SQL - Joins II

Mit dem `JOIN`-Statement lassen sich Werte aus mehreren Tabellen direkt kombinieren, ohne zunächst durch die Abfrage mehrerer Tabellen 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 werden, die beim FROM aufgeführt sind - also 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
```

id	name	vorname	klehrer_in	fach	id	name	vorname	klasse	telefon	KLID
1	Einstein	Albert	7a	Physik	1	Margarine	Carl	8a	7625653	1
1	Einstein	Albert	7a	Physik	2	Schockokeks	Marie	7a	87216625	1
3	Lovelace	Ada	6a	Informatik	3	Erdbeermarmelade	Alan	7b	454261	3
2	Newton	Isaac	6a	Mathe	4	Schinkenwurst	Hermann	6a	872662625	2
6	Freud	Sigmund	6a	Psychologie	5	Gummibärchen	Karl	7a	78265635	6
3	Lovelace	Ada	6a	Informatik	8	Apfelsaft	Lieschen	7b	991823764	3
2	Newton	Isaac	6a	Mathe	9	Aprikosenjoghurt	Klaus-Gustav	7b	9288374	2
5	Gauss	Carl-Friedrich	6a	Mathe	13	Cornfkakes	Crunchie	7b	92848724t36	5
4	Noether	Emmi	6b	Mathe	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*

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
```

name ▲ 1	vorname	name	vorname
Einstein	Albert	Margarine	Carl
Einstein	Albert	Schockokeks	Marie
Freud	Sigmund	Gummibärchen	Karl
Gauss	Carl-Friedrich	Cornfkakes	Crunchie
Lovelace	Ada	Apfelsaft	Lieschen
Lovelace	Ada	Erdbeermarmelade	Alan
Newton	Isaac	Aprikosenjoghurt	Klaus-Gustav
Newton	Isaac	Schinkenwurst	Hermann
Noether	Emmi	*NULL*	*NULL*

```
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
```

name ▲ 1	vorname	name	vorname
Einstein	Albert	Schockokeks	Marie
Einstein	Albert	Margarine	Carl
Newton	Isaac	Schinkenwurst	Hermann
Newton	Isaac	Aprikosenjoghurt	Klaus-Gustav
Noether	Emmi	*NULL*	*NULL*

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
```

Lname	LVname	SName	SVname
Einstein	Albert	Margarine	Carl
Einstein	Albert	Schockokeks	Marie
Freud	Sigmund	Gummibärchen	Karl
Gauss	Carl-Friedrich	Cornfkakes	Crunchie
Lovelace	Ada	Erdbeermarmelade	Alan
Lovelace	Ada	Apfelsaft	Lieschen
Newton	Isaac	Schinkenwurst	Hermann
Newton	Isaac	Aprikosenjoghurt	Klaus-Gustav
Noether	Emmi	*NULL*	*NULL*

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
```

id	name	vorname	klehrer_in	fach	id	name	vorname	klasse	telefon	KLID
2	Newton	Isaac	6a	Mathe	1	Margarine	Carl	8a	7625653	2
3	Lovelace	Ada	6a	Informatik	2	Schockokeks	Marie	7a	87216625	3
4	Noether	Emmi	6b	Mathe	3	Erdbeermarmelade	Alan	7b	454261	4
5	Gauss	Carl-Friedrich	6a	Mathe	4	Schinkenwurst	Hermann	6a	872662625	5
1	Einstein	Albert	7a	Physik	5	Gummibärchen	Karl	7a	78265635	1
2	Newton	Isaac	6a	Mathe	6	Eisbein	Hildegard	7a	7288393	2
NULL	*NULL*	*NULL*	*NULL*	*NULL*	7	Kartoffelchips	Gabriele	6a	928893484	0
3	Lovelace	Ada	6a	Informatik	8	Apfelsaft	Lieschen	7b	991823764	3
4	Noether	Emmi	6b	Mathe	9	Aprikosenjoghurt	Klaus-Gustav	7b	9288374	4
2	Newton	Isaac	6a	Mathe	10	Kaffepulver	Ingeborg	6a	209384	2
NULL	*NULL*	*NULL*	*NULL*	*NULL*	11	Schokocreme	Lisa-Marie	6a	923867346	0
1	Einstein	Albert	7a	Physik	12	Cornfkakes	Crunchie	7b	92848724t36	1

"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
```

id	name	vorname	klehrer_in	fach	id	name	vorname	klasse	telefon	KLID
2	Newton	Isaac	6a	Mathe	1	Margarine	Carl	8a	7625653	2
3	Lovelace	Ada	6a	Informatik	2	Schockokeks	Marie	7a	87216625	3
4	Noether	Emmi	6b	Mathe	3	Erdbeermarmelade	Alan	7b	454261	4
5	Gauss	Carl-Friedrich	6a	Mathe	4	Schinkenwurst	Hermann	6a	872662625	5
1	Einstein	Albert	7a	Physik	5	Gummibärchen	Karl	7a	78265635	1
2	Newton	Isaac	6a	Mathe	6	Eisbein	Hildegard	7a	7288393	2
3	Lovelace	Ada	6a	Informatik	8	Apfelsaft	Lieschen	7b	991823764	3
4	Noether	Emmi	6b	Mathe	9	Aprikosenjoghurt	Klaus-Gustav	7b	9288374	4
2	Newton	Isaac	6a	Mathe	10	Kaffepulver	Ingeborg	6a	209384	2
1	Einstein	Albert	7a	Physik	12	Cornfkakes	Crunchie	7b	92848724t36	1

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
```

Aufgaben

(A1)

Vollziehe die Beispiele oben mit der Datenbank

schule_klein_mit_ids.sql.zip

nach.

(A2)

- Lösche zunächst die Tabellen `schueler` und `lehrer` und importiere die Datenbank
  schule500_sus_keine_keys.zip
  - diese hat mehr Lehrer und Schüler als die Datenbank aus dem vorigen Beispiel.
- Versehe die Tabellen mit Primär und Fremdschlüsseln.
- Skizziere ein ER Diagramm.

Löse die folgenden Aufgaben mit Hilfe des kartesischen Produkts und mit einem INNER JOIN.

- Erstelle eine Klassenliste der 10a
- Erstelle eine Liste aller Schüler, die Feynman als Betreuer haben.
- Wieviele Schüler befinden sich in der Jahrgangsstufe 6?
- Wieviele Schüler haben einen Nachnamen der mit H beginnt.
- Erstelle eine Liste aller Schüler, die in der Unterstufe sind und Salvador Dali als Betreuer haben.

1)

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