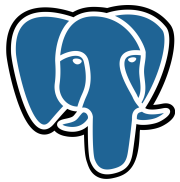
高级SQL提升生产力



阿里云
digoal

目录

- 物化视图
- 实时清洗数据
- 数据采样
- 数据加密
- 数据脱敏
- 数据去重
- 字段范围加速查询
- 行级安全
- WITH ORDINALITY
- LATERAL
- GROUPING SETS, CUBE and ROLLUP , grouping
- 窗口
- cte，递归
- skip locked
- distinct on
- 秒杀
- 批量sql
- 数据分析
- 流计算
- rule、异步消息
- 线性回归、预测
- 化学分析
- 机器学习
- update|delete limit
- prepare 动态sql in function

物化视图

- https://www.postgresql.org/docs/devel/static/sql-creatematerializedview.html
- **预计算**，支持索引

```
CREATE MATERIALIZED VIEW [ IF NOT EXISTS ] table_name
    [ (column_name [, ...] ) ]
    [ WITH ( storage_parameter [= value] [, ... ] ) ]
    [ TABLESPACE tablespace_name ]
    AS query
    [ WITH [ NO ] DATA ]
```

- 刷新物化视图

```
REFRESH MATERIALIZED VIEW [ CONCURRENTLY ] name
    [ WITH [ NO ] DATA ]
```

实时数据清洗、转换

- https://github.com/digoal/blog/blob/master/201706/20170619_02.md
- rule
- 创建来源表结构

  postgres=# create table nt(id int, **c1 numeric, c2 numeric**);

  CREATE TABLE
- 创建目标表结构

  postgres=# create table nt_geo (id int, **geo geometry**);

  CREATE TABLE
- 对来源表创建规则或触发器，例如

  postgres=# create rule r1 as on insert to nt do instead insert into nt_geo values (NEW.id, ST_MakePoint(NEW.c1,NEW.c2));

  CREATE RULE
- 使用来源数据结构，将数据插入来源数据表

  postgres=# insert into nt values (1,1,1);

  INSERT 0 1

实时数据清洗、转换

- rule
- 源表，JSONB非结构化

  postgres=# create table t1 (id int, info text, j jsonb);
- 目标表，结构化

  postgres=# create table t2 (id int, info text, c1 int, c2 int, c3 text);
- 在源表创建规则，自动将JSONB非结构化数据，转换为结构化数据插入

  postgres=# create rule r1 as on insert to t1 do instead insert into t2 values (NEW.ID, NEW.INFO, ((NEW.J)->>'c1')::int, ((NEW.j)->>'c2')::int, (NEW.j)->>'c3');

  postgres=# insert into t1 values (1,'test',**jsonb '{"c1":1, "c2":2, "c3":"text"}'**);

  postgres=# select * from t1;

  (0 rows)

  postgres=# select * from t2;

  id | info | c1 | c2 | c3

  ----+------+----+----+------

  1 | test | 1 | 2 | text

  (1 row)

数据采样

- 使用采样算法
  - 行级随机采样(BERNOULLI(百分比))

    select * from test TABLESAMPLE bernoulli (1);
  - 块级随机采样(SYSTEM(百分比))

    select * from test TABLESAMPLE system (1);
- https://github.com/digoal/blog/blob/master/201706/20170602_02.md
  - 采样应用：估值计算、统计信息、测试环境
- https://github.com/digoal/blog/blob/master/201709/20170911_02.md

数据加密

- pgcrypto
  - https://www.postgresql.org/docs/10/static/pgcrypto.html
- 加密后的查询加速（等值查询）
- 加密
  - 对称、非对称、混淆（使用非对称加密需要交换的对称加密密钥，然后使用对称加密）加密
  - https://github.com/digoal/blog/blob/master/201802/20180226_01.md

字段加密

digoal=# create extension **pgcrypto;**

- 可逆加密

  digoal=# insert into userpwd (userid,pwd) values (1, crypt('this is a pwd source', gen_salt('bf',10)));

  digoal=# create table userpwd(userid int8 primary key, pwd text);

  CREATE TABLE

- 不可逆加密
  - https://github.com/digoal/blog/blob/master/201607/20160727_02.md
  - https://github.com/digoal/blog/blob/master/201711/20171127_02.md

数据脱敏

- https://github.com/digoal/blog/blob/master/201706/20170602_02.md

数据去重

- https://github.com/digoal/blog/blob/master/201706/20170602_01.md
  - **单列去重**
  - **多列去重**
  - **行去重**
  - **多列混合去重**
- 窗口、行号、= any(array())、数组

去重最常用SQL

```
delete from tbl where ctid = any (array (
  select ctid from (select ctid,row_number() over (partition by c1 order by ts desc) as rn from tbl) t where rn<>1
));
```

多字段范围检索加速

- 范围查询，联合索引效果并不好。
- 使用range类型，GIST索引加速。(或者第11课讲到的cube)
  - 开始时间，结束时间范围
  - 经纬度范围
  - DNS匹配（开始IP，结束IP 范围）
    - https://github.com/digoal/blog/blob/master/201206/20120607_01.md

VPD(RLS)

- https://github.com/digoal/blog/blob/master/201602/20160203_03.md
- https://github.com/digoal/blog/blob/master/201504/20150409_01.md
- https://www.postgresql.org/docs/11/sql-createpolicy.html

```
create table test (id int, r name, info text, crt_time timestamp);
insert into test values (1,'digoal','test',now());
insert into test values (2,'r3','test123',now());
create role r3 login;
grant select on test to r3;
alter table test enable row level security ;
\c postgres r3
postgres=> explain verbose select * from test;
                        QUERY PLAN
------------------------------------------------------------
 Seq Scan on public.test  (cost=0.00..19.45 rows=3 width=100)
   Output: id, r, info
   Filter: (test.r = CURRENT_USER)
(3 rows)
```

创建三个角色

```
postgres=# create role r1 login;
CREATE ROLE
postgres=# create role r2 login;
CREATE ROLE
postgres=# create role r3 login;
CREATE ROLE
```

创建测试表

```
postgres=# create table test(id int, r name);
CREATE TABLE
postgres=# insert into test values(1, 'r1');
INSERT 0 1
postgres=# insert into test values(2, 'r2');
INSERT 0 1
postgres=# insert into test values(3, 'r3');
INSERT 0 1
postgres=# grant all on table test to public;
GRANT
```

创建一个新增数据的策略(使用with check)

```
postgres=# create policy p on test for insert to r1 with check( r = current_user);
CREATE POLICY
```

默认情况下策略是disable状态的,

```
postgres=# \d+ test
                          Table "public.test"
 Column |  Type   | Modifiers | Storage | Stats target | Description
--------+---------+-----------+---------+--------------+-------------
 id     | integer |           | plain   |              |
 r      | name    |           | plain   |              |
Policies (Row Security Disabled):
    POLICY "p" FOR INSERT
      TO r1
      WITH CHECK (r = "current_user"())
```

通过pg_policies视图可以查看已经创建的策略.

```
postgres=# select * from pg_policies ;
 schemaname | tablename | policyname | roles |  cmd   | qual |       with_check
------------+-----------+------------+-------+--------+------+------------------------
 public     | test      | p          | {r1}  | INSERT |      | (r = "current_user"())
(1 row)
```

在策略enable前, 是无视策略的.

```
postgres=> insert into test values(4,'r1');
INSERT 0 1
postgres=> insert into test values(4,'r2');
INSERT 0 1
```

使策略生效

```
postgres=# alter table test enable row level security;
ALTER TABLE
postgres=> \d+ test
                          Table "public.test"
 Column |  Type   | Modifiers | Storage | Stats target | Description
--------+---------+-----------+---------+--------------+-------------
 id     | integer |           | plain   |              |
 r      | name    |           | plain   |              |
Policies:
    POLICY "p" FOR INSERT
      TO r1
      WITH CHECK (r = "current_user"())
```

现在策略生效了,再次插入,你会看到只能插入和r1角色同名的r值.

```
postgres=# \c postgres r1
You are now connected to database "postgres" as user "r1".
postgres=> insert into test values(4,'r2');
ERROR:  new row violates WITH CHECK OPTION for "test"
postgres=> insert into test values(4,'r1');
INSERT 0 1
```

