

PG_DWH TASK 3

Legal Notice:

This document contains privileged and/or confidential information and may not be disclosed, distributed or reproduced without the prior written permission of EPAM®.

```
Query
       Query History
   DROP TABLE IF EXISTS labs.person;
   SET search_path TO labs;
  CREATE TABLE labs.person (
6 id integer NOT NULL,
   name varchar(15)
   INSERT INTO person VALUES(1, 'Bob');
10
  INSERT INTO person VALUES(2, 'Alice');
11
   INSERT INTO person VALUES(3, 'Robert');
12
13
14
Data Output Messages Notifications
NOTICE: table "person" does not exist, skipping
INSERT 0 1
Query returned successfully in 214 msec.
```

Table created, values are inserted.

```
J SET search_path TO labs;

4 4 5 CREATE TABLE labs.person (
6 6 1d integer NOT NULL,
7 name varchar(15)
8 1)
9 NSERT INTO person VALUES(1, 'Bob');
10 INSERT INTO person VALUES(2, 'Alice');
11 INSERT INTO person VALUES(3, 'Robert');
12 13
14 CREATE TABLE IF NOT EXISTS labs.test_simple(a int,b int);
15 Insert into test_simple values (generate_series(1,18888989));

Data Output Messages Notifications
INSERT 0 1008888

Query returned successfully in 2 secs 353 msec.

Total rows: 3 of 3 | Query complete 00:00:02.353
```

Simple table created and populated in 2 seconds in first query.

```
Total rows: 3 of 3 Query complete 00:00:11.619
```

Second query took 11 seconds to execute.

```
--INSERT INTO person VALUES(), 'Bab');

10
--INSERT INTO person VALUES();
11
--INSERT INTO person VALUES();
12
13
14
CREATE TABLE IF NOT EXISTS labs.test_simple(a int,b int);
15
--insert into test_simple values (generate_series(1,1000000));
17
--insert into test_simple values (generate_series(1,5000000));
18
19
20
CREATE UNLOGGED TABLE IF NOT EXISTS labs.test_unlogged(a int, b int);
21
1nsert into test_unlogged values (generate_series(1,1000000));
22
23
24
25
Data Output Messages Notifications
INSERT 0 1000000
Query returned successfully in 1 secs 316 msec.
```

Unlogged table took 1.3 second to be populated 1.000.000 rows.

```
--INSERT INTO person VALUES(2, 'Nobert');
10 --INSERT INTO person VALUES(3, 'Robert');
11 --INSERT INTO person VALUES(3, 'Robert');
12 13
14 CREATE TABLE IF NOT EXISTS labs.test_simple(a int,b int);
15 --insert into test_simple values (generate_series(1,1000000));
17 --insert into test_simple values (generate_series(1,5000000));
18 19
20 CREATE UNLOGGED TABLE IF NOT EXISTS labs.test_unlogged(a int, b int);
11 insert into test_unlogged values (generate_series(1,5000000));
12 insert into test_simple values (generate_series(1,5000000));
18 22 Insert into test_simple values (generate_series(1,5000000));
19 20 Output Messages Notifications
10 Outpu
```

And second one took 11 second, as we see unlogged tables are faster than usual tables.

Business Template

```
23
24
25
26 CREATE TABLE labs.users(user_id INT GENERATED BY DEFAULT AS IDENTITY
27 PRIMARY KEY, login VARCHAR(30)) INHERITS (person);
28
29

Data Output Messages Notifications
CREATE TABLE
Query returned successfully in 154 msec.
```

Table created.

```
23
24
25
26 CREATE TABLE labs.users(user_id INT GENERATED BY DEFAULT AS IDENTITY
27 PRIMARY KEY, login VARCHAR(30)) INHERITS (person);
28
29 INSERT INTO users VALUES (1, 'new Bob', 'NewLogin');
30 INSERT INTO users VALUES (999, 'TestUser', 'TestLogin');

Data Output Messages Notifications

ERROR: invalid input syntax for type integer: "NewLogin"
LINE 2: INSERT INTO users VALUES (1, 'new Bob', 'NewLogin');

SQL state: 22P02
Character: 42
```

Third value need to be an integer as its inherits from person.

