Übung 9: Trigger und Funktionen („Bonus“)

Ziele:

- Trigger anwenden
- Einfache PL/pgSQL-Funktionen selber programmieren

Zeit:

- Ca. 40 Min.

Werkzeuge:

- pgAdmin4 und QGIS mit DB-Manager
- PostGIS Manual: https://postgis.net/documentation/manual/
- PostgreSQL-Dokumentation: https://www.postgresql.org/docs/current/interactive/index.html
- PL/pgSQL-Referenz: https://www.postgresql.org/docs/current/interactive/plpgsql.html , Tutorial https://www.postgresqltutorial.com/postgresql-plpgsql/ , Slides plpgsql.pdf (postgresconf.org)

Aufgabe 1: Row-Level Trigger und Trigger-Funktion zur automatischen Berechnung der Fläche

Erstellen Sie im Schema „schweiz“ eine neue Tabelle „flaechen“ mit POLYGON Geometrie und mindestens den folgenden Attributen:

- gid (primary key, Typ „serial“)
- flaeche (Typ „numerical“)
- erstellungs_datum (Typ timestamp without time zone)
- aenderungs_datum (Typ timestamp without time zone)

Erstellen Sie nun einen row-level „BEFORE INSERT OR UPDATE“ Trigger und eine Trigger-Funktion die beim Neuanlegen und Aktualisieren eines Objekts automatisch die Fläche neu berechnet. Ausserdem soll beim Erstellen das Erstellungsdatum und beim Ändern das Änderungsdatum gesetzt werden.

Testen Sie Ihre Trigger und die Trigger-Funktion indem Sie die Tabelle in QGIS laden und Flächen erfassen und speichern. Kontrollieren Sie ob die Werte aktualisiert werden. Laden Sie allenfalls einen anderen Layer oder einen WMS-Layer als „Hintergrundlayer“ hinzu.

Aufgabe 2: Funktion zur Extraktion von Abwasserleitungsinformationen innerhalb einer Parzelle

Schreiben Sie eine Funktion die als Parameter eine Parzellennummer entgegennimmt, eine Abwassernutzungsart, sowie einen Besitzer ('Stadt Uster' oder 'Private Erschliessung'). Als Ergebnis soll die geclippte, aufsummierte Haltungslänge der entsprechenden Parzelle und Nutzungsart, optional gefiltert nach Baujahr extrahiert werden. Das Resultat soll auf Meter Haltungslänge gerundet werden. Schreiben Sie die Funktion entweder in SQL oder einer anderen Scriptingsprache wie Perl oder Python. Speichern Sie die Funktion im Schema „abwasser“.

Nötige Views: av_user.liegenschaften ('gru_nummer') und abwasser.haltung ('nutzungsart', 'eigentuemer')

Testen Sie z.B. mit der Parzelle Nr. 'F1586' und verschiedenen Nutzungsarten ('Schmutzabwasser', 'Regenabwasser', 'entlastetes Mischabwasser')

Aufgabe 3: Funktion zur Extraktion von betroffenen Parzellen eines Gestaltungsplanes

Schreiben Sie eine Funktion die als Parameter den Namen eines Gestaltungsplanes und eine Mindestverschnittfläche in m2 (z.B. 10m2) entgegennimmt. Die Mindestverschnittfläche soll verhindern dass nur berührte oder leicht angeschnittene Parzellen selektiert werden. Als Ausgabe soll eine Liste der betroffenen Parzellen ('gru_nummer') zurückgeliefert werden (Recordset).

HINWEIS: die Gestaltungsplanteilflächen müssen vor dem Verschnitt erst mit ST_UNION zusammengefasst werden.