目录

- WITH ORDINALITY
- LATERAL
- GROUPING SETS, CUBE and ROLLUP , grouping
- 窗口
- cte，递归
- skip locked
- distinct on
- 秒杀
- 批量sql
- 数据分析
- 流计算
- rule、异步消息
- 线性回归、预测
- 化学分析
- 机器学习
- update|delete limit
- prepare 动态sql in function

https://github.com/digoal/blog/blob/master/201307/20130730_01.md

```
db1=> select * from unnest(array[100,200,111,333])  with ordinality as t (v,i);   -- srf函数的行号输出
  v  | i
-----+---
 100 | 1
 200 | 2
 111 | 3
 333 | 4
(4 rows)
```

```
db1=> select t1.i, t1.v, t2.v from
(select * from unnest(array[100,200,111,333]) with ordinality as t (v,i) ) t1
join
(select * from unnest(array['b','c','d','e']) with ordinality as t (v,i) ) t2
using (i);
 i |  v  | v
---+-----+---
 1 | 100 | b
 2 | 200 | c
 3 | 111 | d
 4 | 333 | e
(4 rows)
```

```
create table a (
sid int,   -- 传感器ID
ts timestamp[],  -- 时间戳数组
s1_val float4[],   -- 维度1值数组
s2_val float4[],   -- 维度2值数组
tsr tsrange  -- 范围
);


db1=# select unnest(array[100,200,111,333]) , unnest(array['b','c','d','e']) ;
 unnest | unnest
--------+--------
    100 | b
    200 | c
    111 | d
    333 | e
(4 rows)
```

- db1=# select unnest(array[100,200,111,333]) , unnest(array['b','c','d','e','f']) ;
- unnest | unnest
- --------+--------
- 100 | b
- 200 | c
- 111 | d
- 333 | e
- | f
- (5 rows)

目录

- WITH ORDINALITY
- LATERAL
- GROUPING SETS, CUBE and ROLLUP , grouping
- 窗口
- cte，递归
- skip locked
- distinct on
- 秒杀
- 批量sql
- 数据分析
- 流计算
- rule、异步消息
- 线性回归、预测
- 化学分析
- 机器学习
- update|delete limit
- prepare 动态sql in function

- https://github.com/digoal/blog/blob/master/201210/20121008_01.md

```
SELECT * FROM foo, LATERAL (SELECT * FROM bar WHERE bar.id = foo.bar_id) ss;

在 LATERAL (这里可以关联(引用)lateral左边的表或子句)

所以允许:

LATERAL (SELECT * FROM bar WHERE bar.id = foo.bar_id)
```

目录

- WITH ORDINALITY
- LATERAL
- GROUPING SETS, CUBE and ROLLUP , grouping
- 窗口
- cte，递归
- skip locked
- distinct on
- 秒杀
- 批量sql
- 数据分析
- 流计算
- rule、异步消息
- 线性回归、预测
- 化学分析
- 机器学习
- update|delete limit
- prepare 动态sql in function

多维度透视，一条sql搞定

- https://github.com/digoal/blog/blob/master/201505/20150526_02.md

```
=> SELECT * FROM items_sold;
 brand | size | sales
-------+------+-------
 Foo   | L    |  10
 Foo   | M    |  20
 Bar   | M    |  15
 Bar   | L    |  5
(4 rows)

=> SELECT brand, size, sum(sales) FROM items_sold GROUP BY GROUPING SETS ((brand), (size), ());
 brand | size | sum
-------+------+-----
 Foo   |      |  30
 Bar   |      |  20
       | L    |  15
       | M    |  35
       |      |  50
(5 rows)
```

其中GROUP BY GROUPING SETS ((brand), (size), ());

相当于以下三个group by的union all：

```
group by brand
group by size
group by ()
```

分组集合除了可以用GROUPING SETS来指定，另外还提供了两个特殊的写法rollup和cube.

```
ROLLUP ( e1, e2, e3, ... )
```

代表递减分组，一般用于异构结构的分组如国家，省份，城市，乡镇这样的结构查询。

逐级分组汇总结果，它相当于如下写法：

```
GROUPING SETS (
    ( e1, e2, e3, ... ),
    ...
    ( e1, e2 )
    ( e1 )
    ( )  -- 注意包含全集
)
```

还有一种写法是CUBE

```
CUBE ( a, b, c )
```

cube是任意组合，相当于：

```
GROUPING SETS (
    ( a, b, c ),
    ( a, b    ),
    ( a,    c ),
    ( a       ),
    (    b, c ),
    (    b    ),
    (       c ),
    (         )    -- 注意包含全集
)
```

在cube和rollup中使用括号可以将多个表达式作为单个表达式来处理：

```
ROLLUP ( a, (b,c), d )
```

递减，相当于

```
GROUPING SETS (
    ( a, b, c, d ),
    ( a, b, c    ),
    ( a          ),
    (            )
)
```

```
CUBE ( (a,b), (c,d) )
```

相当于:

```
GROUPING SETS (
    ( a, b, c, d ),
    ( a, b       ),
    (       c, d ),
    (            )
)
```

同时cube,rollup,grouping sets还可以混合使用：

```
GROUP BY a, CUBE(b,c), GROUPING SETS ((d), (e))
```

相当于：

```
GROUP BY GROUPING SETS (
  (a,b,c,d), (a,b,c,e),
  (a,b,d),   (a,b,e),
  (a,c,d),   (a,c,e),
  (a,d),     (a,e)
)
```

用grouping(cols)可以表示未参与聚合的表达式的比特位，并转换为INT输出。

例如：

```
=> SELECT * FROM items_sold;
 make  | model | sales
-------+-------+-------
 Foo   | GT    |  10
 Foo   | Tour  |  20
 Bar   | City  |  15
 Bar   | Sport |  5
(4 rows)
```

grouping()中必须包含group by后面的任意或所有列。

```
=> SELECT make, model, GROUPING(make,model), sum(sales) FROM items_sold GROUP BY ROLLUP(make,model);
 make  | model | grouping | sum
-------+-------+----------+-----
 Foo   | GT    |        0 | 10
 Foo   | Tour  |        0 | 20
 Bar   | City  |        0 | 15
 Bar   | Sport |        0 | 5
 Foo   |       |        1 | 30
 Bar   |       |        1 | 20
       |       |        3 | 50
(7 rows)
```

没有包含的column位置对应1，包含的column位置用0表示。

```
db1=# create table ta (c1 int, c2 int, c3 int);
CREATE TABLE
db1=# select c1, c2,c3,grouping(c1,c2,c3),count(*) from ta group by rollup(c1,c2,c3);
 c1 | c2 | c3 | grouping | count
----+----+----+----------+-------
    |    |    |        7 |     0
(1 row)

db1=# insert into ta values (1,2,3);
INSERT 0 1
db1=# insert into ta values (2,2,3);
INSERT 0 1
db1=# insert into ta values (3,2,3);
INSERT 0 1
db1=# insert into ta values (3,1,3);
INSERT 0 1
db1=# insert into ta values (3,1,5);
INSERT 0 1
db1=# select c1, c2,c3,grouping(c1,c2,c3),count(*) from ta group by rollup(c1,c2,c3);
 c1 | c2 | c3 | grouping | count
----+----+----+----------+-------
    |    |    |        7 |     5    111
  2 |  2 |  3 |        0 |     1
  3 |  1 |  5 |        0 |     1    000
  3 |  1 |  3 |        0 |     1
  1 |  2 |  3 |        0 |     1
  3 |  2 |  3 |        0 |     1
  2 |  2 |    |        1 |     1    001
  3 |  1 |    |        1 |     2
  3 |  2 |    |        1 |     1
  1 |  2 |    |        1 |     1    001
  3 |    |    |        3 |     3    011
  2 |    |    |        3 |     1
  1 |    |    |        3 |     1
```

group包含该列，bit设置为0
group不包含该列，bit设置为1

目录

- WITH ORDINALITY
- LATERAL
- GROUPING SETS, CUBE and ROLLUP , grouping
- 窗口
- cte，递归
- skip locked
- distinct on
- 秒杀
- 批量sql
- 数据分析
- 流计算
- rule、异步消息
- 线性回归、预测
- 化学分析
- 机器学习
- update|delete limit
- prepare 动态sql in function

