

## Arbeiten mit mehreren Tabellen

### Ausgangslage

Das „R“ in „RDBMS“ steht für „Relation“ bzw. „Beziehung“ (zwischen Tabellen). In relationalen Datenbanken werden die Datenmodelle normalisiert. Normalisieren heisst unter Anderem, dass Tabellen, zum Eliminieren von Redundanzen, in mehrere Tabellen aufgeteilt werden. Es muss also meistens mit mehreren Tabellen gearbeitet werden, damit man alle nötigen Informationen bekommt.

Mit dem DML Statement „SELECT“ können mehrere Tabellen gleichzeitig abgefragt werden um zu den nötigen Informationen zu kommen. Die Tabellen werden im Statement „zusammen verlinkt“ oder „gejoined“.

### Testdaten

In den Beispielen werden folgende 3 Tabellen verwendet:

Tabelle „MITARBEITER“:

- MA\_ID
- NAME
- VORNAME
- ADRESSE
- PLZ
- FERIENLAND

Tabelle „ORTE“:

- PLZ
- ORT

Tabelle „LAENDER“:

- L\_ID
- LAND

## Syntax generell

Als Beispiel soll die vollständige Adresse der Mitarbeiters angezeigt werden. Dazu müssen Daten aus zwei Tabellen (MITARBEITER und ORTE) selektiert werden.

Beim Syntax von Joins müssen ein paar Sachen beachtet werden.

- Die FROM Klausel kann eine Komma separierte Liste von Tabellen enthalten

```
SELECT name, vorname, adresse, plz, ort
FROM mitarbeiter, orte
ORDER BY name, vorname
```

- Das Attribut PLZ kommt in beiden Tabellen vor. Wir müssen dem DBMS also sagen welches Attribut wir angezeigt haben wollen. Dies wird erzieht, indem man dem Attribut den entsprechenden Tabellennamen voranstellt:

```
SELECT name, vorname, adresse, mitarbeiter.plz, ort
FROM mitarbeiter, orte
ORDER BY name, vorname
```

- Wenn man mit Joins arbeitet, ist es üblich (best Practice) jedem Attribut den entsprechenden Tabellennamen voranzustellen um zu vermeiden, dass ein Fehler ausgegeben wird, falls in Zukunft einer Tabelle ein Attribut mit einem gleichen Namen hinzugefügt wird:

```
SELECT mitarbeiter.name,
       mitarbeiter.vorname,
       mitarbeiter.adresse,
       mitarbeiter.plz,
       ort.ort
FROM mitarbeiter, orte
ORDER BY mitarbeiter.name, mitarbeiter.vorname
```

Dies gibt einmal mehr viel Tipparbeit, die vermieden werden soll.

- Den Tabellen können Aliasse vergeben werden. Diese Aliasse können den Attributen an Stelle des vollständigen Tabellennamens vorangestellt werden:

```
SELECT m.name,
       m.vorname,
       m.adresse,
       m.plz,
       o.ort
FROM mitarbeiter m, orte o
ORDER BY m.name, m.vorname
```

## Kartesisches Produkt

Werden in der FROM Klausel mehrere Tabellen angegeben aber keine Angaben gemacht, wie sich die Tabellen verbinden sollen, wird jede mögliche Kombination der Daten ausgegeben (Anzahl Datensätze aus Tabelle 1 x Anzahl Datensätze aus Tabelle 2). Dies kann eine sehr grosse Datenmenge zur Folge haben.

```
SELECT m.name, m.vorname, l.land
FROM mitarbeiter m, laender l
ORDER BY m.name, m.vorname, l.land;
```

NAME	VORNAME	LAND
-----	-----	-----
Bieri	Erich	Mexico
Bieri	Erich	Nigeria
Bieri	Erich	Japan
Bieri	Erich	Italien
Bieri	Erich	Indien
Bieri	Erich	England
Blaser	Beat	Nigeria
Blaser	Beat	England
Blaser	Beat	Mexico
Blaser	Beat	Indien
Blaser	Beat	Italien
Blaser	Beat	Japan
...		
...		

In unserem Beispiel haben die Tabellen Mitarbeiter und Laender je 6 Rows. Ein kartesisches Produkt der beiden Tabellen hat also 36 Rows.

Hätten die Tabellen z.B. 250'000 Rows (Mitarbeiter) und 250 Rows (Länder), hätte die Ausgabe bereits 62'500'000 Rows.

## Klassischer Join Syntax über WHERE Klausel

Der klassische Weg zum Verbinden von Tabellen wird über die WHERE Klausel gemacht.

In der WHERE Klausel wird angegeben, über welches Attribut die Tabellen verknüpft werden sollen.

```
SELECT m.name, m.vorname, m.adresse, m.plz, o.ort
FROM mitarbeiter m, orte o
WHERE m.plz = o.plz
ORDER BY m.name, m.vorname
```

Der klassische Weg über die WHERE Klausel ist unflexibel und schliesst viele Möglichkeiten aus. Aus diesem Grund wurde eine weitere Möglichkeit zum Verbinden von Tabellen entwickelt, die „JOIN“ Klausel.

