Range Partitioning

Let us understand how we can take care of range partitioning of tables.

- It is primarily used to create partitions based up on a given range of values.
- Here are the steps involved in creating table using range partitioning strategy.
  - Create table using `PARTITION BY RANGE`
  - Add default and range specific partitions
  - Validate by inserting data into the table
- We can detach as well as drop the partitions from the table.

Create Partitioned Table

Let us create partitioned table with name `users_range_part`.

- It contains same columns as `users`.
- We will partition the table based up on `created_dt` field.
- We will create one partition per year with naming convention **users_range_part_yyyy** (users_range_part_2016).

```
%load_ext sql
```

```
The sql extension is already loaded. To reload it, use:
  %reload_ext sql
```

```
%env DATABASE_URL=postgresql://itversity_sms_user:sms_password@localhost:5432/itversity_sms_db
```

```
env:
DATABASE_URL=postgresql://itversity_sms_user:sms_password@localhost:5432/itversity_sms_db
```

```
%sql DROP TABLE IF EXISTS users_range_part
```

```
 * postgresql://itversity_sms_user:***@localhost:5432/itversity_sms_db
Done.
```

```
[]
```

```
%%sql

CREATE TABLE users_range_part (
    user_id SERIAL,
    user_first_name VARCHAR(30) NOT NULL,
    user_last_name VARCHAR(30) NOT NULL,
    user_email_id VARCHAR(50) NOT NULL,
    user_email_validated BOOLEAN DEFAULT FALSE,
    user_password VARCHAR(200),
    user_role VARCHAR(1) NOT NULL DEFAULT 'U', --U and A
    is_active BOOLEAN DEFAULT FALSE,
    created_dt DATE DEFAULT CURRENT_DATE,
    last_updated_ts TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    PRIMARY KEY (created_dt, user_id)
) PARTITION BY RANGE(created_dt)
```

```
 * postgresql://itversity_sms_user:***@localhost:5432/itversity_sms_db
Done.
```

```
[]
```

Note

We will not be able to insert the data until we add at least one partition.

By Durga Gadiraju