- **窗口**
  - 窗口指计算的数据区间，指与当前row在同一个分组中的数据。
- **帧**
  - 帧也是指计算的数据区间，但是是在分组内的指定，可以根据当前row设定辐射半径，或者当前row的前后分开设置辐射范围。
- 窗口函数
- 帧函数
- https://www.postgresql.org/docs/current/functions-aggregate.html
- https://www.postgresql.org/docs/current/sql-expressions.html#SYNTAX-WINDOW-FUNCTIONS

窗口、帧查询

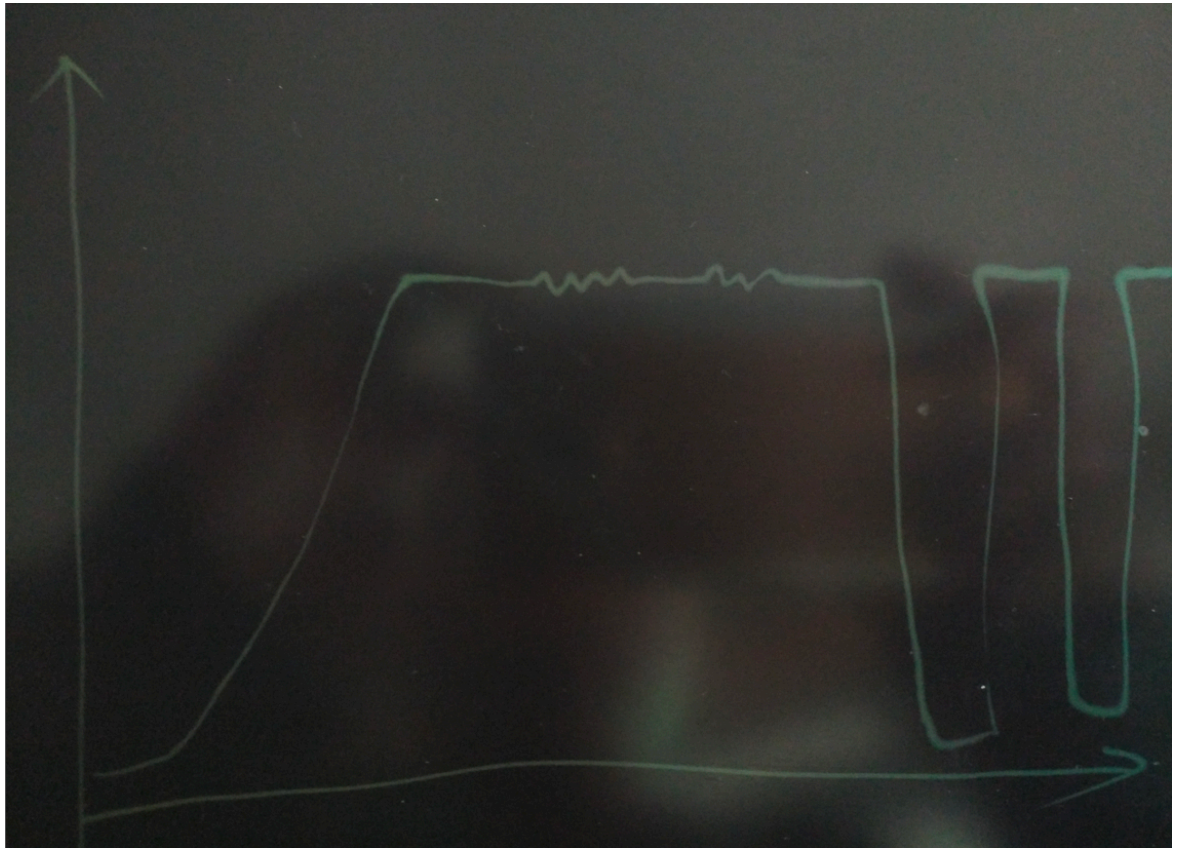
- 窗口查询语法
  - https://www.postgresql.org/docs/devel/static/sql-expressions.html#SYNTAX-WINDOW-FUNCTIONS
  - https://github.com/digoal/blog/blob/master/201802/20180224_01.md
- 窗口函数
  - https://www.postgresql.org/docs/devel/static/functions-window.html
- 聚合函数
  - https://www.postgresql.org/docs/devel/static/functions-aggregate.html

