Laboratorio

En este laboratorio comenzarás a practicar con las tablas particionadas

Nota:

Las tareas se realizará sobre una máquina virtual VMWare y un sistema operativo Ubuntu.

Práctica 1: Crear tablas particionadas

Objetivos

El objetivo de la práctica es la creación de tablas particionadas.

Notas

Para utilizar la máquina virtual puede utilizar el software gratuito VMWare Player.

Todas las tareas deberán ser ejecutadas en la consola de Linux (Aplicación Terminal).

Tareas

1.- Particionar la tabla stock de la base de datos bd1. Dado que sólo contiene datos de 2016, se ha de particionar por trimestres, dando lugar a las tablas stock_2016_t{1,2,3,4}. Crear las tablas hijas con sus restricciones apropiadas. Nota: el formato de fecha debe ser YYYY-M-D.

2.- Copiar los datos, según correspondan, de la tabla padre a las tablas hijas.

3.- Borrar los datos de la tabla padre, de forma que quede vacía.

4.- Crear regla(s) en la tabla padre de forma que las inserciones en la misma se redirijan de forma apropiada a las tablas hijas.

5.- Verificar que las reglas anteriores funcionan adecuadamente probando a insertar algún dato adicional en la tabla stock. Verificar también que en la tabla stock padre sigue sin haber registros.

6.- Borrar las reglas anteriormente creadas. Crear en su lugar un trigger (programado en plpgsql) que realice la misma función.

1.- Soluciones

1.Particionar la tabla stock de la base de datos bd1. Dado que sólo contiene datos de 2016, se ha de particionar por trimestres, dando lugar a las tablas stock_2016_t{1,2,3,4}. Crear las tablas hijas con sus restricciones apropiadas. Nota: el formato de fecha debe ser YYYY-M-D.

```
CREATE TABLE stock_2016_t1 (CHECK (day >='2016-1-1' AND day < '2016-1-1'::date +
interval '3 months')) INHERITS (stock);
CREATE TABLE stock_2016_t2 (CHECK (day >='2016-4-1' AND day < '2016-4-1'::date +
interval '3 months')) INHERITS (stock);
CREATE TABLE stock_2016_t3 (CHECK (day >='2016-7-1' AND day < '2016-7-1'::date +
interval '3 months')) INHERITS (stock);
CREATE TABLE stock_2016_t4 (CHECK (day >='2016-10-1' AND day < '2016-10-1'::date +
interval '3 months')) INHERITS (stock);
```

2.Copiar los datos, según correspondan, de la tabla padre a las tablas hijas.

```
INSERT INTO stock_2016_t1 SELECT * FROM stock WHERE day >='2016-1-1' AND day <
'2016-1-1'::date + interval '3 months';
INSERT INTO stock_2016_t2 SELECT * FROM stock WHERE day >='2016-4-1' AND day <
'2016-4-1'::date + interval '3 months';
INSERT INTO stock_2016_t3 SELECT * FROM stock WHERE day >='2016-7-1' AND day <
'2016-7-1'::date + interval '3 months';
INSERT INTO stock_2016_t4 SELECT * FROM stock WHERE day >='2016-10-1' AND day <
'2016-10-1'::date + interval '3 months';
```

3.Borrar los datos de la tabla padre, de forma que quede vacía.

```
DELETE FROM ONLY stock;          -- muy importante ONLY o se perderán todos los datos
DELETE 2623
```

4.Crear regla(s) en la tabla padre de forma que las inserciones en la misma se redirijan de forma apropiada a las tablas hijas.

```
CREATE RULE stock_2016_t1_insert_rule AS ON INSERT TO stock WHERE day >='2016-1-1'
AND day < '2016-1-1'::date + interval '3 months' DO INSTEAD INSERT INTO stock_2016_t1
VALUES (NEW.*);
CREATE RULE stock_2016_t2_insert_rule AS ON INSERT TO stock WHERE day >='2016-4-1'
AND day < '2016-4-1'::date + interval '3 months' DO INSTEAD INSERT INTO stock_2016_t2
VALUES (NEW.*);
```

```
CREATE RULE stock_2016_t3_insert_rule AS ON INSERT TO stock WHERE day >='2016-7-1'
AND day < '2016-7-1'::date + interval '3 months' DO INSTEAD INSERT INTO stock_2016_t3
VALUES (NEW.*);
CREATE RULE stock_2016_t4_insert_rule AS ON INSERT TO stock WHERE day >='2016-10-1'
AND day < '2016-10-1'::date + interval '3 months' DO INSTEAD INSERT INTO stock_2016_t4
VALUES (NEW.*);
```

5.Verificar que las reglas anteriores funcionan adecuadamente probando a insertar algún dato adicional en la tabla stock. Verificar también que en la tabla stock padre sigue sin haber registros.

```
INSERT INTO stock VALUES ('JAVA', '2016-3-20', 0.2, 1.0, 0.1, 0.7, 1000, 0.7);
INSERT INTO stock VALUES ('JAVA', '2016-9-26', 0.2, 1.0, 0.1, 0.7, 1000, 0.7);

SELECT count(*) FROM ONLY stock;
 count
-------
     0

SELECT count(*) FROM stock_2016_t1 WHERE (company, day, price_open, price_high,
price_low, price_close, volume, price_adj_close) = ('JAVA', '2016-3-20', 0.2, 1.0, 0.1, 0.7,
1000, 0.7);
 count
-------
     1

SELECT count(*) FROM stock_2016_t WHERE (company, day, price_open, price_high,
price_low, price_close, volume, price_adj_close) = ('JAVA', '2016-9-26', 0.2, 1.0, 0.1, 0.7,
1000, 0.7);
 count
-------
     1
```

6. Borrar las reglas anteriormente creadas. Crear en su lugar un trigger (programado en plpgsql) que realice la misma función.

```
DROP RULE stock_2016_t1_insert_rule ON stock;
DROP RULE stock_2016_t2_insert_rule ON stock;
DROP RULE stock_2016_t3_insert_rule ON stock;
DROP RULE stock_2016_t4_insert_rule ON stock;
```

```
CREATE FUNCTION stock_insert_trigger() RETURNS TRIGGER AS $$
        BEGIN
                IF extract(year FROM NEW.day) = 2016 THEN
                        EXECUTE 'INSERT INTO stock_2016_t' || date_part('quarter', NEW.day) ||
' VALUES ($1.*)' USING NEW;
                END IF;

                RETURN NULL;
        END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trigg_stock_insert BEFORE INSERT ON stock FOR EACH ROW EXECUTE
PROCEDURE stock_insert_trigger();
```