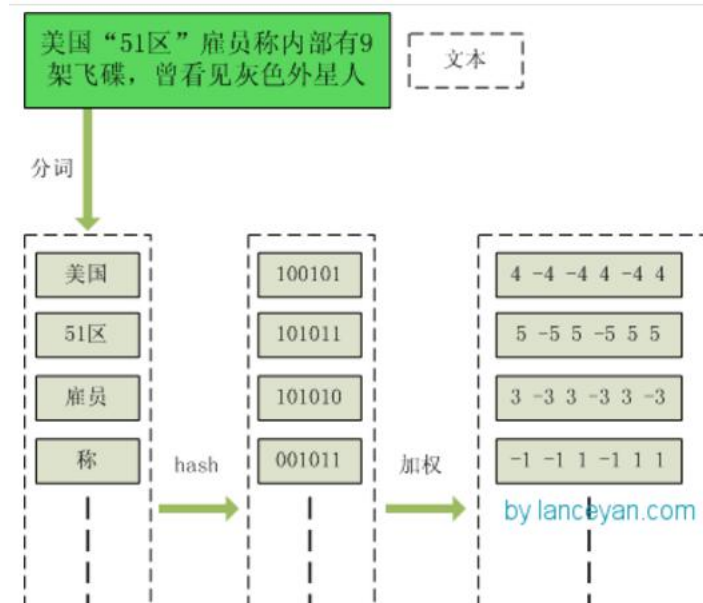


导购文章有效性判定要素

- 商品重复率
- 实时

相关技术

- 文本相似
 - 机器学习
 - 关键次提取
 - 海明距离
- 图像相似
 - haar wavelet
- 数组相似
 - PostgreSQL smlar



Input image ($2^N \times 2^N$)



Transformed image ($2^N \times 2^N$)