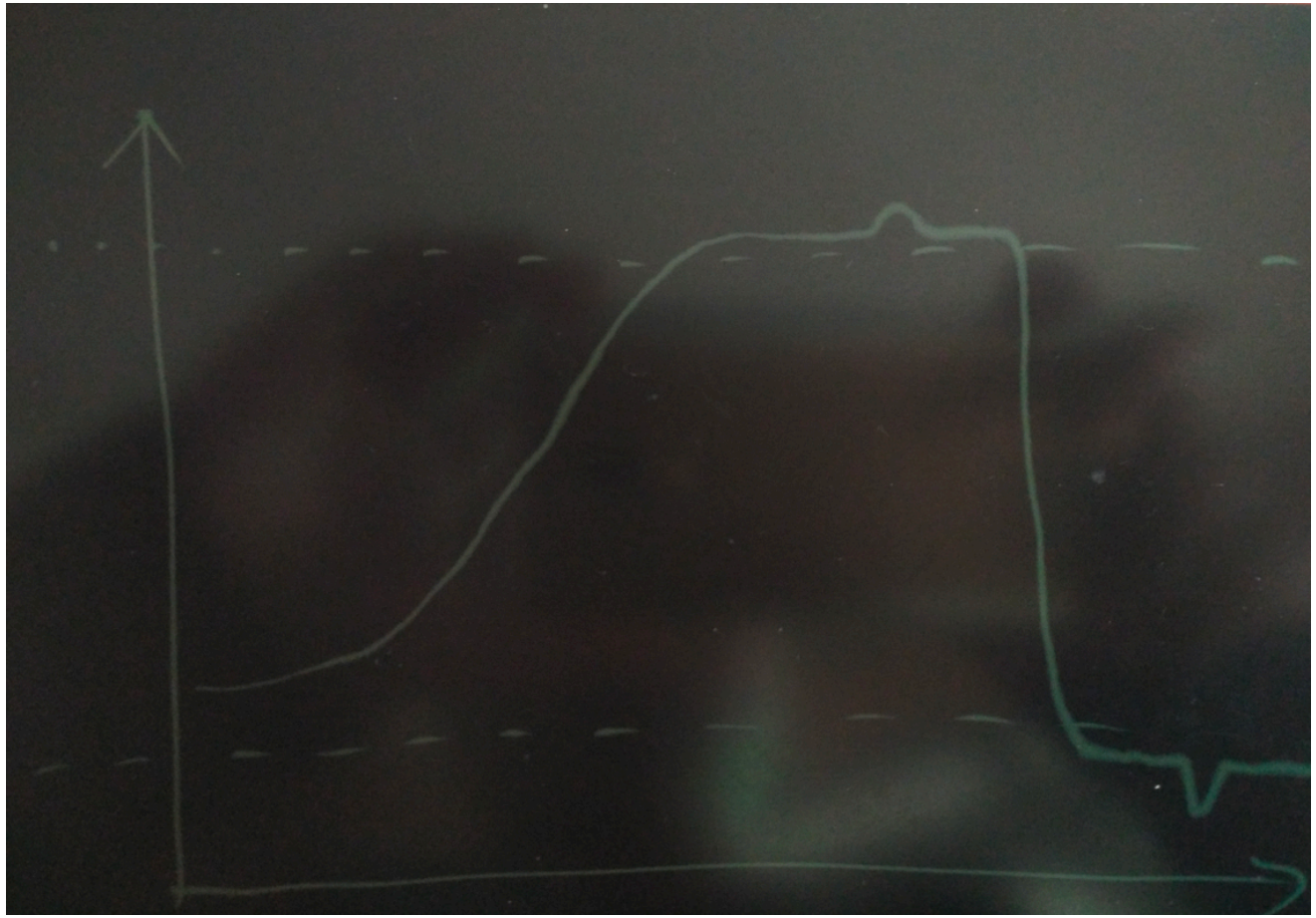
窗口、帧查询例子

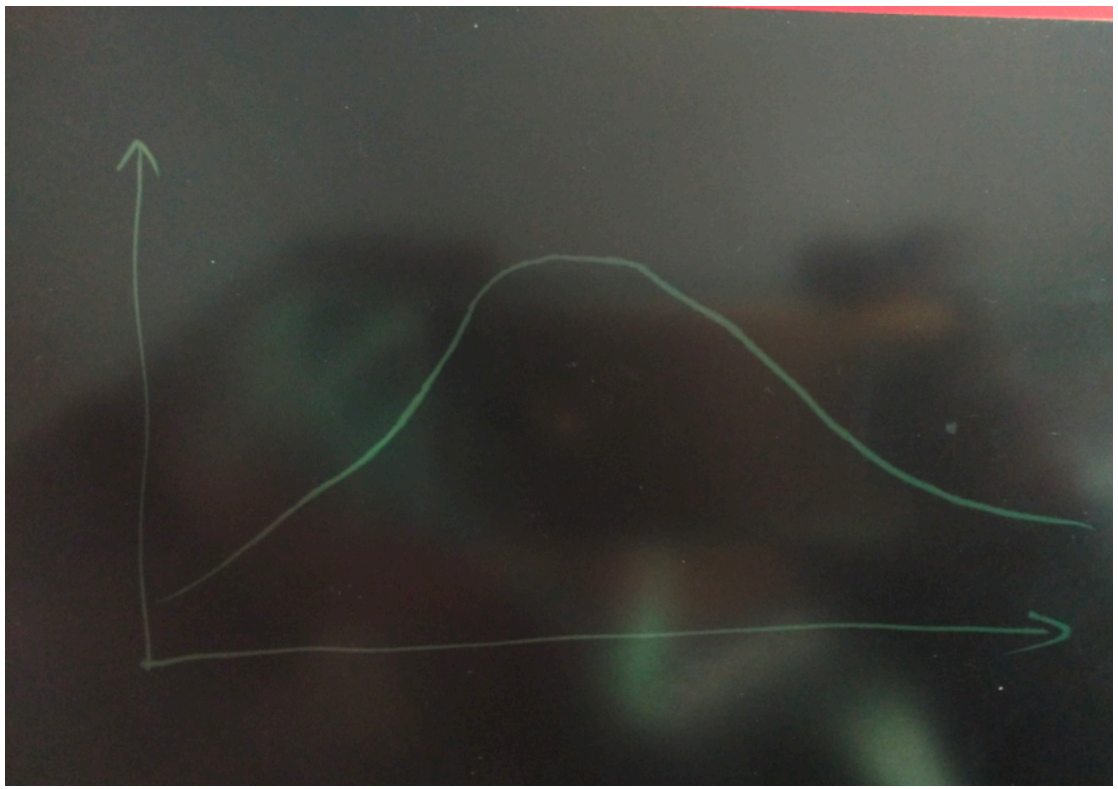
- 与第一名分差
  - select id, **first_value(score) over(partition by sub order by score desc)** - score, score, sub from t order by sub,id;
- 每门课程排名
  - select id,sub,score,**rank() over (partition by sub order by score desc)** from t order by sub,id;
- 滑窗分析-每条记录附近10(11)条记录的平均值
  - select id,class,score,**avg(score) over (partition by class rows between 5 preceding and 5 following)** from t order by class,id;
- 滑窗分析-每一天相比前一天的新增UV，最近7天新增UV
  - SELECT date, **(# hll_union_agg(users) OVER two_days) - (# lag(users) over (ORDER BY date ASC))** AS new_uniques FROM daily_uniques **WINDOW two_days AS (ORDER BY date ASC ROWS 1 PRECEDING);**
  - SELECT date, **# hll_union_agg(users) OVER seven_days** FROM daily_uniques **WINDOW seven_days AS (ORDER BY date ASC ROWS 6 PRECEDING)**;
- 数据去重
  - delete from tbl where ctid = any (array(
  - select ctid from
  - (select **ctid,row_number() over (partition by id order by crt_time desc) as rn** from tbl) t
  - where t.rn<>1
  -));