## Moderner Syntax, JOIN ... ON Klausel

Syntax generell:

```
SELECT [Attributsliste]
FROM [Linke Tabelle]
JOIN [Rechte Tabelle] ON [Linkes Attribut = Rechtes Attribut]
```

Generell ist der Syntax der JOIN Klausel ähnlich wie beim Verbinden von Tabellen über die WHERE Klausel. Es wird angegeben welche Tabelle über welches Attribut verbunden werden soll:

```
SELECT m.name, m.vorname, m.adresse, m.plz, o.ort
FROM mitarbeiter m
JOIN orte o ON m.plz = o.plz
ORDER BY m.name, m.vorname
```

## JOIN Variationen

Der JOIN Syntax bietet im Gegensatz zum klassischen WHERE Syntax einige Möglichkeiten mehr.

Es können in verschiedenen Kombinationen Daten ausgegeben werden, die dem Verbindungsattribut entsprechen oder Daten, die im Join Attribut NULL Werte enthalten.

## Testtabellen

Als Beispiel werden folgende Testdaten verwendet:

Tabelle Mitarbeiter:

NAME	VORNAME	FERIENLAND
Mueller	Peter	IN
Oehrli	Maria	IT
Bieri	Erich	
Blaser	Beat	UK
Hurni	Rudolf	IN
Lehner	Thomas	

Tabelle Laender:

L_ID	LAND
IN	Indien
IT	Italien
JP	Japan
MX	Mexico
NG	Nigeria
UK	England

## Klassischer Syntax

Beim klassischen Syntax ist es nicht möglich Daten der Join Attribute auszugeben, die NULL Values enthalten.

```
SELECT m.name, m.vorname, l.land
FROM mitarbeiter m, laender l
WHERE m.ferienland = l.l_id
ORDER BY m.name, m.vorname;
```

NAME	VORNAME	LAND
Blaser	Beat	England
Hurni	Rudolf	Indien
Mueller	Peter	Indien
Oehrli	Maria	Italien

4 rows selected.

Die zwei Mitarbeiter, die nicht in die Ferien gehen, werden nicht ausgegeben. Auf der anderen Seite werden die Länder nicht ausgegeben, in welche niemand in die Ferien geht.

Mit dem JOIN...ON Syntax kann dies gesteuert werden.

## JOIN...ON Möglichkeiten

Im JOIN Syntax wird von „INNER-“, „OUTER-“, „FULL-“, „LEFT-“ und „RIGHT JOIN“ gesprochen.

### INNER JOIN:

Wie beim klassischen Syntax mit der WHERE Klausel. Es werden von beiden Tabellen nur die Daten ausgegeben, die abgefüllt sind.

Das Keyword „INNER“ selber ist optional und muss nicht angegeben werden.

### OUTER JOIN:

Bei OUTER Join's werden von einer- oder von beiden Tabellen alle Daten ausgegeben, auch wenn das Join Attribut leer ist.

Welche Tabelle vollständig ausgegeben werden soll, wird mit „RIGHT“ (Tabelle in der JOIN Klausel), „LEFT“ (Tabelle in der FROM Klausel) oder mit „FULL“ (beide Tabellen) angegeben.

Die Bezeichnungen „LEFT“ und „RIGHT“ beziehen sich auf die Tabellen in Leserichtung. Wir lesen von links nach rechts, also ist die erste Tabelle, die wir lesen die „Linke“ Tabelle.

Wie bei INNER Join's auch, ist das Keyword „OUTER“ optional.

## LEFT OUTER JOIN

LEFT OUTER JOIN gibt alle Daten der linken Tabelle aus sowie diejenigen aus der rechten Tabelle, die dem Join Kriterium entsprechen.

```
SELECT m.name, m.vorname, l.land
FROM mitarbeiter m
LEFT JOIN laender l ON m.ferienland = l.l_id
ORDER BY m.name, m.vorname;
```

NAME	VORNAME	LAND
-----	-----	-----
Bieri	Erich	
Blaser	Beat	England
Hurni	Rudolf	Indien
Lehner	Thomas	
Mueller	Peter	Indien
Oehrli	Maria	Italien

6 rows selected.

Was passiert jetzt, wenn wir die beiden Tabellen tauschen?

```
SELECT m.name, m.vorname, l.land
FROM laender l
LEFT JOIN mitarbeiter m ON m.ferienland = l.l_id
ORDER BY m.name, m.vorname;
```

NAME	VORNAME	LAND
-----	-----	-----
Blaser	Beat	England
Hurni	Rudolf	Indien
Mueller	Peter	Indien
Oehrli	Maria	Italien
		Japan
		Nigeria
		Mexico

7 rows selected.

--> Es spielt also sehr wohl eine Rolle in welcher Reihenfolge das Statement formuliert wird.

Die OUTER JOIN Klausel muss also entsprechend der Reihenfolge gewählt werden.