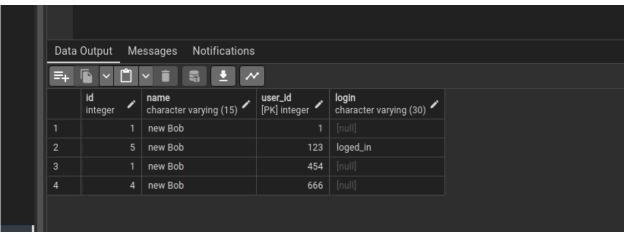
```
25
26 CREATE TABLE labs.users(user_id INT GENERATED BY DEFAULT AS IDENTITY
27 PRIMARY KEY, login VARCHAR(30)) INHERITS (person);
28
29 INSERT INTO users VALUES (5, 'new Bob', 123,'loged_in');
30 INSERT INTO users VALUES (999, 'TestUser', 'TestLogin');

Data Output Messages Notifications

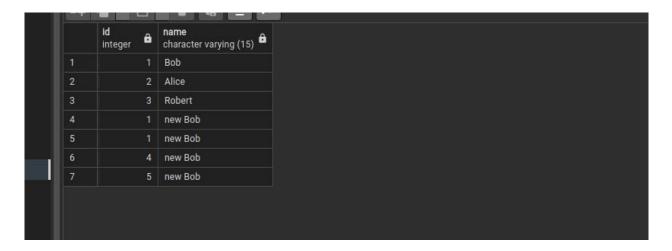
INSERT 0 1

Query returned successfully in 123 msec.
```

Inserted new row, so 5 and 'new Bob' values are going to be values for person table also, so it's insering there as well.



Users table after inserting and person table.



In other words users table will contain person table in it.

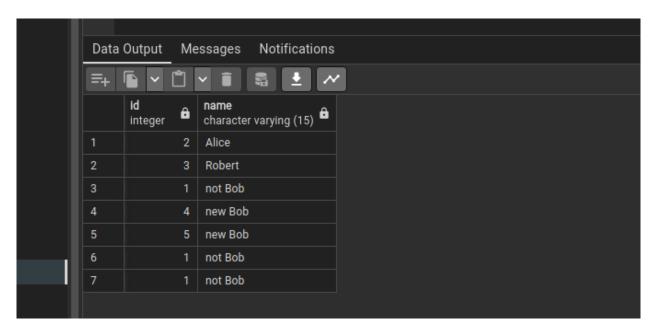
```
31
32
33
34 UPDATE person
35 SET name = 'not Bob'
36 where id = 1;

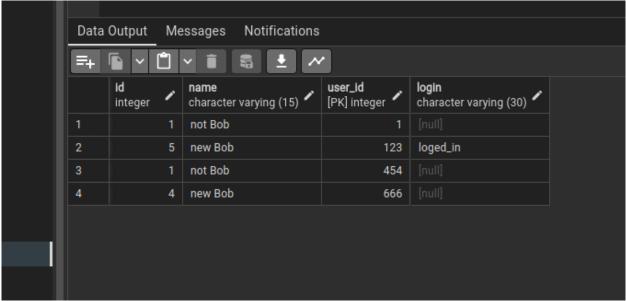
Data Output Messages Notifications

UPDATE 3

Query returned successfully in 237 msec.
```

Update statement updates all new Bobs to not Bobs in both tables .





Alter person.

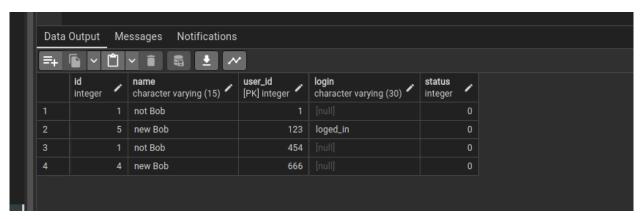
```
37
38
39
ALTER TABLE person ADD COLUMN status integer DEFAULT 0;
40
ALTER TABLE person ADD CONSTRAINT status CHECK (status in (0,1)) NO
1NHERIT;
42
ALTER TABLE person ADD CONSTRAINT id UNIQUE (id, name);

Data Output Messages Notifications

ALTER TABLE

Query returned successfully in 175 msec.
```

We can see that by Alterin person table users table were altered as well



2.

Table created and populated with 15776640 values.

666 MB's

```
INSERT INTO test_index(num, load_date)

SELECT random(), x

FROM generate_series('2017-01-01 0:00'::timestamptz,

'2021-12-31 23:59:59'::timestamptz, '10 seconds'::interval) x;

SELECT pg_size_pretty(pg_relation_size('test_index'));

Data Output Messages Notifications

Pg_size_pretty
text

1 6666 MB
```

Took 2.6seconds.

```
ORDER BY 1;
68
69
70
CREATE INDEX test_index_date ON test_index (load_date);

Data Output Messages Notifications

CREATE INDEX
Query returned successfully in 17 secs 484 msec.
```

Creating index took 17.4 seconds and size is 338MB.

```
CREATE INDEX test_index_date ON test_index (load_date);

SELECT pg_size_pretty(pg_relation_size('test_index_date'));

Data Output Messages Notifications

SHOW FOR SIZE_PRETTY FIRST CONTROL FOR SIZE ('Test_index_date'));

Pg_size_pretty First Control Firs
```

Business Template

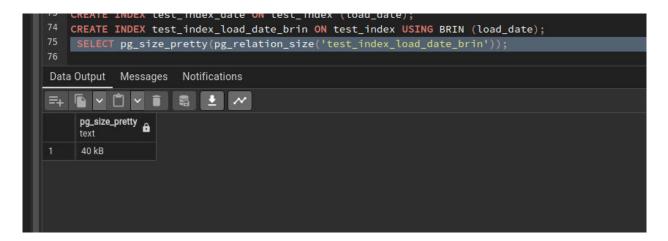
```
71 SELECT pg_size_pretty(pg_relation_size('test_index_date'));
72 DROP INDEX test_index_date;
73 CREATE INDEX test_index_date ON test_index (load_date);
74 CREATE INDEX test_index_load_date_brin ON test_index USING BRIN (load_date);
75

Data Output Messages Notifications

CREATE INDEX

Query returned successfully in 7 secs 682 msec.
```

BRIN index created took 7.6 sec and it's just 40kb.