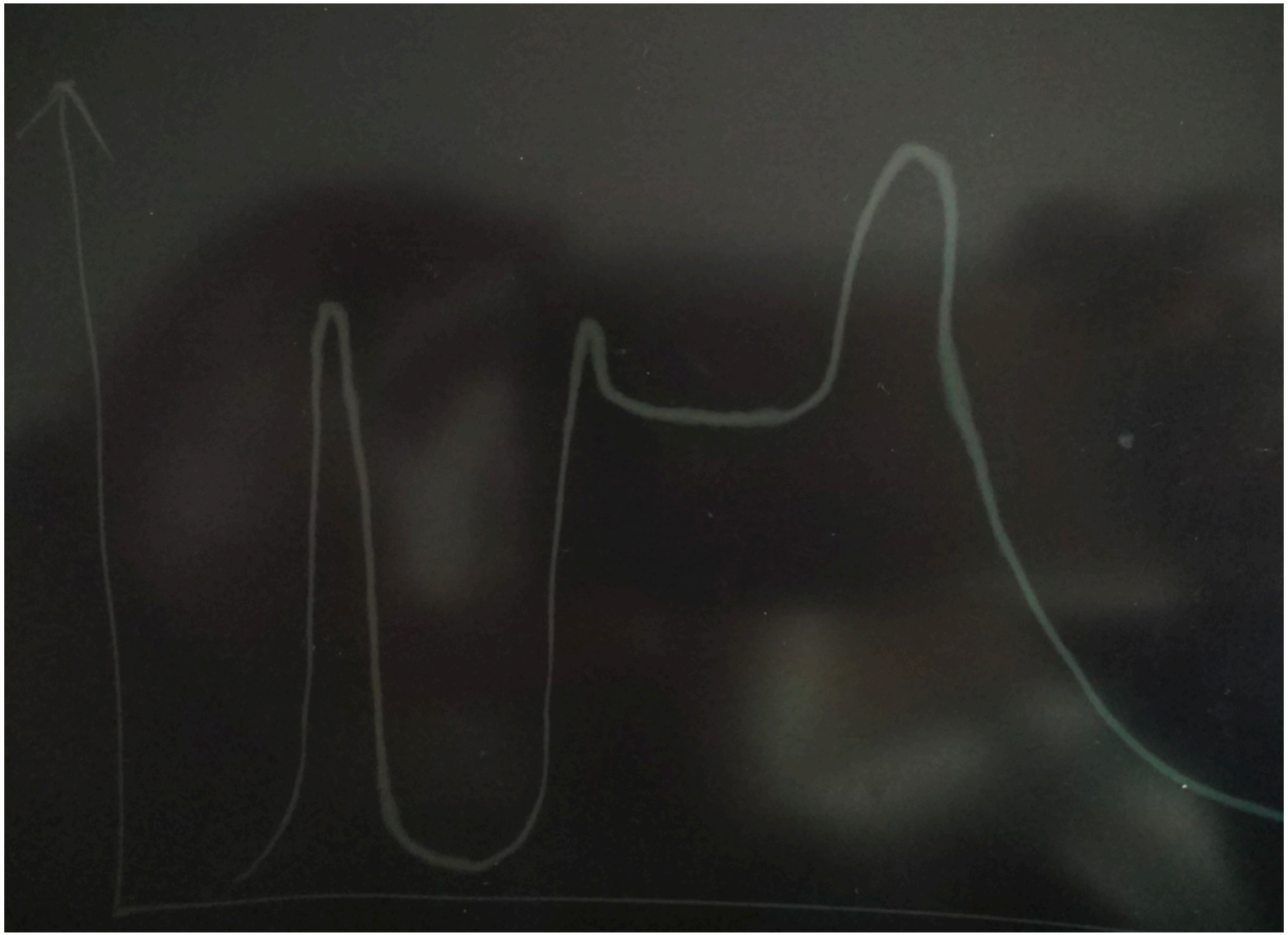
找到数据关键点-高潮、尿点、低谷

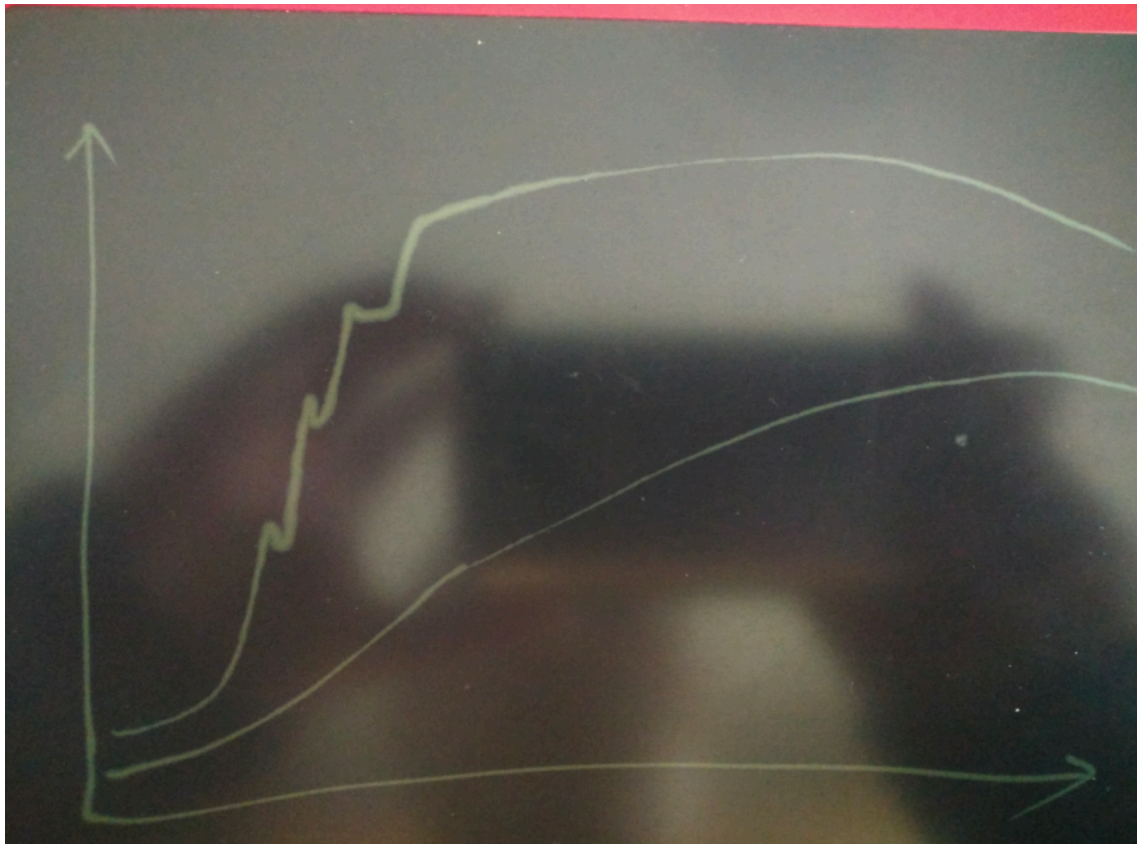
- https://github.com/digoal/blog/blob/master/201612/20161203_01.md











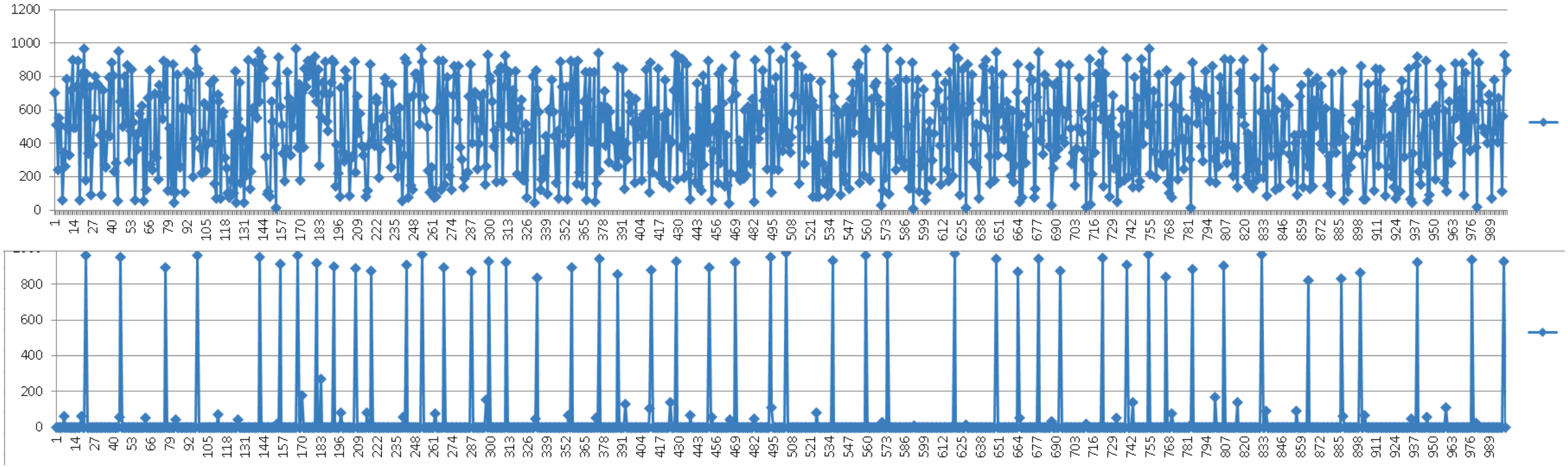
- query 5
- https://github.com/digoal/blog/blob/master/201612/20161203_01.md

```
select id,
  val,
  min(val) over(order by id rows between 10 preceding and current row) as left_min,  -- 左边相邻10个点的最小
  max(val) over(order by id rows between 10 preceding and current row) as left_max,  -- 左边相邻10个点的最大
  min(val) over(order by id rows between current row and 10 following) as right_min,  -- 右边相邻10个点的最小
  max(val) over(order by id rows between current row and 10 following) as right_max,  -- 右边相邻10个点的最大
  min(val) over(order by id rows between 10 preceding and 10 following) as range_min,  -- 左边相邻10个点的最
  max(val) over(order by id rows between 10 preceding and 10 following) as range_max,  -- 右边相邻10个点的最
  lag(val) over(order by id) as lag_val, -- 上一个值(第一条为空)
  lead(val) over(order by id) as lead_val, -- 下一个值(最后一条为空)
  -- avg(val) over(order by id rows between 10 preceding and current row) as range_left_avg, -- 左局部平均值
  -- stddev(val) over(order by id rows between 10 preceding and current row) as range_left_stddev, -- 左局
  -- avg(val) over(order by id rows between current row and 10 following) as range_right_avg, -- 右局部平均
  -- stddev(val) over(order by id rows between current row and 10 following) as range_right_stddev, -- 右
  min(val) over() as global_min, -- 全局的最小值 (包括当前点)
  max(val) over() as global_max  -- 全局的最大值 (包括当前点)
  from wind
```

