Data Engineering and SQL

Igor Wojnicki

May 30, 2023

Table of Contents

SQL Highlights

- out-of-core processing
- performance considerations
- window functions

Table of Contents

Traffic

- Case: traffic and lighting class analysis.

```
CREATE TABLE public.traffic (
    detector_id INTEGER NOT NULL,
    starttime TIMESTAMP WITHOUT TIME ZONE NOT NULL,
    endtime TIMESTAMP WITHOUT TIME ZONE NOT NULL,
    count SMALLINT,
    CONSTRAINT traffic_check CHECK ((endtime > starttime))
);
```

- SQL Dump: 33 GB
- Database: 50 GB

Test Case

- 1 detector,
- 1 day,
- 15 min. moving window sum.

Traffic

```
1  -- traffic in 15 min. intervals,
2  -- moving 15 min window at each reading
3  SELECT
4    t1.starttime,
5    sum(t2.count)*4*24 as sum
6  FROM
7    traffic t1
8    JOIN
9    traffic t2
10   USING (detector_id)
11 WHERE
12   t2.starttime < (t1.starttime + INTERVAL '15 minutes')
13   AND t2.starttime >= t1.starttime
14   AND (t1.starttime between '2015-06-23' AND '2015-06-24')
15   AND t1.detector_id=1971 -- detector
16 GROUP BY t1.starttime
17 ;
18 Time: 167.139 ms
```

Traffic, Subquery

```
1  -- traffic in 15 min. intervals,
2  -- moving 15 min window at each reading, subquery
3  SELECT
4    t1.starttime,
5    (SELECT sum(t2.count)*4*24 as sum FROM traffic t2
6    WHERE t2.detector_id=t1.detector_id
7    AND t2.starttime < (t1.starttime + INTERVAL '15 minutes')
8    AND t2.starttime >= t1.starttime) AS sum
9  FROM
10   traffic t1
11 WHERE
12   (t1.starttime BETWEEN '2015-06-23' AND '2015-06-24') -- t
13   AND t1.detector_id=1971 -- detector
14 ;
15
16
17
18 Time: 16.318 ms
```

Traffic, Does it matter?

- 141 detectors
- 1.5 minutes between readings
- 1 year worth of data

Traffic, Materialized Views

```
1  CREATE MATERIALIZED VIEW traffic15 AS
2  (SELECT
3    detector_id,
4    (timestamp 'epoch' +
5      (floor(extract(epoch from starttime)/(15*60))*15*60)*
6       interval '1 second') as time,
7    sum(count)*4*24 traffic_from
8    FROM traffic
9    GROUP by time, detector_id)
10 ;
11
12 CREATE INDEX traffic15_time_idx ON traffic15 (time);
13 CREATE INDEX traffic15_detector_id_idx
14        ON traffic15 (detector_id);
15
16 -- data consistency:
17 REFRESH MATERIALIZED VIEW;
```

Traffic, Window, Deltas

- Window Functions, Window Tutorial

```
1  SELECT *,
2    lag(traffic) OVER (PARTITION BY segment_id)
3      AS lag,
4    abs(traffic-lag(traffic) OVER (PARTITION BY segment_id))
5      AS delta
6  FROM (
7    SELECT segment_id,starttime,sum(count)*40*24 AS traffic
8    FROM traffic
9    ...
10   GROUP BY segment_id,starttime
11   ORDER BY segment_id,starttime
12 ) AS traffic ORDER BY delta DESC;
```

Traffic, Window, Deltas, Results

Excerpt, some rows at the beginning skipped, `lag` and `delta` being `nulls`.

segment_id	starttime	traffic	lag	delta
100638	2016-06-20 10:16:31	352320	27840	324480
100638	2016-06-20 10:18:01	37440	352320	314880
100638	2016-02-10 08:39:01	21120	245760	224640
100638	2016-02-10 08:37:31	245760	33600	212160
100638	2015-10-23 08:42:01	209280	33600	175680
100638	2015-07-08 10:34:31	147840	14400	133440
100638	2015-07-08 10:36:01	27840	147840	120000
100697	2015-09-16 13:51:01	159360	42240	117120
100562	2015-07-27 20:57:01	116160	960	115200
100562	2015-07-27 20:58:31	1920	116160	114240
100627	2016-05-15 16:58:31	111360	0	111360
100562	2015-07-26 23:39:01	0	108480	108480
100697	2015-09-16 13:52:31	59520	159360	99840
100638	2015-10-23 08:45:01	31680	131520	99840
100562	2015-07-27 23:46:31	99840	0	99840

