

Abfragen über mehrere Tabellen

Für viele Abfragen müssen Daten aus mehreren Tabellen miteinander verknüpft werden.

Wir betrachten noch einmal die Beispieldatenbank einer Universität.

Eine typische Abfrage wäre z.B. „Wie heißt der Dozent, der den Kurs 'Java' leitet?“

Student

<u>MatrNr</u>	Name
1011	Schmitz
1012	Müller
1013	Jansen
1014	Meier

besucht

<u>MatrNr</u>	<u>KursNr</u>
1012	K01
1012	K03
1013	K02
1014	K03

Kurs

<u>KursNr</u>	Titel	PersNr
K01	Java	D02
K02	HTML	D02
K03	Datenbank	D01

Dozent

<u>PersNr</u>	Name
D01	Wirth
D02	Dijkstra
D03	Knuth

Als Mensch würden wir in der Kurs-Tabelle die Personalnummer in der Zeile mit dem Kurs 'Java' nachschauen (das wäre die D02), und anhand der Personalnummer dann den Namen des Dozenten in der Dozenten-Tabelle nachschlagen (Dijkstra).

Man benötigt dazu zwei Schritte. Mit SQL erledigt man solche Abfragen jedoch in einem Schritt mit einem sog. **Verbund (Join)**. Ein Verbund schafft aus den beiden betroffenen Tabellen „Kurs“ und „Dozent“ eine neue Tabelle, die das **Kreuzprodukt** beider Tabellen enthält.

Das Kreuzprodukt kombiniert jeden Datensatz der Kurs-Tabelle mit jedem Datensatz der Dozent-Tabelle – das ergibt im Beispiel $3 \times 3 = 9$ Datensätze:

Kurs

<u>KursNr</u>	Titel	PersNr
K01	Java	D02
K02	HTML	D02
K03	Datenbank	D01

Dozent

<u>PersNr</u>	Name
D01	Wirth
D02	Dijkstra
D03	Knuth

Verbund von Kurs und Dozent (Kreuzprodukt)

KursNr	Titel	PersNr	PersNr	Name
K01	Java	D02	D01	Wirth
K01	Java	D02	D02	Dijkstra
K01	Java	D02	D03	Knuth
K02	HTML	D02	D01	Wirth
K02	HTML	D02	D02	Dijkstra
K02	HTML	D02	D03	Knuth
K03	Datenbank	D01	D01	Wirth
K03	Datenbank	D01	D02	Dijkstra
K03	Datenbank	D01	D03	Knuth

Von diesen 9 Datensätzen sind nicht alle sinnvoll, sondern nur die, bei denen die Personalnummer für die Kurse mit der Personalnummer für die Dozenten übereinstimmt:

Verbund von Kurs und Dozent (Kreuzprodukt)

KursNr	Titel	PersNr (Kurs)	PersNr (Dozent)	Name
K01	Java	D02	D01	Wirth
K01	Java	D02	D02	Dijkstra
K01	Java	D02	D03	Knuth
K02	HTML	D02	D01	Wirth
K02	HTML	D02	D02	Dijkstra
K02	HTML	D02	D03	Knuth
K03	Datenbank	D01	D01	Wirth
K03	Datenbank	D01	D02	Dijkstra
K03	Datenbank	D01	D03	Knuth

Man schränkt das Kreuzprodukt mit SQL so ein, dass nur diese sinnvollen Datensätze in der Tabelle verbleiben. Dann hat man in jeder Zeile die Daten der Kurse verbunden mit den Daten der Dozenten, und kann so leicht herausfinden, wie der Dozent heißt, der den Kurs 'Java' leitet.

