Time Machine Implementation Details

July 2020

Amin Rezaei

In this document the Time Machine implementation is described.

We will use some special features of PostgreSQL:

- tstzrange datatype to represent date intervals
- Views / Updateable Views
- Exclusion Constraints in btree_gist

Descriptions:

For each Table in source we add a __lifetime column to destination Table with type tstzrange, it proposes that in which time interval this record was valid. We can't use unique constraint on ID because of records that are valid with same id in different times, to resolve this we will utilize Exclusion Constraints declared in btree_gist like this:

```
EXCLUDE USING gist (ID WITH =, __lifetime WITH &&)
```

This constraint ensures that no record with same id and colliding interval exists. ID is checked via equality operator (=) and __lifetime is checked with && operator that is declared to show that ranges are colliding. This is exactly what we need for a time machine.

Two Views are created as following for each table:

```
--! TimeMachine View:
CREATE VIEW ${table}_t AS
    SELECT  ${cols}
    FROM    ${table}
    WHERE   current_setting('time_machine.time')::tstzrange <@ __lifetime;
--! TimeMachine View Now:
CREATE VIEW ${table}_now AS
    SELECT  ${cols}
    FROM    ${table}
    WHERE   current_timestamp <@ __lifetime;
```

First View represents table at a special time that is determined via a parameter and Second one represents current state of table.

These views follow Updateable View rules and all Update/Insert/Delete operations can be done on them that will result on main table. We use this option and declare a Trigger for view in current time for each table. All operations discussed will be done on current time view instead of main table and Trigger will manage time machine versioning.

```
--! TimeMachine Func:
CREATE FUNCTION time_machine_versioning_${table}() RETURNS TRIGGER AS
$$
BEGIN
    IF TG_OP = 'UPDATE'
    THEN
        IF ${confliction_check} THEN
            RAISE EXCEPTION 'ID Cols is considered to be constant and not changed during
time!';
        END IF;

        UPDATE  ${table} SET __lifetime = tstzrange(lower(__lifetime),
current_timestamp)
        WHERE   ${id_search} AND current_timestamp <@ __lifetime;

        IF NOT FOUND THEN
            RETURN NULL;
        END IF;
    END IF;

    IF TG_OP IN ('INSERT', 'UPDATE')
    THEN
        INSERT INTO ${table} (${id_cols}, __lifetime ${cols}) VALUES
            (${id_cols_new}, tstzrange(current_timestamp, TIMESTAMPTZ 'infinity')
${col_datas});
        RETURN NEW;
    END IF;

    IF TG_OP = 'DELETE'
    THEN
        UPDATE ${table} SET __lifetime = tstzrange(lower(__lifetime), current_timestamp)
        WHERE ${id_search_old} AND current_timestamp <@ __lifetime;
        IF FOUND THEN
            RETURN OLD;
        ELSE
            RETURN NULL;
        END IF;
    END IF;
END;
$$ LANGUAGE plpgsql;
--! TimeMachine Trigger:
CREATE TRIGGER ${table}_trigger
    INSTEAD OF INSERT OR UPDATE OR DELETE
    ON ${table}_now
    FOR EACH ROW
    EXECUTE PROCEDURE time_machine_versioning_${table}();
```

Define function manages applying the changes and versioning , Trigger captures the operations and uses the function to manage them.

Now, Time machine is fully implemented. For accessing current state, views with suffix _now can be used. For accessing state in a particular time, first special time parameter should be set in PostgreSQL like:

```
--! Set TimeMachine Time:
SET time_machine.time = '2020-01-01 00:00:00';
```

After setting the parameter, view with suffix _t can be used to achieve state at desired time.

Used References:

- https://www.postgresql.org/docs/12/rangetypes.html
- https://www.postgresql.org/docs/12/functions-range.html
- https://www.postgresql.org/docs/current/ddl-constraints.html
- https://www.postgresql.org/docs/10/plpgsql-declarations.html
- https://www.cybertec-postgresql.com/en/implementing-as-of-queries-in-postgresql/
- https://www.postgresql.org/docs/current/indexes-types.html
- https://www.postgresql.org/docs/9.1/plpgsql-trigger.html