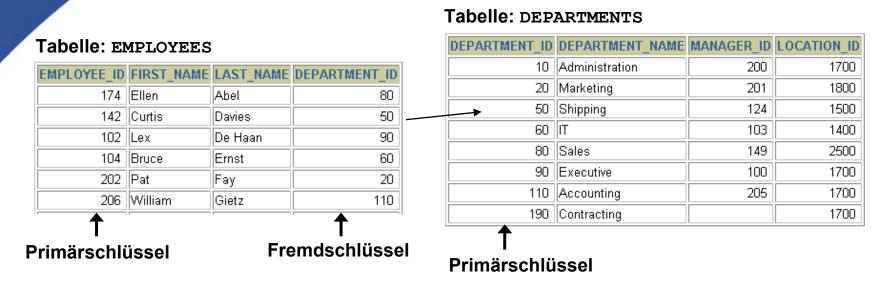
# **SQL** – Joins

Stephan Karrer

### Beziehungen zwischen Tabellen



- 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)

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 1
                     ON d.location id = l.location id
                   INNER JOIN countries c
                     ON l.country id = c.country id ;
```

# Non-Equi Join

#### **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 d.department id, e.last name -- Oracle-spez. Syntax
       FROM departments d, employees e
       WHERE d.department id = e.department id(+)
      ORDER BY d.department id, e.last name;
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

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