DataCraft

10
Advanced SQL

DataCraft

- Leerzeichen am rechten und linken Rand entfernen

```
trim(string)      -- alternativ rtrim, ltrim
```

- Einen String auftrennen

```
split_part(string, trennzeichen, nr)
```

- Einen String anhand eines Trennzeichen in ein Array umwandeln

```
string_to_array(string, trennzeichen)
```

Arrays sind geordnete Listen (Vektoren) bzw. höherdimensional Matrizen, …

- Definition eines Arrays mit eckigen Klammern

```
telefon text[]
```

- Einfügen von Werten mit dem Schlüsselwort ARRAY

```
insert into kontakte (telefon) values (ARRAY ['123','456','789']);
```

- Zugriff auf einzelne Werte (Zähung beginnt bei 1)

```
select telefon[1] from kontakte
```

- Wert in Array enthalten

```
where '123' = any(telefon)
```

- Array in Zeilen auftrennen

```
select unnest(telefon) from kontakte
```

- siehe https://www.postgresql.org/docs/current/arrays.html

DataCraft

Reguläre Ausdrücke in PostgreSQL

- Vergleich mit ~ (Tilde)

```
where string ~ '[a-z]*'
```

- Ersetzen eines Musters

```
regexp_replace(string, pattern, ersatz, 'g')
-- Parameter 'g' um alle Vorkommen zu ersetzen
```

- Finden eines Musters

```
regexp_matches(string, pattern)
-- Parameter 'g', um jedes Vorkommen in neue Zeile zu schreiben
```

- siehe https://www.postgresql.org/docs/current/functions-matching.html#FUNCTIONS-POSIX-REGEXP

Ändern sich Tabellen des Quellsystems häufig oder liegen Daten direkt im JSON-Format vor, unterstützt PostgreSQL das Datenformat JSON

```
CREATE TABLE orders (
        id serial NOT NULL PRIMARY KEY,
        info json NOT NULL
);

INSERT INTO orders (info)
VALUES('{ "customer": "Magda", "items": {"product": "Bleistift","qty": 24}}'),
      ('{ "customer": "Nina", "items": {"product": "Lineal","qty": 1}}'),
      ('{ "customer": "Otto", "items": {"product": "Radiergummi","qty": 2}}');

SELECT
  info ->> 'customer' as customer,
  (info -> 'items' ->> 'qty')::int as quantity
from orders;
```

Ein rollierendes Fenster (rolling window) wird dann benötigt, wenn zum Beispiel der Durchschnitt über die letzten 30 Tage gebildet werden soll

```
SELECT
  datum,
  avg(umsatz) OVER (
     ORDER BY datum
     ROWS BETWEEN 29 PRECEDING AND CURRENT ROW) as mw_umsatz_30t
FROM umsaetze;
```

DataCraft

Mit (Stored) **Procedures** können Abfolgen von Befehlen in eine Funktion zusammengefasst werden. Im Gegensatz zu Funktionen haben Procedures keine Rückgabewert.

```
create procedure transfer(sender_id int, receiver_id int, amount dec)
language plpgsql
as $$
begin
    update accounts
    set balance = balance - amount
    where id = sender_id;

    update accounts
    set balance = balance + amount
    where id = receiver_id;

    commit;
end;$$;
```

Es gibt mehrere mögliche Sprachen

Aufruf mit

```
call transfer(10, 11, 19.99)
```

DataCraft

Funktionen haben im Gegensatz zu Procedures einen Rückgabewert.

```
create function transfer(sender_id int, receiver_id int, amount dec) returns void
language plpgsql
as $$
begin
    update accounts
    set balance = balance - amount
    where id = sender_id;

    update accounts
    set balance = balance + amount
    where id = receiver_id;

    commit;
end;$$;
```

Aufruf mit

```
select transfer(10, 11, 19.99)
```

Ein Trigger ist eine Funktion, die aufgerufen wird, wenn bei einer Tabelle ein Ereignis (INSERT, UPDATE, DELETE oder TRUNCATE) passiert

```
CREATE FUNCTION es_passiert() RETURNS TRIGGER
   LANGUAGE PLPGSQL
AS $$
BEGIN
   -- hier kommt der Code rein
END;
$$

CREATE TRIGGER trigger_name
   {BEFORE | AFTER} { event }
   ON table_name
   [FOR [EACH] { ROW | STATEMENT }]
       EXECUTE PROCEDURE es_passiert
```

Mit OLD und NEW kann auf den Zustand vor bzw. nach dem Event zugegriffen werden

```
CREATE OR REPLACE FUNCTION log_last_name_changes() RETURNS TRIGGER
  LANGUAGE PLPGSQL
  AS $$
BEGIN
  IF NEW.last_name <> OLD.last_name THEN
    INSERT INTO employee_audits(employee_id,last_name,changed_on)
         VALUES(OLD.id,OLD.last_name,now());
  END IF;
  RETURN NEW;
END;
$$

CREATE TRIGGER last_name_changes
  BEFORE UPDATE ON employees
  FOR EACH ROW EXECUTE PROCEDURE log_last_name_changes();
```