## RIGHT OUTER JOIN

RIGHT OUTER JOIN gibt alle Daten der rechten Tabelle aus, sowie diejenigen aus der linken Tabelle, die dem Join Kriterium entsprechen.

```
SELECT m.name, m.vorname, l.land
FROM mitarbeiter m
RIGHT JOIN laender l ON m.ferienland = l.l_id
ORDER BY m.name, m.vorname;
```

NAME	VORNAME	LAND
Blaser	Beat	England
Hurni	Rudolf	Indien
Mueller	Peter	Indien
Oehrli	Maria	Italien
		Japan
		Nigeria
		Mexico

7 rows selected.

## FULL OUTER JOIN

Dieser JOIN liefert alle Datensätze beider Tabellen, ggf. unter Berücksichtigung der WHERE-Klausel zurück. Wenn Datensätze nach der Verknüpfungsbedingung zusammenpassen, werden sie in einer Zeile ausgegeben, sonst auf eigenen Zeilen mit entsprechenden NULL Values.

```
SELECT m.name, m.vorname, l.land
FROM mitarbeiter m
FULL JOIN laender l ON m.ferienland = l.l_id
ORDER BY m.name, m.vorname;
```

NAME	VORNAME	LAND
-----		
Bieri	Erich	
Blaser	Beat	England
Hurni	Rudolf	Indien
Lehner	Thomas	
Mueller	Peter	Indien
Oehrli	Maria	Italien
		Nigeria
		Japan
		Mexico

9 rows selected.

Was würde hier jetzt passieren, wenn man die beiden Tabellen vertauscht?

```
SELECT m.name, m.vorname, l.land
FROM laender l
FULL JOIN mitarbeiter m ON m.ferienland = l.l_id
ORDER BY m.name, m.vorname;
```

NAME	VORNAME	LAND
-----		
Bieri	Erich	
Blaser	Beat	England
Hurni	Rudolf	Indien
Lehner	Thomas	
Mueller	Peter	Indien
Oehrli	Maria	Italien
		Nigeria
		Japan
		Mexico








9 rows selected.

--> Es ändert sich nichts, da von beiden Tabellen jeweils alles ausgegeben wird.

Die Sortierreihenfolge würde anders sein, da hier jedoch sortiert ausgegeben wurde, ist das nicht ersichtlich.



## JOIN Arten

INNER JOIN		Wie die klassische Methode. Nur die Daten, die abgefüllt sind werden ausgegeben.	<pre>SELECT m.name, l.land FROM mitarbeiter m, laender l WHERE m.ferienland = l.l_id oder SELECT m.name, l.land FROM mitarbeiter m JOIN laender l ON m.ferienland = l.l_id oder SELECT m.name, l.land FROM mitarbeiter m INNER JOIN laender l ON m.ferienland = l.l_id</pre>
LEFT JOIN		<ul style="list-style-type: none"> <li>• Alle Mitarbeiter werden ausgegeben</li> <li>• Nur die Länder werden ausgegeben, welche auch verknüpft sind</li> </ul>	<pre>SELECT m.name, l.land FROM mitarbeiter m LEFT JOIN laender l ON m.ferienland = l.l_id oder SELECT m.name, l.land FROM mitarbeiter m LEFT OUTER JOIN laender l ON m.ferienland = l.l_id</pre>
RIGHT JOIN		<ul style="list-style-type: none"> <li>• Nur die Mitarbeiter werden ausgegeben, die auch verknüpft sind</li> <li>• Alle Länder werden ausgegeben</li> </ul>	<pre>SELECT m.name, l.land FROM mitarbeiter m RIGHT JOIN laender l ON m.ferienland = l.l_id</pre>
LEFT JOIN IS NULL		Nur die Mitarbeiter werden ausgegeben, die nicht in die Ferien gehen	<pre>SELECT m.name, l.land FROM mitarbeiter m LEFT JOIN laender l ON m.ferienland = l.l_id WHERE l.l_id IS NULL</pre>
RIGHT JOIN IS NULL		Nur die Länder werden ausgegeben, in welche niemand in die Ferien geht	<pre>SELECT m.name, l.land FROM mitarbeiter m RIGHT JOIN laender l ON m.ferienland = l.l_id WHERE m.ferienland IS NULL</pre>
FULL JOIN		<ul style="list-style-type: none"> <li>• Alle Mitarbeiter werden ausgegeben</li> <li>• Alle Länder werden ausgegeben</li> </ul>	<pre>SELECT m.name, l.land FROM mitarbeiter m FULL JOIN laender l ON m.ferienland = l.l_id</pre>
FULL JOIN IS NULL		<ul style="list-style-type: none"> <li>• Alle Mitarbeiter, welche nicht in die Ferien gehen, werden ausgegeben</li> <li>• Alle Länder, in die niemand in die Ferien geht, werden ausgegeben</li> </ul>	<pre>SELECT m.name, l.land FROM mitarbeiter m FULL JOIN laender l ON m.ferienland = l.l_id WHERE m.ferienland IS NULL OR l.l_id IS NULL</pre>