Verbund von Kurs und Dozent (nur sinnvolle Zeilen)

KursNr	Titel	PersNr (Kurs)	PersNr (Dozent)	Name
K01	Java	D02	D02	Dijkstra
K02	HTML	D02	D02	Dijkstra
K03	Datenbank	D01	D01	Wirth

Die SQL-Abfrage für einen solchen Verbund verwendet das Schlüsselwort „INNER JOIN“:

```
SELECT Dozent.Name FROM Kurs INNER JOIN Dozent
ON Kurs.PersNr = Dozent.PersNr
WHERE Kurs.Name = 'Java' ;
```

Zur Erläuterung:

- Kurs **INNER JOIN** Dozent bildet das Kreuzprodukt der Tabellen Kurs und Dozent. „inner join“ bedeutet, dass man nur die Zeilen auswählt, bei denen die Schlüsselattribute der beiden Tabellen übereinstimmen – die anderen, sinnlosen Datensätze werden aussortiert.
- **ON** Kurs.PersNr = Dozent.PersNr ist die Bedingung, mit der die „sinnvollen“ Datensätze definiert werden – die Datenbank erkennt nämlich nicht automatisch, welches die Schlüsselattribute sind, auch wenn diese (wie hier im Beispiel) den gleichen Namen haben. D.h. wir müssen bei INNER JOINS immer angeben, welche Spalten übereinstimmen sollen.
- Sobald eine SQL-Abfrage mehrere Tabellen beinhaltet, muss man für jede Spalte auch angeben, welche Tabelle gemeint ist, mithilfe des **Punktoperators**: `Tabelle.Spalte` oder im Beispiel: `Dozent.Name`, `Kurs.PersNr`, usw.

Alternative Schreibweisen für JOINS

Für den Verbund von Tabellen gibt es in SQL zwei alternative Schreibweisen. Bei der einen wird explizit das Schlüsselwort JOIN verwendet, bei der anderen lässt man es weg.

Schreibweise mit Schlüsselwort JOIN:

```
SELECT Dozent.Name FROM Kurs INNER JOIN Dozent
ON Kurs.PersNr = Dozent.PersNr
WHERE Kurs.Name = 'Java' ;
```

Alternative ohne Schlüsselwort JOIN (gleiche Bedeutung):

```
SELECT Dozent.Name FROM Kurs, Dozent
WHERE Kurs.PersNr = Dozent.PersNr
AND Kurs.Name = 'Java' ;
```

JOINS über drei oder mehr Tabellen

Für die Abfrage „Welche Kurse besucht der Student mit Namen 'Jansen'“ braucht es schon drei Tabellen: Aus Tabelle „Student“ erhalten wir die Matrikelnummer, damit aus der Tabelle „besucht“ die Kursnummer, und darüber aus der Tabelle „Kurs“ den Namen des Kurses:

<i>Student</i>		<i>besucht</i>		<i>Kurs</i>		
<u>MatrNr</u>	Name	<u>MatrNr</u>	<u>KursNr</u>	<u>KursNr</u>	Titel	PersNr
1013	Jansen	1013	K02	K02	HTML	D02

Bei diesem JOIN müssen wir zwei Schlüsselattribute definieren:

MatrNr für die Tabellen Student und besucht, sowie **KursNr** für die Tabellen besucht und Kurs.

Bei der Schreibweise *mit* JOIN hängt man mehrere JOINS mit ON hintereinander.

Dabei werden die JOINS nacheinander in Klammern gesetzt.

```
SELECT Kurs.Titel FROM
(Student INNER JOIN besucht ON Student.MatrNr = besucht.MatrNr)
INNER JOIN Kurs ON besucht.KursNr = Kurs.KursNr
WHERE Student.Name = 'Jansen' ;
```

Bei der Schreibweise *ohne* JOIN hängt man die Bedingungen mit AND hintereinander:

```
SELECT Kurs.Titel FROM
Student, besucht, Kurs
WHERE Student.MatrNr = besucht.MatrNr (1. Schlüsselattribut)
AND besucht.KursNr = Kurs.KursNr (2. Schlüsselattribut)
AND Student.Name = 'Jansen' ; (Suchkriterium)
```