Kapitel 9

Stored Procedures

Stefan Keller

Dank an Dr. Andreas Neumann

Stored Procedures

- «PostgreSQL eats it's own dogfood – everything is catalogue-driven!»
- Nicht nur Tabellen, Views und Spalten, etc. sind im Systemkatalog abgelegt, sondern auch Datentypen, Funktionen, Trigger, Regeln, etc.
- PostgreSQL ist die am besten erweiterbare objektrelationale Datenbank der Welt!
- Andere Datenbanken legen viele Objekte hardcodiert ausserhalb der Datenbank ab und referenzieren sie
- PostgreSQL kann externen Code (z.B. Shared libraries) dynamisch laden wenn sie benötigt werden

4 Arten von Funktionen

- «Query Language Functions» geschrieben in SQL
- «Procedural Language Functions , z.B. PL/pgSQL, PL/Tcl, PL/Perl, PL/Python – interpretierte Scripting-Sprachen
- Interne Funktionen (geschrieben in C, statisch gelinkt) - werden bei der Initialisierung des DB-Clusters erzeugt
- C-language Functions (dynamisch gelinkt/shared libraries) – z.B. PostGIS/GEOS/GDAL
- Zwei oder mehr Funktionen mit dem gleichen Namen können existieren wenn die Parameter unterschiedlich sind → wird z.B. rege von PostGIS-Funktionen genutzt

Einführung in PL/pgSQL

- PL/pgSQL ist die prozedurale Programmiersprache für PostgreSQL-Datenbanken
- Erweitert SQL um prozedurale Elemente wie Variablen, Kontrollstrukturen und Funktionen
- Ermöglicht die Erstellung von Stored Procedures und Funktionen direkt in der Datenbank
- Syntax ähnelt der von Oracle PL/SQL, aber mit PostgreSQL-spezifischen Erweiterungen; inspiriert von Ada

PL/pgSQL – Grundlegende Struktur

```
CREATE FUNCTION function_name(parameter_list)
RETURNS return_type AS $$
DECLARE
  -- Variablendeklarationen
BEGIN
  -- Funktionskörper
  -- SQL-Anweisungen und prozedurale Logik
  RETURN result;
END;
$$
LANGUAGE plpgsql;
```

PL/pgSQL – Wichtige Sprachelemente

- Variablen: Deklariert mit Datentyp (z.B. my_var INTEGER;)
- Rekursive Funktionen
- Kontrollstrukturen:
  - IF-THEN-ELSE
  - CASE
  - LOOP, WHILE LOOP, FOR LOOP
- Cursor: Für die Verarbeitung von Abfrageergebnissen
- Zugriff auf SQL: Direkte Einbettung von SQL-Anweisungen, dynamisches SQL mit EXECUTE
- Transaktionsmanagement mit BEGIN, COMMIT, und ROLLBACK
- Ausnahmebehandlung: Mit EXCEPTION-Block

PL/pgSQL – Besonderheiten und Vorteile

- Enge Integration mit PostgreSQL-Datenbanksystem
- Unterstützung für komplexe Datentypen wie Arrays und Records
- Möglichkeit, Trigger und benutzerdefinierte Funktionen zu erstellen
- Verbesserte Leistung durch serverseitige Ausführung

PL/pgSQL – Best Practices

- Verwende Parametervalidierung zur Erhöhung der Robustheit
- Verwende Präfixe wie „p_“ vor Parameternamen und „v_“ vor Variablennamen
- Nutze Kommentare für bessere Wartbarkeit
- Implementiere eine gute Ausnahmebehandlung
- Nutze RAISE NOTICE für Debugging-Zwecke während der Entwicklung
- Lagere wiederkehrenden Code zu einer (Hilfs-)Funktion aus
- Teile komplexe Funktionen in kleinere, wiederverwendbare Funktionen auf
- Vermeide unnötige Cursor-Operationen, nutze stattdessen Set-basierte Operationen (Tabellen)

Beispiel PL/SQL- und PL/Python-Funktion

PL/SQL

```
CREATE FUNCTION av_user.gbkreisflaeche (gbkreis text)
    RETURNS numeric AS $$
     SELECT SUM(ST_Area(the_geom))::numeric
       FROM av_user.liegenschaften
       WHERE grundbuchkreis = $1;
 $$ LANGUAGE SQL STABLE;
```

Aufruf:

```
SELECT av_user.gbkreisflaeche('Niederuster');
SELECT av_user.gbkreisflaeche('Kirchuster‘);
```

PL/Python

```
CREATE FUNCTION pymax (a integer, b integer)
  RETURNS integer AS $$
  if a > b:
    return a
  return b
$$ LANGUAGE plpythonu;
```

PL/pgSQL - Weitere Informationen

- PL/pgSQL-Referenz: www.postgresql.org/docs/current/interactive/plpgsql.html
- PL/pgSQL-Tutorial: www.postgresqltutorial.com/postgresql plpgsql/
- Slides aus Vortag zu «Introduction to PL/pgSQL» (2019): https://postgresconf.org/system/events/document/000/001/086/plpgsql.pdf