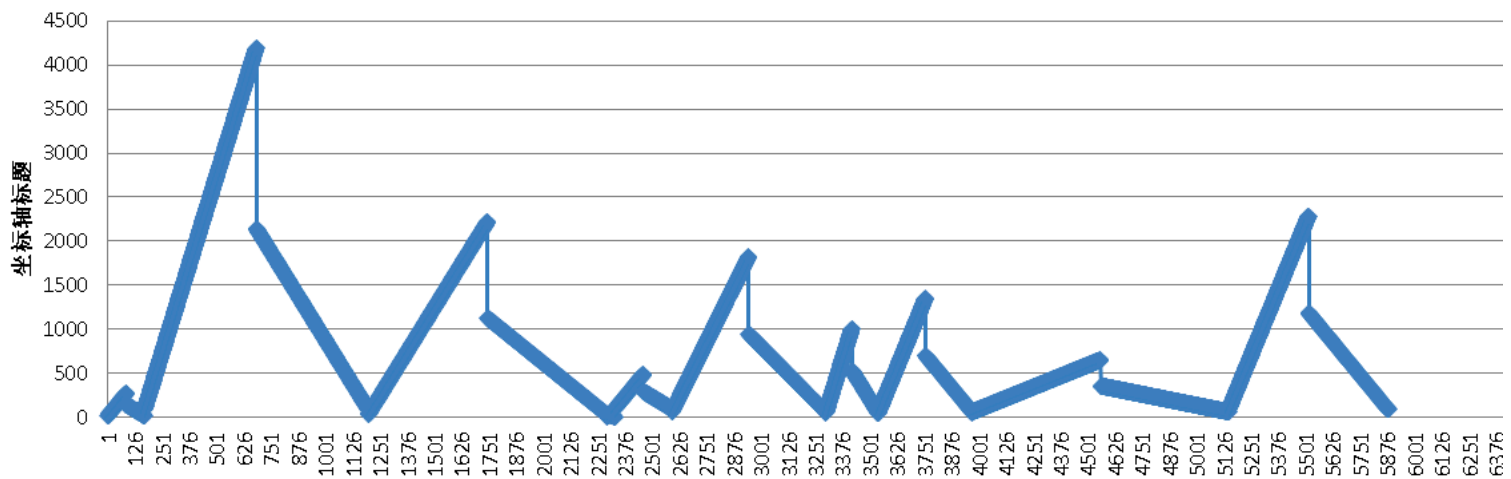
```
 left_min,  -- 左边相邻10个点的最小值（包括当前点）, 辐射半径的左边 , 值越大，可以找到更平滑变化的高潮和低谷
 left_max,  -- 左边相邻10个点的最大值（包括当前点）, 辐射半径的左边 , 值越大，可以找到更平滑变化的高潮和低谷
 right_min,  -- 右边相邻10个点的最小值（包括当前点）, 辐射半径的右边 , 值越大，可以找到更平滑变化的高潮和低谷
 right_max,  -- 右边相邻10个点的最大值（包括当前点）, 辐射半径的右边 , 值越大，可以找到更平滑变化的高潮和低谷
s range_min,  -- 左边相邻10个点的最小值（包括当前点）, 辐射半径两边全部 , 值越大，可以找到更平滑变化的高潮和低谷
s range_max,  -- 右边相邻10个点的最大值（包括当前点）, 辐射半径两边全部 , 值越大，可以找到更平滑变化的高潮和低谷


 as range_left_avg, -- 左局部平均值 , 值越大，可以找到更平滑变化的高潮和低谷
ow) as range_left_stddev, -- 左局部采样标准方差 , 值越大，可以找到更平滑变化的高潮和低谷
 as range_right_avg, -- 右局部平均值 , 值越大，可以找到更平滑变化的高潮和低谷
ng) as range_right_stddev, -- 右局部采样标准方差 , 值越大，可以找到更平滑变化的高潮和低谷
```

- query5 找关键点


1200
1000
800
600
400
200
0
1 14 27 40 53 66 79 92 105 118 131 144 157 170 183 196 209 222 235 248 261 274 287 300 313 326 339 352 365 378 391 404 417 430 443 456 469 482 495 508 521 534 547 560 573 586 599 612 625 638 651 664 677 690 703 716 729 742 755 768 781 794 807 820 833 846 859 872 885 898 911 924 937 950 963 976 989
800
600
400
200
0
1 14 27 40 53 66 79 92 105 118 131 144 157 170 183 196 209 222 235 248 261 274 287 300 313 326 339 352 365 378 391 404 417 430 443 456 469 482 495 508 521 534 547 560 573 586 599 612 625 638 651 664 677 690 703 716 729 742 755 768 781 794 807 820 833 846 859 872 885 898 911 924 937 950 963 976 989

- query 5 找关键点


坐标轴标题
4500
4000
3500
3000
2500
2000
1500
1000
500
0
1
126
251
376
501
626
751
876
1001
1126
1251
1376
1501
1626
1751
1876
2001
2126
2251
2376
2501
2626
2751
2876
3001
3126
3251
3376
3501
3626
3751
3876
4001
4126
4251
4376
4501
4626
4751
4876
5001
5126
5251
5376
5501
5626
5751
5876
6001
6126
6251
6376

id	val	left_min	left_max	right_min	right_max	range_min	range_max	lag_val	lead_v
92632	273	243	273	133.5	273	133.5	273	270	147
92633	147.0	147.0	273	132.0	147.0	132.0	273	273	145
92716	22.5	22.5	37.5	22.5	165	22.5	165	24.0	
92717	93	22.5	93	93	173	22.5	173	22.5	1
93230	4197	4117	4197	2105.0	4197	2105.0	4197	4189	2141
93231	2141.0	2141.0	4197	2101.0	2141.0	2101.0	4197	4197	2137
93744	89.0	89.0	129.0	36	89.0	36	129.0	93.0	
93745	36	36	125.0	36	76	36	125.0	89.0	
94292	2224	2184	2224	1110.0	2224	1110.0	2224	2220	1128
94293	1128.0	1128.0	2224	1108.0	1128.0	1108.0	2224	2224	1126
94840	34.0	34.0	54.0	10	34.0	10	54.0	36.0	
94841	10	10	52.0	10	30	10	52.0	34.0	
94858	44	24	44	17.0	44	17.0	44	42	26
94859	26.0	26.0	44	16.0	26.0	16.0	44	44	25
94876	9.0	9.0	19.0	9.0	128	9.0	128	10.0	1
94877	101	9.0	101	101	131	9.0	131	9.0	1
95007	491	461	491	281.0	491	281.0	491	488	294
95008	294.5	294.5	491	279.5	294.5	279.5	491	491	293
95138	99.5	99.5	114.5	67	112	67	114.5	101.0	
95139	67	67	113.0	67	117	67	117	99.5	
95491	1827	1777	1827	922.0	1827	922.0	1827	1822	944
95492	944.5	944.5	1827	919.5	944.5	919.5	1827	1827	942
95965	1013	933	1013	493.0	1013	493.0	1013	1005	529
95966	529.0	529.0	1013	489.0	529.0	489.0	1013	1013	525
96086	49.0	49.0	89.0	49.0	114	49.0	114	53.0	
96302	1350	1290	1350	675.0	1350	675.0	1350	1344	702
96303	702.0	702.0	1350	672.0	702.0	672.0	1350	1350	699
96518	57.0	57.0	87.0	57.0	75	57.0	87.0	60.0	
97106	653	643	653	354.5	653	354.5	653	652	359
97107	359.0	359.0	653	354.0	359.0	354.0	653	653	358
98061	2287	2227	2287	1159.0	2287	1159.0	2287	2281	1186
98062	1186.0	1186.0	2287	1156.0	1186.0	1156.0	2287	2287	1183

(32 rows)

目录

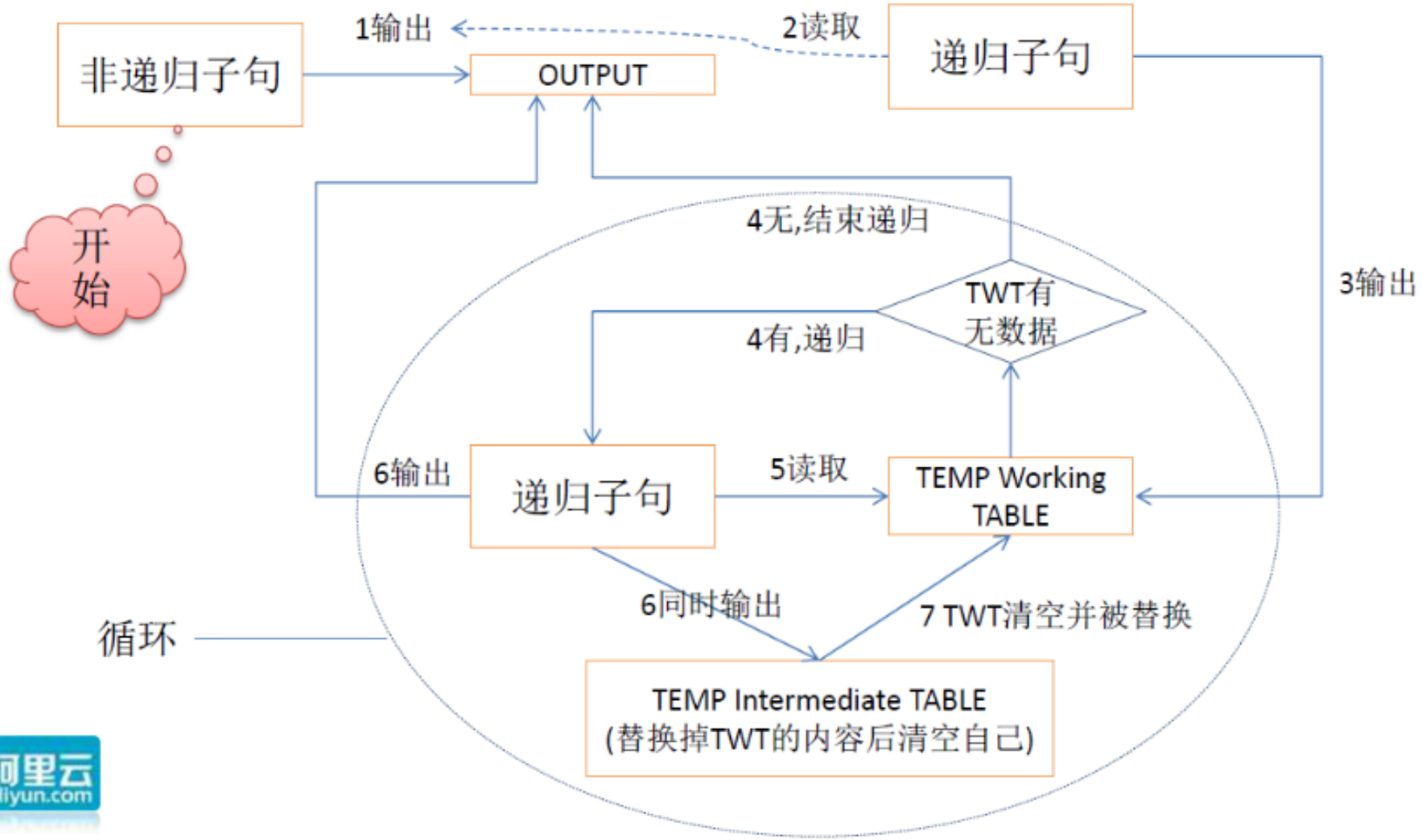
- WITH ORDINALITY
- LATERAL
- GROUPING SETS, CUBE and ROLLUP , grouping
- 窗口
- cte，递归
- skip locked
- distinct on
- 秒杀
- 批量sql
- 数据分析
- 流计算
- rule、异步消息
- 线性回归、预测
- 化学分析
- 机器学习
- update|delete limit
- prepare 动态sql in function

