Interactive - 39 - materialized views

One of the ways that you can improve performance in PostgreSQL and other SQL databases is a materialized view. A regular view is a kind of "macro" that replaces the query's from with the body of the view.

In a materialized view the data itself is copied into the "view" and you can add indexes and other things that optimize queries on this set of data.

The downsize to materialized views is that the data can get out of sync with the original data.

Materialized views are very useful when the source of the data is a foreign data wrapper. Especially when the foreign data is slow or only periodically changes.

Let's use our table with some names in it. (From Interactive Homework 08)

Run File `hw39_0.sql`:

```
CREATE EXTENSION if not exists "uuid-ossp";
CREATE EXTENSION if not exists pgcrypto;

DROP TABLE if exists name_list ;

CREATE TABLE name_list (
    name_list_id UUID NOT NULL DEFAULT uuid_generate_v4() primary key,
    real_name text check ( length(real_name) >= 1 ) not null,
    age int check ( age > 0 and age < 154 ) not  null,
    state char varying (2) not null,
    pay numeric(10,2)
);

CREATE INDEX name_list_idx1 on name_list ( real_name );
CREATE INDEX user_real_name_ci_idx1 ON name_list ((lower(real_name)));
```

First let's refresh the data in name_list so that we know what is in it.

Run File `hw39_2.sql`:

```
DELETE FROM name_list ;

INSERT INTO name_list ( real_name, age, state, pay ) values
    ( 'Bob True',            22, 'WY', 31000 ),
    ( 'Jane True',           20, 'WY', 28000 ),
    ( 'Tom Ace',             31, 'NJ', 82500 ),
    ( 'Steve Pen',           33, 'NJ', 89400 ),
    ( 'Laura Jean Alkinoos', 34, 'PA', 120000 ),
    ( 'Philip Schlump',      62, 'WY', 101200 ),
    ( 'Liz Trubune',         30, 'WY', 48000 ),
    ( 'Lary Smith',          58, 'NJ', 48000 ),
    ( 'Dave Dave',           21, 'NJ', 48000 ),
    ( 'Laura Ann Alkinoos',  34, 'PA', 48000 )
;
```

Now a materialize view that rolls up the count of the number of names in each state.

```
DROP materialized view count_by_state_of_names ;

CREATE materialized view count_by_state_of_names as
    SELECT count(1) as count_by_state,
        state
    FROM name_list
    GROUP BY state
;
CREATE INDEX count_by_state_of_names_p1 on count_by_state_of_names ( count_by_state );
CREATE INDEX count_by_state_of_names_p2 on count_by_state_of_names ( state );
```

To refresh a materialized view you use (you need to do a refresh to populate the view for the first time also):

```
REFRESH MATERIALIZED VIEW count_by_state_of_names;
```

Now let's select the data:

```
SELECT * from count_by_state_of_names ;
```

You should get back 3 rows.

FilesToRun: hw39_0.sql

FilesToRun: hw39_2.sql