PostgreSQL – Joins

Stephan Karrer

Beziehungen zwischen Tabellen

Tabelle: `EMPLOYEES`

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	DEPARTMENT_ID
174	Ellen	Abel	80
142	Curtis	Davies	50
102	Lex	De Haan	90
104	Bruce	Ernst	60
202	Pat	Fay	20
206	William	Gietz	110

Primärschlüssel **Fremdschlüssel**

Tabelle: `DEPARTMENTS`

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales	149	2500
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting		1700

Primärschlüssel

- Die Daten werden in der Regel auf mehrere Tabellen verteilt, um Redundanzen zu vermeiden (sog. Normalisierung)
- Der Wert in der Fremdschlüsselspalte der Tabelle "EMPLOYEES" verweist auf den zugehörigen Datensatz (Primärschlüssel) in der Tabelle "DEPARTMENTS"
- Diese Daten wieder zusammenzuführen ist der häufigste Anwendungsfall des Joins

Joins unter Oracle (ANSI 99 konform)

```
FROM first_table <JOIN_TYPE> second_table
     [ON (<JOIN_CONDITION>)]
```

join_type gibt an, welche Art der Verknüpfung

- Oracle-eigene Syntax oder ANSI SQL Syntax ist möglich
- Unterstützte Arten:
  - Inner Join (Equi Join als Spezialform)
  - Self Join
  - Kartesisches Produkt
  - Ein- und zweiseitige Outer Joins
  - Inner und Outer Joins mit beliebigen Bedingungen

Equi-Join (Spezialform des Inner-Join)

```
SELECT  employees.employee_id, employees.last_name,
employees.department_id, departments.location_id
        FROM employees INNER JOIN departments
         ON (employees.department_id = departments.department_id);


SELECT  e.employee_id, e.last_name, e.department_id,
        d.location_id
        FROM employees e JOIN departments d
        ON (e.department_id = d.department_id);
```

Equi-Join über WHERE-KLausel: alte Schreibweise (ANSI 89)

```
SELECT  e.employee_id, e.last_name, e.department_id,
        d.location_id

        FROM employees e, departments d

        WHERE e.department_id = d.department_id

                AND d.location_id > 1000;
```

Mehrfach-Join

```
/* Das Schlüsselwort "ON" muss nach dem jeweiligen "JOIN"
folgen  */


SELECT e.last_name, d.department_name, l.city, c.country_name
  FROM employees e INNER JOIN departments d
                     ON e.department_id = d.department_id
                   INNER JOIN locations l
                     ON d.location_id = l.location_id
                   INNER JOIN countries c
                     ON l.country_id = c.country_id ;
```

Non-Equi Join

```
select e.last_name, e.department_id,

       d.department_id, d.department_name

   from employees e JOIN departments d

         ON e.department_id > d.department_id

        WHERE e.department_id = 50 and e.employee_id = 140;
```

Outer Joins

```
SELECT  e.employee_id, e.last_name, e.department_id,
        d.department_id, d.department_name
        FROM employees e LEFT OUTER JOIN departments d
        ON (e.department_id = d.department_id);

SELECT  e.employee_id, e.last_name, e.department_id,
        d.department_id, d.department_name
        FROM employees e RIGHT OUTER JOIN departments d
        ON (e.department_id = d.department_id);

SELECT  e.employee_id, e.last_name, d.department_id,
        d.department_name
        FROM employees e FULL OUTER JOIN departments d
        ON (e.department_id = d.department_id);
```

Cross Join (Kartesisches Produkt)

```
SELECT  e.employee_id, e.last_name, e.department_id,
        d.department_id, d.department_name
        FROM employees e CROSS JOIN departments d
        WHERE e.department_id = d.department_id
        ORDER BY e.employee_id ;
```

Self Join

```
SELECT  e.last_name AS emp, m.last_name AS man
        FROM employees e INNER JOIN employees m
        ON (e.manager_id = m.employee_id) ;
```

Natural JOIN

```
SELECT e.last_name, d.department_name
FROM   employees e NATURAL JOIN departments d
ORDER BY e.last_name, d.department_name;

-- entspricht
SELECT e.last_name, d.department_name
FROM   employees e JOIN departments d
    ON e.department_id = d.department_id AND
       e.manager_id = d.manager_id
ORDER BY e.last_name, d.department_name;
```

- JOIN erfolgt automatisch über alle gleich benannten Spalten
- Ist zwar ANSI, wird aber eher nicht verwendet

JOIN mit USING-Klausel

```
SELECT e.last_name, d.department_name
FROM   employees e JOIN departments d
       USING (department_id)
ORDER BY e.last_name, d.department_name;
```

- Entspricht dem NATURAL JOIN mit Einschränkung der zu verwendenden Spalten
- Ist zwar ANSI, wird aber eher nicht verwendet