cte,递归查询应用场景

- 图式搜索
  - https://github.com/digoal/blog/blob/master/201801/20180102_04.md
- skip index scan模拟
  - https://github.com/digoal/blog/blob/master/201611/20161128_02.md
- 树状数据处理(多级分佣、消费链累加、溯源)
  - https://github.com/digoal/blog/blob/master/201808/20180808_02.md
  - https://github.com/digoal/blog/blob/master/201604/20160405_01.md
- 降低大量扫描、计算-性能优化
  - https://github.com/digoal/blog/blob/master/201612/20161201_01.md

递归查询

- UNION 去重复(去重复时NULL 视为等同)
- 图中所有输出都涉及UNION [ALL]的操作, 包含以往返回的记录和当前返回的记录



递归查询案例 - 图式搜索

https://github.com/digoal/blog/blob/master/201706/20170601_02.md
https://github.com/digoal/blog/blob/master/201801/20180102_04.md

```
create or replace function graph_search1(
  IN i_root int,                          -- 根据哪个节点开始搜
  IN i_depth int  default 99999,          -- 搜索层级、深度限制
  IN i_limit int8 default 2000000000,     -- 限制每一层返回的记录数
  IN i_weight float8 default 0,           -- 限制权重
  OUT o_path int[],                       -- 输出：路径，ID 组成的数组
  OUT o_point1 int,                       -- 输出：点1 ID
  OUT o_point2 int,                       -- 输出：点2 ID
  OUT o_link_prop jsonb,                  -- 输出：当前两点之间的连接属性
  OUT o_depth int                         -- 输出：当前深度、层级
) returns setof record as $$
declare
  sql text;
begin
sql := format($_$
```

递归查询案例 - 图式搜索

```
WITH RECURSIVE search_graph(
  c1,     -- 点1
  c2,     -- 点2
  prop,   -- 当前边的属性
  depth,  -- 当前深度，从1开始
  path,   -- 路径，数组存储
  cycle   -- 是否循环
) AS (
        select c1,c2,prop,depth,path,cycle from (
        SELECT                                  -- ROOT节点查询
          g.c1,                                 -- 点1
          g.c2,                                 -- 点2
          g.prop,                               -- 边的属性
          1 depth,                              -- 初始深度=1
          ARRAY[g.c1] path,                     -- 初始路径
          false  as cycle                       -- 是否循环(初始为否)
        FROM a AS g
        WHERE
          c1 = %s                                                   -- ROOT节点=?
          AND coalesce((g.prop->>'weight')::float8,0) >= %s         -- 相关性权重
          ORDER BY coalesce((g.prop->>'weight')::float8,0) desc     -- 可以使用ORDER BY，例如返回权重排在前面的N条。
          limit %s                              -- 每个层级限制多少条？
        ) t
```

递归查询案例 - 图式搜索

```
      UNION ALL
        select c1,c2,prop,depth,path,cycle from (
        SELECT                                  -- 递归子句
          g.c1,                                 -- 点1
          g.c2,                                 -- 点2
          g.prop,                               -- 边的属性
          sg.depth + 1 depth,                   -- 深度+1
          path || g.c1 path,                    -- 路径中加入新的点
          (g.c1 = ANY(path)) as cycle           -- 是否循环，判断新点是否已经在之前的路径中
        FROM a AS g, search_graph AS sg         -- 循环 INNER JOIN
        WHERE
          g.c1 = sg.c2                          -- 递归JOIN条件
          AND NOT cycle                         -- 防止循环
          AND sg.depth <= %s                    -- 搜索深度=?
          AND coalesce((g.prop->>'weight')::float8,0) >= %s          -- 相关性权重
          ORDER BY coalesce((g.prop->>'weight')::float8,0) desc      -- 可以使用ORDER BY，例如返回权重排在前面的N条。
          limit %s                              -- 每个层级限制多少条？
        ) t
)
```

递归查询案例 - 图式搜索

```
SELECT path||c2 as o_path, c1 as o_point1, c2 as o_point2, prop as o_link_prop, depth as o_depth
FROM search_graph;                                  -- 查询递归表，可以加LIMIT输出，也可以使用游标
$_$, i_root, i_weight, i_limit, i_depth, i_weight, i_limit
);

return query execute sql;

end;
$$ language plpgsql strict;
```

递归查询案例 - 图式搜索

```
postgres=# select * from graph_search1(31208, 3, 100, 0);
             o_path              | o_point1 | o_point2 | o_link_prop | o_depth
---------------------------------+----------+----------+-------------+---------
 {31208,344710}                  |    31208 |   344710 |             |       1
 {31208,319951}                  |    31208 |   319951 |             |       1
 {31208,340938}                  |    31208 |   340938 |             |       1
 {31208,325272}                  |    31208 |   325272 |             |       1
 {31208,346519}                  |    31208 |   346519 |             |       1
 {31208,314594}                  |    31208 |   314594 |             |       1
 {31208,307217}                  |    31208 |   307217 |             |       1
 {31208,348009}                  |    31208 |   348009 |             |       1
 {31208,300046}                  |    31208 |   300046 |             |       1
 {31208,344359}                  |    31208 |   344359 |             |       1
 {31208,318790}                  |    31208 |   318790 |             |       1
 {31208,321034}                  |    31208 |   321034 |             |       1
 {31208,301609}                  |    31208 |   301609 |             |       1
 {31208,344339}                  |    31208 |   344339 |             |       1
 {31208,314087}                  |    31208 |   314087 |             |       1
------
 {31208,319951,3199647,31991191} |  3199647 | 31991191 |             |       3
 {31208,319951,3199647,31954904} |  3199647 | 31954904 |             |       3
 {31208,319951,3199647,31986691} |  3199647 | 31986691 |             |       3
 {31208,319951,3199647,31986448} |  3199647 | 31986448 |             |       3
 {31208,319951,3199647,31993624} |  3199647 | 31993624 |             |       3
 {31208,319951,3199647,31997771} |  3199647 | 31997771 |             |       3
 {31208,319951,3199647,31982764} |  3199647 | 31982764 |             |       3
 {31208,319951,3199647,31993420} |  3199647 | 31993420 |             |       3
 {31208,319951,3199647,31962666} |  3199647 | 31962666 |             |       3
 {31208,319951,3199647,31957536} |  3199647 | 31957536 |             |       3
(300 rows)
```

目录

- WITH ORDINALITY
- LATERAL
- GROUPING SETS, CUBE and ROLLUP , grouping
- 窗口
- cte，递归
- skip locked、advisory lock
- distinct on
- 秒杀
- 批量sql
- 数据分析
- 流计算
- rule、异步消息
- 线性回归、预测
- 化学分析
- 机器学习
- update|delete limit
- prepare 动态sql in function