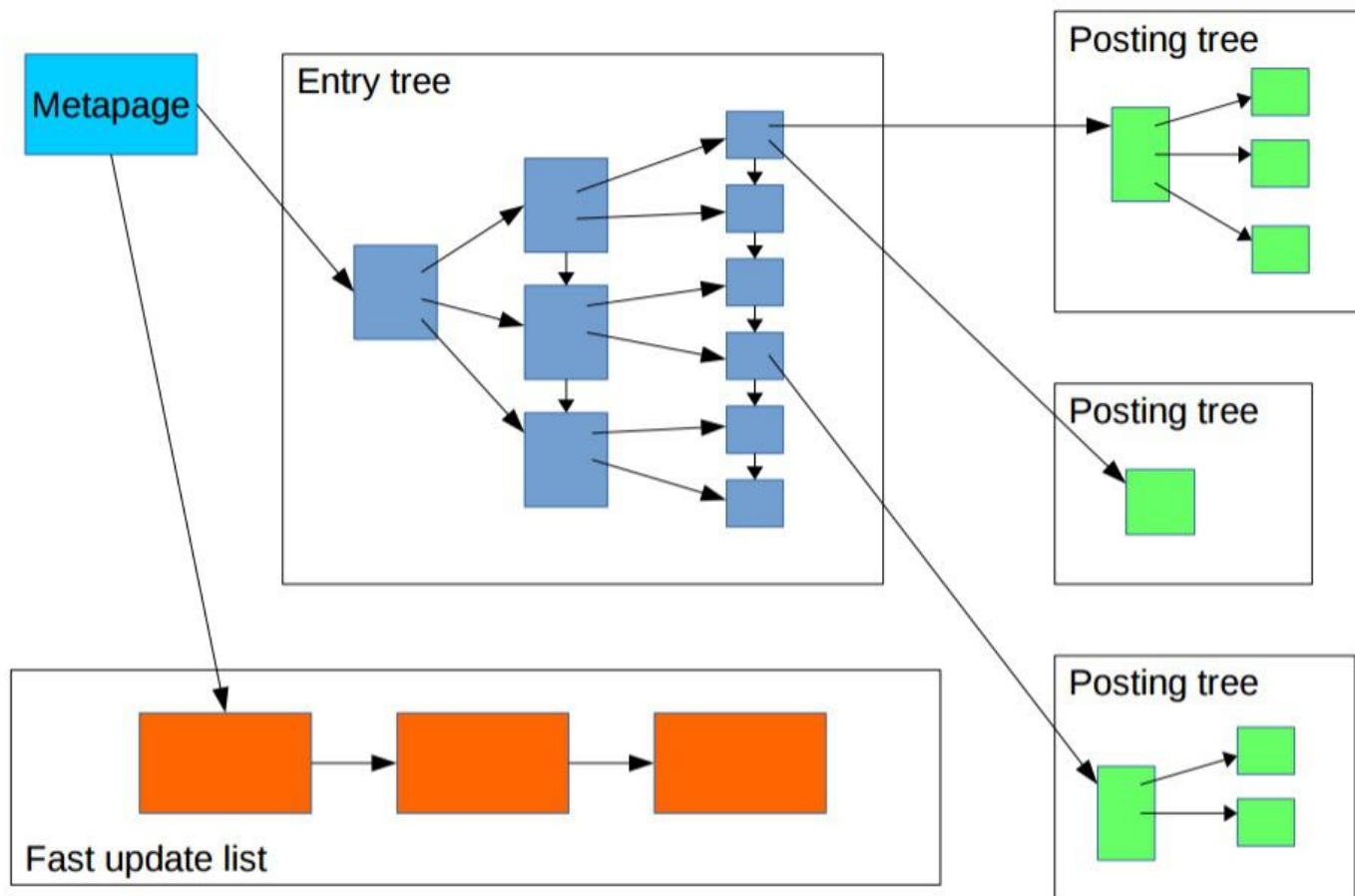


smlar

- 相似度公式
 - cosine
 - $N.i / \sqrt{N.a * N.b}$
 - overlap
 - $N.i$
 - tfidf
 - $N.i$: 相交的元素个数(被比较的两数组会先去重)
 - $N.a$: 第一个数组的元素个数(去重后)
 - $N.b$: 第二个数组的元素个数(去重后)

GIN索引结构

Complete GIN



GIN索引

- 倒排效果
- postgres=# select id, count(*) from tbl group by 1 order by 1;
- id | count
- ----+-----
- 0 | 10096
- 1 | 91
- 2 | 97
- 3 | 92
- 4 | 109
- 5 | 108
- 6 | 97
- 7 | 100
- 8 | 103
- 9 | 88
- 10 | 89
- 11 | 99
- 12 | 91

GIN索引

- 数组，GIN索引，倒排效果
- postgres=# select info from arr ;
- info
- -----
- {9632,6798,2069,3533,2702,3191,5561,8756,4391,3290}
- {4179,8012,6926,1643,5025,8958,8984,2823,8730,3273}
- {4226,4074,9788,8491,8941,3609,5942,8588,5395,307}
- {2632,5028,7105,4702,8561,9807,7893,4121,8563,2284}
- {7412,2743,296,4337,4386,5320,3295,3370,8143,2026}
- {6643,2369,6100,6431,860,5041,40,6803,3628,5435}

GIN索引

- 数组，GIN索引，倒排效果
- postgres=# select ctid,unnest(info) from arr;
- ctid | unnest
- -----+-----
- (0,1) | 9632
- (0,1) | 6798
- (0,1) | 2069
- (0,1) | 3533
- (0,1) | 2702
- (0,1) | 3191
- (0,1) | 5561
- (0,1) | 8756
- (0,1) | 4391
- (0,1) | 3290
- (0,2) | 4179
- (0,2) | 8012
- (0,2) | 6926
-

GIN索引

- 数组，GIN索引，倒排效果
- postgres=# select id,array_agg(ctid) from (select unnest(info) id,ctid from arr) t group by 1 order by 1;
- ...
- 1491 | {"(0,39)","(1,3)"}
- 1496 | {"(0,18)"}
- 1500 | {"(0,23)","(0,31)","(0,76)"}
- ...