Nötige Views: raumplanung.nutzungszonen (zonenbez_gemeinde, bemerkung) und av_user.liegenschaften ('gru_nummer')

Testen Sie z.B. mit dem Gestaltungsplan mit der Bemerkung 'GP Müliholz' oder 'GP Turicum'

Lösungen

Aufgabe 1:

Row-Level trigger für Tabelle schweiz.flaechen:

```
CREATE TABLE schweiz.flaechen
(
  gid serial NOT NULL,
  flaeche numeric,
  erstellungs_datum timestamp without time zone,
  aenderungs_datum timestamp without time zone,
  name character varying(40),
  geom geometry(Polygon,21781),
  CONSTRAINT flaechen_pkey PRIMARY KEY (gid)
);

CREATE INDEX ch_flaechen_geom_gist
  ON schweiz.flaechen
  USING gist
  (geom);


CREATE OR REPLACE FUNCTION schweiz.update_flaechen()
  RETURNS trigger AS
$BODY$
  BEGIN
    NEW.flaeche := ST_Area(NEW.geom);
    IF (TG_OP = 'INSERT') THEN
            NEW.erstellungs_datum := now();
      END IF;
    IF (TG_OP = 'UPDATE') THEN
            NEW.aenderungs_datum := now();
      END IF;
    RETURN NEW;
  END;
$BODY$
  LANGUAGE plpgsql;

CREATE TRIGGER trigger_update_flaechen
  BEFORE INSERT OR UPDATE
  ON schweiz.flaechen
  FOR EACH ROW
  EXECUTE PROCEDURE schweiz.update_flaechen();
```

Aufgabe 2:

Funktion zur Extraktion von Abwasserleitungsinformationen innerhalb einer Parzelle

```
CREATE FUNCTION
  abwasser.haltungslaenge_in_parzelle (gru_nummer text, nutzungsart text,
eigentuemer text)
RETURNS numeric AS $$
  SELECT round(SUM(ST_Length(ST_Intersection(l.the_geom,h.the_geom))))::numeric
   FROM abwasser.haltung h, av_user.liegenschaften l
   WHERE ST_Intersects(l.the_geom,h.the_geom)
    AND l.gru_nummer = $1 AND h.nutzungsart = $2 AND h.eigentuemer = $3
   GROUP BY l.the_geom;
```

```
$$ LANGUAGE SQL STABLE;
```

Aufruf 1:
```
SELECT abwasser.haltungslaenge_in_parzelle('F1586','Schmutzabwasser','Private
Erschliessung');
```

Resultat 1:
```
110
```

Aufruf 2:
```
SELECT abwasser.haltungslaenge_in_parzelle('F1586','Regenabwasser','Private
Erschliessung');
```

Resultat 2:
```
160
```

Aufruf 3:
```
SELECT abwasser.haltungslaenge_in_parzelle('B5267','Mischabwasser','Stadt
Uster')
```

Resultat 3:
```
124
```

Aufgabe 3:

Funktion zur Extraktion von Parzellen innerhalb eines Gestaltungsplanes

```
CREATE FUNCTION raumplanung.parzellen_in_gp (gp_name text, min_flaeche numeric, OUT
gru_nummer text)
    RETURNS SETOF text AS $$
SELECT gru_nummer::text
  FROM av_user.liegenschaften l,
  (SELECT ST_Union(the_geom) AS the_geom FROM raumplanung.nutzungszonen WHERE
zonenbez_gemeinde = 'RG' AND bemerkung = $1) n
WHERE ST_Intersects(l.the_geom,n.the_geom) AND
  ST_Area(ST_Intersection(l.the_geom,n.the_geom)) > min_flaeche
ORDER BY gru_nummer ASC;
 $$ LANGUAGE SQL STABLE;
```

Aufruf 1 :
```
SELECT raumplanung.parzellen_in_gp('GP Turicum',5);
```

Resultat 1:

```
"C2851"
"C2852"
"C2853"
"C2854"
"C2855"
"C2971"
"C2972"
"C3471"
"C3472"
```

Aufruf 2 :
```
SELECT raumplanung.parzellen_in_gp('GP Müliholz',5);
```

Resultat 2:

```
"B7038"
"B7039"
"B7040"
"B7041"
"B7178"
```