به نام خدا

کوروش مسلمی-۹۸۱۳۰۲۷

ابتدا با یک مثال چگونگی سفر به گذشته، با پیاده سازی انجام شده را نشان می دهیم سپس جزئیات آن را شرح می دهیم.

در ابتدا جدول زیر را در دیتابیس مبدا داریم:

| | translatorId integer | translatorName text |
|---|---|---|
| 1 | 1 | Koorosh |
| 2 | 2 | Ali |
| 3 | 3 | Nima |

رکورد دوم را از جدول حذف، رکورد سوم را ویرایش و یک رکورد با آیدی ۴ اضافه می کنیم. در انبار داده اطلاعات به این صورت ذخیره می شوند:

| | translatorId integer | translatorName text | __valid tstzrange |
|---|---|---|---|
| 1 | 1 | Koorosh | ["2021-07-10 18:12:23.969967+04:30",infinity) |
| 2 | 4 | Jafar Nezhad Ghomi | ["2021-07-10 18:38:10.455973+04:30",infinity) |
| 3 | 2 | Ali | ["2021-07-10 18:12:23.969967+04:30","2021-07-10 18:38:10.465361+04:30") |
| 4 | 3 | Nima | ["2021-07-10 18:12:23.969967+04:30","2021-07-10 18:38:10.476349+04:30") |
| 5 | 3 | Nimaa | ["2021-07-10 18:38:10.476349+04:30",infinity) |

همانطور که مشاهده می شود هیچ رکوردی از انبار داده حذف نشده است. تنها کران بالای ستون __valid در رکورد متناظر با رکوردی که از دیتابیس مبدا حذف شده بود از infinity به مقداری دیگر تغییر کرد. رکوردی که آپدیت شده بود نیز به همین شکل با این تفاوت که رکورد جدیدی که حاوی مقدار جدید بود با کران بالای infinity اضافه شد. رکورد با آیدی چهار نیز به طور معمول با کران بالای infinity اضافه شده است. در حقیقت ستون __valid بازه ای که هر رکورد در دیتابیس مبدا معتبر بوده است را ذخیره می کند. اکنون در دیتابیس مقصد می توانیم به قبل تغییر سفر کرده و به جدول اولیه برسیم:

```
3  SET timerobot.as_of_time = '2021-07-10 18:13:00';
4  select * from Translator_historic;
```

Data Output | Explain | Messages | Notifications

| | translatorId integer | translatorName text |
|---|---|---|
| 1 | 1 | Koorosh |
| 2 | 2 | Ali |
| 3 | 3 | Nima |

در اینجا یک زمان مشخص را تنظیم کردیم و اطلاعات جدول را از ویویی مشخص خواندیم. این در حالی است که وضعیت فعلی این جدول در دیتابیس مبدا متفاوت است. می توانیم این وضعیت را مستقیم از دیتابیس مبدا پرس و جو کنیم و یا در دیتابیس مقصد از ویویی دیگر آن را مشاهده کنیم:

```
6  select * from Translator_recent;
```

Data Output | Explain | Messages | Notifications

| | translatorId integer | translatorName text |
|---|---|---|
| 1 | 1 | Koorosh |
| 2 | 4 | Jafar Nezhad Ghomi |
| 3 | 3 | Nimaa |

در حالتی که کاربر امکان سفر به گذشته را در برنامه نوشته شده فعال کرده باشد متناظر با هر جدول در دیتابیس مبدا، یک جدول در دیتابیس مقصد با ستون اضافه valid__ از نوع tstzrange، دو ویو با پسوند های historic و recent ، یک تریگر برای مدیریت حذف، درج و ویرایش روی ویو recent به ازای هر جدول ساخته می شود.

مجموعه عملیات هایی که پس از اعمال تغییر در جدول Translator دیتابیس مبدا در انبارداده رخ داد با تریگری به صورت زیر پیاده سازی شده است:

```
BEGIN
    IF TG_OP = 'UPDATE'
    THEN
        IF NEW."translatorId" <> OLD."translatorId"
        THEN
            RAISE EXCEPTION 'the ID must not be changed';
        END IF;
        UPDATE  "Translator"
        SET     __valid = tstzrange(lower(__valid), current_timestamp)
        WHERE   "translatorId" = NEW."translatorId"
            AND current_timestamp <@ __valid;
        IF NOT FOUND THEN
            RETURN NULL;
        END IF;
    END IF;
    IF TG_OP IN ('INSERT', 'UPDATE')
    THEN
        INSERT INTO "Translator" ("translatorId", __valid, "translatorName")
            VALUES (NEW."translatorId",
                tstzrange(current_timestamp, TIMESTAMPTZ 'infinity'),
                NEW."translatorName");
        RETURN NEW;
    END IF;
    IF TG_OP = 'DELETE'
    THEN
        UPDATE  "Translator"
        SET     __valid = tstzrange(lower(__valid), current_timestamp)
        WHERE "translatorId" = OLD."translatorId"
            AND current_timestamp <@ __valid;
        IF FOUND THEN
            RETURN OLD;
        ELSE
            RETURN NULL;
        END IF;
    END IF;
END;
```

لازم به ذکر است این تابع به صورت داینامیک برای هر جدول در صورت فعال بودن versioning توسط کد پایتون تعریف می شود  و به جای هر عملیات درج/حذف/ویرایش روی ویو recent جدول مربوطه اجرا می شود.

از دیگر نکات مهم این پیاده سازی استفاده از exclusion constraint است:

```
CREATE TABLE public."Translator"
(
    "translatorId" integer NOT NULL,
    "translatorName" text COLLATE pg_catalog."default" NOT NULL,
    __valid tstzrange,
    CONSTRAINT "Translator_translatorId___valid_excl" EXCLUDE USING gist (
        translatorId WITH =,
        __valid WITH &&)

)
```

که برای مثال در جدول فوق تصمین می کند رکورد هایی که translatorId برابری دارند در ستون __valid همپوشانی نداشته باشند.

**منابع:**


1) https://www.cybertec-postgresql.com/en/implementing-as-of-queries-in-postgresql/
2) https://www.postgresql.org/docs/current/ddl-constraints.html