3.

Table created and populated with 10.000.000 values, size 651 MB

```
81
82 CREATE TABLE test_index AS SELECT id, md5(id::text) as t_hash
83 FROM generate_series(1, 10000000) AS id;
84
85 SELECT pg_size_pretty(pg_relation_size('test_index'));
86

Data Output Messages Notifications

pg_size_pretty
text

1 651 MB
```

```
Total rows: 1 of 1 Query complete 00:00:02:509
```

2.5 second to find pattern.

```
Total rows: 1 of 1 Query complete 00:13:35.696
```

Creation gist index took 13 minutes:(

Business Template

```
SELECT pg_size_pretty(pg_relation_size('test_index'));

SELECT * FROM test_index WHERE t_hash LIKE '%ceeal67a5a%';

REATE EXTENSION pg_trgm;

GREATE INDEX idx_text_index_gist ON test_index USING gist(t_hash gist_trgm_ops);

DROP INDEX idx_text_index_gist;

GREATE INDEX idx_text_index_gin ON test_index USING gin (t_hash gin_trgm_ops);

Data Output Messages Notifications

CREATE INDEX

Query returned successfully in 4 min 31 secs.

Total rows: 1 of 1 Query complete 00:04:31.598
```

With gin index it took 4minutes and size is 613 MB

```
SELECT pg_size_pretty(pg_relation_size('test_index'));

SELECT * FROM test_index WHERE t_hash LIKE '%ceeal67a5a%';

CREATE EXTENSION pg_trgm;

CREATE INDEX idx_text_index_gist ON test_index USING gist(t_hash gist_trgm_ops);

DROP INDEX idx_text_index_gist;

CREATE INDEX idx_text_index_gin ON test_index USING gin (t_hash gin_trgm_ops);

Data Output Messages Notifications

CREATE INDEX

Query returned successfully in 4 min 31 secs.

Totalrows: 1 of 1 Query complete 00:04:31.598
```

Population table this time took 1 minute which is way faster then first time.

4.

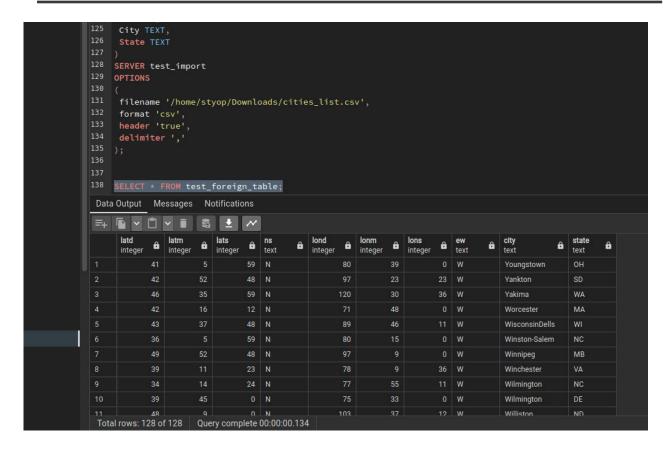
```
113
114
115 CREATE FOREIGN TABLE labs.test_foreign_table
(117 LatD INT,
118 LatW INT,
119 LatS INT,
120 NS TEXT,
121 Lond INT,
122 Lond INT,
123 Lons INT,
124 EW TEXT,
125 City TEXT,
126 State TEXT
127 )
128 SERVER test_import
129 OPTIONS
130 (
131 filename '/home/styop/Downloads/ities_list.csv',
132 format 'csv',
133 header 'true',
134 delimiter ','
135 );

Data Output Messages Notifications

CREATE FOREIGN TABLE

Query returned successfully in 115 msec.
```

Foreign table created using csv file.



There is 128 rows.

Materialized view created with 128 rows.



I deleted 2 rows from file but the count in materialized view remains same, as we store our previous data in materialized view (it means we copy the file in the view)

```
format 'csv',
header 'true',
delimiter ','
135
);
136
137
138
SELECT * FROM test_foreign_table;
139
140
SELECT count(*) FROM test_foreign_table;
141
142
143
CREATE MATERIALIZED VIEW mview as ( SELECT * FROM test_foreign_table);
select count(*) from mview

Data Output Messages Notifications

The Count bigint a

1 128

Total rows: 1 of 1 Query complete 00.00.00.210
```