GIN索引, OVERLAP相似计算检索举例

A handwritten GIN index matrix on a chalkboard. The matrix has columns for 'Heap ID' (1, 35, 44, ..., 101, 109, 1000) and rows for 'key' (1, 3, 101, 198, 798, 1000). The cells contain binary values (0 or 1) indicating the presence of a key on a page. Below the matrix, there are numbers representing the overlap counts for each key across the pages.

key	1	35	44	...	101	109	1000
1	1	0	0	⋮	1	0	0
3	0	0	1	⋮	0	1	1
101	0	0	1	⋮	1	0	0
198	0	1	0	⋮	1	1	0
798	1	0	0	⋮	1	0	1
1000	0	0	1	⋮	0	0	0
	2	1	3	...	4	2	2

第一重过滤

- 收敛 BLOCK ID (if overlap ≥ 3)

Heap ID	key	Page	1	35	49	101	109	1000
1	1		1	0	0	⋮	1	0	0
3	3		0	0	1	⋮	0	1	1
101	101		0	0	1	⋮	1	0	0
198	198		0	1	0	⋮	1	1	0
798	798		1	0	0	⋮	1	0	1
1000	1000		0	0	1	⋮	0	0	0
			2	1	3	4	2	2

第二重过滤

- CPU CHECK BLOCK's tuples
- BLOCK: 49
- BLOCK: 101

导购推荐平台-例子

- 沉淀导购文章：6000万
- 涉及商品数量：1000万
- 平均每篇文章涉及商品数量：11 - 50个商品
- 热点商品：50万热点商品，商品ID分布区域1-50万，热点商品被1000万篇文章推荐过。

导购推荐平台-例子

- [测试方法，如何造数据？](#)
- https://github.com/digoal/blog/blob/master/201701/20170112_02.md
- create extension smlar;
- create unlogged table test (id serial, -- 文章ID
- arr int8[] -- 商品ID组成的数组，假设商品ID为int8类型，那么数组就是int8[]);
- 插入5000万记录，要求如下
- int8 取值范围1~1000万，即历史上被推荐的商品有1000万个。
- int8[] 数组长度 11 ~ 50，即每篇导购文章，包含11到50个商品。
- 调用一次插入40条记录。
- create or replace function f() returns void as \$\$
- declare begin
- for i in 11..50 loop
- insert into test (arr) select array_agg((10000000*random())::int8) from generate_series(1,i);
- end loop;
- end; \$\$ language plpgsql strict;

总共生成6000万历史导购数据

- 使用pgbench调用以上函数，将生成5000万测试数据
- `vi test.sql select f(); pgbench -M prepared -n -r -P 1 -f ./test.sql -c 100 -j 100 -t 12500`
- 生成1000万热点商品的推荐数据
- 假设商品ID范围在 1 ~ 50万 的为热点商品，被1000万篇文章推荐过。
- `create or replace function f() returns void as $$ declare begin`
- `for i in 1..50 loop`
- `insert into test (arr) select array_agg((500000*random())::int8) from generate_series(1,i);`
- `end loop; end; $$ language plpgsql strict;`
- 使用pgbench调用以上函数，生成1000万热点数据
- `pgbench -M prepared -n -r -P 1 -f ./test.sql -c 100 -j 100 -t 2500`

创建GIN索引

- `set maintenance_work_mem='64GB';`
- `create index on test using gin (arr_int8_sml_ops);`
- 虽然smlar插件还支持 gist 索引，但是本文的CASE不建议使用gist索引
- `-- create index on test using gist (arr_int8_sml_ops);`

性能?

- 审核性能
- 导购文章包含普通商品40个，其中39个与历史导购文章重复。
 - 4ms
- 导购文章包含普通商品40个，其中20个与历史导购文章重复。
 - 4ms
- 导购文章包含热点商品10个,普通商品30个，其中39个与历史导购文章重复。
 - 6ms
- 导购文章包含热点商品10个,普通商品30个，其中20个与历史导购文章重复。
 - 6ms
- 导购文章包含热点商品40个，其中39个与历史导购文章重复。
 - 15ms
- 导购文章包含热点商品40个，其中20个与历史导购文章重复。
 - 15ms

性能?

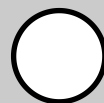
- 审核压测
- 普通商品35个，热点商品5个， overlap=35
- tps = 9455.190803 (including connections establishing)
- tps = 9460.748975 (excluding connections establishing)

小结



传统方法

全表扫描匹配相似度，
效率低，资源消耗巨大。
审核延迟非常高，无法
实时。



PostgreSQL GIN+smlar插 件。

效率高，资源消耗低。
实时判定数组相似度，
(实时审核导购内容)。

谢谢

- 详情
- https://github.com/digoal/blog/blob/master/201701/20170112_02.md
- 个人GIT
- <https://github.com/digoal/blog/blob/master/README.md>



官方微信公众号



digoal's 微信