消除锁等待，提高处理吞吐

- https://github.com/digoal/blog/blob/master/201610/20161018_01.md

目录

- WITH ORDINALITY
- LATERAL
- GROUPING SETS, CUBE and ROLLUP , grouping
- 窗口
- cte，递归
- skip locked
- distinct on
- 秒杀
- 批量sql
- 数据分析
- 流计算
- rule、异步消息
- 线性回归、预测
- 化学分析
- 机器学习
- update|delete limit
- prepare 动态sql in function

- https://github.com/digoal/blog/blob/master/201710/20171017_02.md

目录

- WITH ORDINALITY
- LATERAL
- GROUPING SETS, CUBE and ROLLUP , grouping
- 窗口
- cte，递归
- skip locked
- distinct on
- 秒杀
- 批量sql
- 数据分析
- 流计算
- rule、异步消息
- 线性回归、预测
- 化学分析
- 机器学习
- update|delete limit
- prepare 动态sql in function

- https://github.com/digoal/blog/blob/master/201801/20180105_03.md

目录

- WITH ORDINALITY
- LATERAL
- GROUPING SETS, CUBE and ROLLUP , grouping
- 窗口
- cte，递归
- skip locked
- distinct on
- 秒杀
- 批量sql
- 数据分析
- 流计算
- rule、异步消息
- 线性回归、预测
- 化学分析
- 机器学习
- update|delete limit
- prepare 动态sql in function

- https://github.com/digoal/blog/blob/master/201704/20170424_05.md

目录

- WITH ORDINALITY
- LATERAL
- GROUPING SETS, CUBE and ROLLUP , grouping
- 窗口
- cte，递归
- skip locked
- distinct on
- 秒杀
- 批量sql
- 数据分析
- 流计算
- rule、异步消息
- 线性回归、预测
- 化学分析
- 机器学习
- update|delete limit
- prepare 动态sql in function

- https://github.com/digoal/blog/blob/master/201712/20171225_01.md

目录

- WITH ORDINALITY
- LATERAL
- GROUPING SETS, CUBE and ROLLUP , grouping
- 窗口
- cte，递归
- skip locked
- distinct on
- 秒杀
- 批量sql
- 数据分析

- 流计算
- rule、异步消息
- 线性回归、预测
- 化学分析
- 机器学习
- update|delete limit
- prepare 动态sql in function

- https://github.com/digoal/blog/blob/master/201811/20181101_02.md
- https://github.com/digoal/blog/blob/master/201807/20180713_03.md
- https://github.com/digoal/blog/blob/master/201711/20171111_01.md
- https://github.com/digoal/blog/blob/master/201712/20171225_01.md

目录

- WITH ORDINALITY
- LATERAL
- GROUPING SETS, CUBE and ROLLUP , grouping
- 窗口
- cte，递归
- skip locked
- distinct on
- 秒杀
- 批量sql
- 数据分析
- 流计算
- rule、异步消息
- 线性回归、预测
- 化学分析
- 机器学习
- update|delete limit
- prepare 动态sql in function

- https://github.com/digoal/blog/blob/master/201904/20190426_01.md
- https://github.com/digoal/blog/blob/master/201904/20190426_01.md
- https://github.com/digoal/blog/blob/master/201503/20150305_01.md

目录

- WITH ORDINALITY
- LATERAL
- GROUPING SETS, CUBE and ROLLUP , grouping
- 窗口
- cte，递归
- skip locked
- distinct on
- 秒杀
- 批量sql
- 数据分析
- 流计算
- rule、异步消息
- 线性回归、预测
- 化学分析
- 机器学习
- update|delete limit
- prepare 动态sql in function

- https://github.com/digoal/blog/blob/master/201911/20191125_01.md

目录

- WITH ORDINALITY
- LATERAL
- GROUPING SETS, CUBE and ROLLUP , grouping
- 窗口
- cte，递归
- skip locked
- distinct on
- 秒杀
- 批量sql
- 数据分析
- 流计算
- rule、异步消息
- 线性回归、预测
- 化学分析
- 机器学习
- update|delete limit
- prepare 动态sql in function

- https://github.com/digoal/blog/blob/master/201511/20151111_01.md

目录

- WITH ORDINALITY
- LATERAL
- GROUPING SETS, CUBE and ROLLUP , grouping
- 窗口
- cte，递归
- skip locked
- distinct on
- 秒杀
- 批量sql
- 数据分析
- 流计算
- rule、异步消息
- 线性回归、预测
- 化学分析
- 机器学习
- update|delete limit
- prepare 动态sql in function

- https://github.com/digoal/blog/blob/master/201608/20160827_01.md

目录

- WITH ORDINALITY
- LATERAL
- GROUPING SETS, CUBE and ROLLUP , grouping
- 窗口
- cte，递归
- skip locked
- distinct on
- 秒杀
- 批量sql
- 数据分析
- 流计算
- rule、异步消息
- 线性回归、预测
- 化学分析
- 机器学习
- update|delete limit
- prepare 动态sql in function

- https://github.com/digoal/blog/blob/master/201803/20180323_02.md

参考资料

- 案例
  - https://github.com/digoal/blog/blob/master/201704/20170411_04.md
  - https://github.com/digoal/blog/blob/master/201802/20180226_05.md
- MySQL手册
  - https://www.mysqltutorial.org/
  - https://dev.mysql.com/doc/refman/8.0/en/
- PG 管理、开发规范
  - https://github.com/digoal/blog/blob/master/201609/20160926_01.md
- PG手册
  - https://www.postgresql.org/docs/current/index.html
  - https://www.postgresqltutorial.com/postgresql-tutorial/postgresql-vs-mysql/
- GIS手册
  - http://postgis.net/docs/manual-3.0/

一期开课计划(PG+MySQL联合方案)

- - 2019.12.30 19:30 RDS PG产品概览，如何与MySQL结合使用
- - 2019.12.31 19:30 如何连接PG，GUI，CLI的使用
- - 2020.1.3 19:30 如何压测PG数据库、如何瞬间构造海量测试数据
- - 2020.1.6 19:30 MySQL与PG对比学习(面向开发者)
- - 2020.1.7 19:30 如何将MySQL数据同步到PG（DTS）
- - 2020.1.8 19:30 PG外部表妙用 - mysql_fdw, oss_fdw（直接读写MySQL数据、冷热分离）
- - 2020.1.9 19:30 PG应用场景介绍 - 并行计算，实时分析
- - 2020.1.10 19:30 PG应用场景介绍 - GIS
- - 2020.1.13 19:30 PG应用场景介绍 - 用户画像、实时营销系统
- - 2020.1.14 19:30 PG应用场景介绍 - 多维搜索
- - 2020.1.15 19:30 PG应用场景介绍 - 向量计算、图像搜索
- - 2020.1.16 19:30 PG应用场景介绍 - 全文检索、模糊查询
- - 2020.1.17 19:30 PG 数据分析语法介绍
- - 2020.1.18 19:30 PG 更多功能了解：扩展语法、索引、类型、存储过程与函数。如何加入PG技术社群

本课程习题

- 物化视图和普通视图的区别
- 物化视图用在什么场景
- 实时清洗数据可以用PG的什么功能
- 数据采样用在什么场景，有哪两种采样粒度
- 数据加密用什么插件
- 数据去重通常使用什么语法
- 多个字段范围检索，除了联合索引，还有什么索引加速更快
- 行级别安全特性可以实现什么安全隔离需求,通常用在什么业务场景

本课程习题

- 返回多条记录的函数，如何给每条记录一个唯一数值标记
- 子查询中能查询子查询外面的TABLE吗
- 什么语法可以一次聚合计算多个分组
- 递归查询通常用在哪些场景
- 如何避免并发更新、删除等行级别锁冲突的等待
- 秒杀为什么慢？如何优化？

本课程习题

- 批量sql的语法
- 流计算有哪些方法
- rule、异步消息用在什么场景非常高效
- 线性回归、预测，使用了数据库的什么功能
- 化学分析用了哪个插件
- 机器学习使用哪个插件
- update|delete limit用什么语法
- prepare 动态sql in function使用了什么语法

技术社群

PG技术交流钉钉群(3600+人)