
Making your PostgreSQL database sing
Frank Wiles

Performance is about doing less not doing something faster

REVSYS | revolution systems

Important postgresql.conf knobs:

- shared_buffers

REVSYS | revolution systems

Important postgresql.conf knobs:

- shared_buffers set to 10-20% of RAM

REVSYS | revolution systems

Important postgresql.conf knobs:

- shared_buffers — set to 10-20% of RAM
- effective_cache_size

REVSYS | revolution systems

Important postgresql.conf knobs:

- shared_buffers set to 10-20% of RAM
- effective_cache_size set to 75% of RAM

REVSYS | revolution systems

Important postgresql.conf knobs:

- shared_buffers — set to 10-20% of RAM
- effective_cache_size — set to 75% of RAM
- work_mem

REVSYS | revolution systems

Important postgresql.conf knobs:

- shared_buffers — set to 10-20% of RAM
- effective_cache_size — set to 75% of RAM
- work_mem — size of ORDER BYs

REVSYS | revolution systems

Materialized views!

REVSYS | revolution systems

The dead simple way...

```
CREATE SEQUENCE item_seq;
CREATE TABLE item (
    id          int4 PRIMARY KEY DEFAULT NEXTVAL('item_seq'),
    title       varchar,
    description text
);

CREATE SEQUENCE vote_seq;

CREATE TABLE vote (
    id          int4 PRIMARY KEY DEFAULT NEXTVAL('vote_seq'),
    item_id     int4 REFERENCES item(id),
    vote        int2
);
```

4.2 seconds

REVSYS | revolution systems

The dead simple way...

```
CREATE TABLE items_by_vote_total
AS select item.id, item.title,
(select count(id) FROM vote
WHERE item.id=vote.item_id AND
vote.vote = 1) AS vote_total
FROM item ORDER BY vote_total
DESC;
```

REVSYS | revolution systems

The dead simple way...

```
test1=# select * from items_by_vote_total LIMIT 20;
 id  |      title       | vote_total
-----+------------------+------------
 635 | Item #635 Title  |         26
 944 | Item #944 Title  |         24
 606 | Item #606 Title  |         24
 590 | Item #590 Title  |         24
 301 | Item #301 Title  |         23
 932 | Item #932 Title  |         23
 523 | Item #523 Title  |         22
 952 | Item #952 Title  |         22
  88 | Item #88 Title   |         22
 452 | Item #452 Title  |         22
 715 | Item #715 Title  |         22
 189 | Item #189 Title  |         22
 935 | Item #935 Title  |         21
  67 | Item #67 Title   |         21
 117 | Item #117 Title  |         21
  43 | Item #43 Title   |         21
 828 | Item #828 Title  |         21
 795 | Item #795 Title  |         21
 254 | Item #254 Title  |         21
 290 | Item #290 Title  |         21
(20 rows)
```

0.345 seconds!

REVSYS | revolution systems

Real time updates with pl/python

```
CREATE SEQUENCE item_seq;
CREATE TABLE item (
    id          int4 PRIMARY KEY DEFAULT NEXTVAL('item_seq'),
    title       varchar,
    description text
);

CREATE SEQUENCE vote_seq;

CREATE TABLE vote (
    id          int4 PRIMARY KEY DEFAULT NEXTVAL('vote_seq'),
    item_id     int4 REFERENCES item(id),
    vote        int2
);

CREATE SEQUENCE items_by_vote_seq;

CREATE TABLE items_by_vote(
    id          int4 PRIMARY KEY DEFAULT NEXTVAL('items_by_vote_seq'),
    item_id     int4 REFERENCES item(id),
    votes       int4
);

CREATE INDEX items_by_vote_item_id ON items_by_vote(item_id);
```

Real time updates with pl/python

```
CREATE OR REPLACE FUNCTION item_add()
RETURNS TRIGGER AS
$$
item_id = TD["new"]["id"]
rv = plpy.execute("INSERT INTO items_by_vote (item_id,votes) VALUES (%d,0)" % item_id)

$$ LANGUAGE plpythonu;

CREATE OR REPLACE FUNCTION item_delete()
RETURNS TRIGGER AS
$$
item_id = TD["old"]["id"]
rv = plpy.execute("DELETE FROM vote WHERE item_id = %d" % item_id)
rv = plpy.execute("DELETE FROM items_by_vote WHERE item_id = %d" % item_id)

$$ LANGUAGE plpythonu;

CREATE OR REPLACE FUNCTION vote()
RETURNS TRIGGER AS
$$
    item_id = TD["new"]["item_id"]
    vote = TD["new"]["vote"]

    if vote == 1:
        rv = plpy.execute("UPDATE items_by_vote SET votes = votes + 1 WHERE item_id = %d" % item_id)
    elif vote == 0:
        rv = plpy.execute("UPDATE items_by_vote SET votes = votes - 1 WHERE item_id = %d" % item_id)

$$ LANGUAGE plpythonu;

CREATE TRIGGER item_add_trigger AFTER INSERT ON item FOR EACH ROW EXECUTE PROCEDURE item_add();

CREATE TRIGGER item_delete_trigger BEFORE DELETE ON item FOR EACH ROW EXECUTE PROCEDURE item_delete();

CREATE TRIGGER vote_add_trigger AFTER INSERT ON vote FOR EACH ROW EXECUTE PROCEDURE vote();
```

REVSYS | revolution systems

Other places to use this...

- Time based aggregation (hours -> days -> weeks)
- To denormalize frequent queries (User profiles maybe?)
- Any place you find SELECTs are more frequent than INSERTs/UPDATEs

REVSYS | revolution systems

Questions?

Slides will be available at http://www.revsys.com/talks later today

REVSYS | revolution systems