SQL - Joins II

Mit dem `JOIN`-Statement lassen sich Werte aus mehreren Tabellen direkt kombinieren, ohne zunächst durch die Abfrage mehrerer Tabellen zunächst das Tabellenprodukt zu bilden und dieses anschliessend zu filtern.

Es wird also ein einzelnes Statement an das DMBS gesendet mit dem mehrere Tabellen zugleich abgefragt und direkt verknüpft werden - das Prinzip bleibt jedoch gleich, auch beim Einsatz des JOIN Statements müssen Primär- und Fremdschlüsselspalten angegeben werden, damit eine sinnvolle Ergebnistabelle zurückgegeben wird.

In MySQL stehen mehrere JOIN-Typen zur Verfügung unter anderem LEFT JOIN, RIGHT JOIN, INNER JOIN.

LEFT JOIN

Die Syntax für einen LEFT JOIN ist wie folgt:

```
SELECT * FROM tabelle1
LEFT JOIN tabelle2 ON tabelle1.SpaltennameA = tabelle2.Spaltenname
LEFT JOIN tabelle3 ON tabelle1.SpaltennameB = tabelle3.Spaltenname
WHERE ...
```

LEFT JOIN bedeutet nun, dass stets alle Zeilen der Tabelle zurückgegeben, die beim FROM aufgeführt sind - als gewissermaßen „links“ stehen. Diese Tabelle stellt die Basis für das Ergebnis dar.

Es kann jetzt aber sein, dass in der Tabelle die per LEFT JOIN verknüpft wird kein passender Eintrag gefunden wird, es gibt also keinen Datensatz in den beiden Tabellen, bei denen `tabelle1.SpaltennameA = tabelle2.Spaltenname` ist. In diesem Fall erhalten diese Felder den NULL Wert, die Datenfelder der Ausgangstabelle werden aber auf jeden Fall ausgegeben.

Beispiel:

```
SELECT * FROM lehrer LEFT JOIN schueler ON lehrer.id=schueler.KLID
```

Es werden also die Lehrer zusammen mit den Schülern ausgegeben, die sie unterrichten. Weil als Selektor * angegeben war, werden alle Felder beider Tabellen, bei denen die Join-Bedingung erfüllt ist ausgegeben - das wird man meist nicht wollen, s.u. Emmi Nöther hat keine Schüler, da es sich jedoch um einen Left Join handelt, wird ihr Datensatz dennoch ausgegeben und die fehlenden Schüler:innen durch NULL-Felder ersetzt.

Natürlich kann man nun wie immer selektieren, welche Felder ausgegeben werden sollen, mit einer WHERE-Clause bestimmte Datensätze herausfiltern oder die Ergebnisse sortieren:

```
SELECT lehrer.name,lehrer.vorname, schueler.name, schueler.vorname
FROM lehrer
LEFT JOIN schueler ON lehrer.id=schueler.KLID
ORDER BY lehrer.name ASC
```

```
SELECT lehrer.name,lehrer.vorname, schueler.name, schueler.vorname
FROM lehrer
LEFT JOIN schueler ON lehrer.id=schueler.KLID
WHERE lehrer.name LIKE "%t%"
ORDER BY lehrer.name ASC
```

Hier kann man auch nochmal schön demonstrieren, wie man mit der Benennung von Feldern mittels AS die Lesbarkeit der Statements verbessern kann:

```
SELECT lehrer.name  AS Lname,lehrer.vorname AS LVname, schueler.name AS
SName, schueler.vorname AS SVname
FROM lehrer
LEFT JOIN schueler ON lehrer.id=schueler.KLID
ORDER BY lehrer.name ASC
```

Weitere Join-Statements

RIGHT JOIN

Die Syntax von RIGHT JOIN entspricht der von LEFT JOIN. Der Unterschied ist, dass hier die Tabelle die im JOIN hinzugefügt wird als Basis für die Datensätze dient - gibt es keine Treffer, werden die Felder der mit FROM selektierten Tabelle mit NULL-Werten gefüllt:

```
SELECT * FROM lehrer RIGHT JOIN schueler ON lehrer.id=schueler.KLID
```

„Kartoffelchips“ und „Schokocreme“ haben keinen Klassenlehrer, diese werden mit NULL aufgefüllt.

INNER JOIN

Bei einem INNER JOIN muss eine passende Zeile in den Tabellen gefunden werden, Datensätze, die die JOIN-Bedingung nicht erfüllen werden nicht zurückgegeben.

```
SELECT * FROM lehrer INNER JOIN schueler ON lehrer.id=schueler.KLID
```

Das entspricht unserer bisherigen Praxis, zunächst das kartesische Produkt aller beteiligten Tabellen abzufragen[1] und dann die passenden Datensätze mit WHERE herauszufiltern:

```
SELECT * FROM lehrer, schueler WHERE lehrer.id=schueler.KLID
```

1)

Das geht übrigens mit einem „CROSS JOIN“ auch, d.h. `SELECT * FROM lehrer,schueler` ist dasselbe wie `SELECT * FROM lehrer CROSS JOIN schueler`