Traffic, Histogram, Window

```
1  SELECT delta, count(*) FROM(
2   SELECT *,
3     abs(traffic-lag(traffic) OVER (PARTITION BY segment_id))
4       AS delta
5   FROM (
6     SELECT segment_id,starttime,sum(count)*40*24 AS traffic
7     FROM traffic, ...
8     GROUP BY segment_id, starttime
9     ORDER BY segment_id, starttime
10  ) AS traffic ORDER BY delta DESC
11 ) AS d GROUP BY delta ORDER BY count;
```

Traffic, Histogram, Window, Results

delta	count
224640	1
111360	1
96000	2
86400	2
...	...

Warning: window sequential scan possible! Check: `EXPLAIN`.

Table of Contents

SQL From File

```
with open("/tmp/tmp.csv","w") as f:
    f.write("num1,num2,data\n")
    for i in range(30000000):
        f.write("{},{},test data {}\n".format(i,i,i))
```

```
ls -lh /tmp/tmp.csv
```

```
-rw-rw-r-- 1 wojnicki wojnicki 1.1G May 23 16:59 /tmp/tmp.cs
```

textql

- https://github.com/dinedal/textql

```
/usr/bin/time -f '%E %M' textql -header -sql \
              'SELECT max(num1),data FROM tmp'  \
              /tmp/tmp.csv 2>&1

29999999,test data 29999999
1:56.92 1159524
```

dsq

- https://github.com/multiprocessio/dsq

```
/usr/bin/time -f '%E %M' dsq /tmp/tmp.csv \
              'SELECT max(num1),data FROM {}' 2>&1

[{"max(num1)":"9999999","data":"test data 9999999"}]
1:40.39 5950080
```

- Column types are TEXT, hence results are strange...

sqlite

- https://sqlite.org

```
CREATE TABLE tmp (num1 integer, num2 integer, data text);
/usr/bin/time -f '%E %M' sqlite3 /tmp/tmp.sqlite \
      ".import --skip 1 --csv /tmp/tmp.csv tmp" 2>&1

0:29.89 7088

ls -lh /tmp/tmp.sqlite

-rw-r--r-- 1 wojnicki wojnicki 1.1G May 23 17:31 /tmp/tmp.sq

/usr/bin/time -f '%E %M' sqlite3 /tmp/tmp.sqlite \
      'SELECT max(num1),data FROM tmp' 2>&1

29999999|test data 29999999
0:01.97 7100
```

Pandas

- https://pandas.pydata.org/
- file: test-tmp.py

```
1 import pandas as pd
2 df = pd.read_csv('/tmp/tmp.csv')
3 print(df.loc[df['num1'].idxmax()].data)
```

```
test data 29999999
```

```
/usr/bin/time -f '%E %M' python3 test-tmp.py 2>&1
```

```
test data 29999999
0:15.64 3972532
```

Quick Reports ;)

- file: reports

```
1  #!/bin/bash
2  export PGHOST=localhost
3  export PGDATABASE=database
4  export PGUSER=user
5  export PGPASSWORD=secret
6
7  psql -H -q < `dirname $0`/reports.sql | \
8      mailx -a 'Content-Type: text/html'  \
9            -s "Exfluency Report" \
10           vip@somewhere.pl robert@somewhere.com
```

Quick Reports ;)

reports.sql

```
1  \echo Requester
2  SELECT name, surname, email,
3    count(*) AS all,
4    sum(CASE WHEN status = 'DELETED' THEN 1 ELSE 0 END)
5      AS deleted,
6    sum(CASE WHEN status = 'FINISHED' THEN 1 ELSE 0 END)
7      AS finished,
8    sum(CASE WHEN status = 'FINISHED' AND
9        delivery_date > (CURRENT_DATE-'1 week'::INTERVAL)
10       THEN 1 ELSE 0 END) AS finished7
11       -- finished in last 7 days
12 FROM user_information ui
13   ...
14 GROUP BY name, surname, email
15 ORDER BY 2,1;
```