



Join the tables

Bei normalisierten relationalen Datenbanken befinden sich gesuchte Informationen häufig in mehreren Tabellen. Durch einen sogenannten "Join" ("Verbund") lassen sich die gewünschten Informationen aus den erforderlichen Tabellen zu einer Ergebnistabelle zusammen führen (Sie erinnern sich: Alle Operationen auf Relationen liefern als Ergebnis ebenfalls eine Relation!). Beginnen wir mit einem sehr einfachen -aber zumeist auch völlig nutzlosen- Join.

Cross Join oder das kartesische Produkt

Die einfachste Join-Operation ist der Cross Join. Nehmen wir als Beispiel folgende zwei Relationen. Eine Relation **PERSONEN** und eine Relation **ORT** mit folgendem Aufbau:

PERSONEN(ID, VORNAME, NAME, STRASSE, ORT_PLZ)

ORT(PLZ, STADT)

Nehmen Sie mit mir an, es seien folgende Tupel in den Relationen enthalten:

PERSONEN

| ID | VORNAME | NAME | STRASSE | ORT_PLZ |
|----|------------|----------|-----------------|---------|
| 1 | Isaac | Newton | Rotdornweg 12 | 6000 |
| 2 | Albert | Einstein | Blumenstraße 14 | 5000 |
| 3 | Richard P. | Feynmann | Nelkenallee 23 | 6000 |

ORT

| PLZ | STADT |
|------|-----------|
| 5000 | Köln |
| 6000 | Frankfurt |

Wenn wir nun einen Cross Join (**PERSONEN** x **ORT**) ausführen, erhalten wir eine Ergebnisrelation, bei der jedes Tupel aus der ersten Relation (**PERSONEN**) mit jedem Tupel aus der zweiten Relation (**ORT**) ein neues Tupel bildet. Unschwer lässt sich die Kardinalität der Ergebnisrelation bestimmen:

$$|\text{PERSONEN} \times \text{ORT}| = |\text{PERSONEN}| \cdot |\text{ORT}| = 3 \cdot 2 = 6.$$

Der Grad der Relation beträgt nun 7. Wie sieht nun die Ergebnisrelation aus?

PERSONEN x ORT

| ID | VORNAME | NAME | STRASSE | ORT_PLZ | PLZ | STADT |
|----|------------|----------|-----------------|---------|------|-----------|
| 1 | Isaac | Newton | Rotdornweg 12 | 6000 | 5000 | Köln |
| 1 | Isaac | Newton | Rotdornweg 12 | 6000 | 6000 | Frankfurt |
| 2 | Albert | Einstein | Blumenstraße 14 | 5000 | 5000 | Köln |
| 2 | Albert | Einstein | Blumenstraße 14 | 5000 | 6000 | Frankfurt |
| 3 | Richard P. | Feynmann | Nelkenallee 23 | 6000 | 5000 | Köln |
| 3 | Richard P. | Feynmann | Nelkenallee 23 | 6000 | 6000 | Frankfurt |

Wie man sich leicht sehen kann, liefert diese Ergebnisrelation nicht wirklich sinnvolle oder informative Tupel. Und: Ein Cross Join liefert u.U. riesige Ergebnistabellen, die dann eine beachtliche Speichermenge erfordern. Der zugehörige SQL-Befehl lautet:

SELECT * FROM PERSONEN,ORT



| | | |
|---|------------------------------------|---------------------|
| B | Lehrgang: Datenbanken | Arbeitsblatt Nr. 18 |
| S | Thema: Verbinden mehrerer Tabellen | Datum: |
| G | Name: | Seite 2 von 6 |

Gleiches mit gleichem...

Wenden wir uns einem etwas sinnvollerem Verbund zu, dem **Equi-Join**. Der Equi-Join (oder auch **Inner Join** genannt) liefert eine Ergebnisrelation, die sich aus der **Identität zweier Attributwerte** in den zu verbindenden Relationen ergibt.

Sehen wir uns das an einem Beispiel an, das ich zuerst als Frage formulieren möchte:

Wähle mir alle Attributwerte aus den Tabellen Personen und Ort bei denen in der Tabelle Personen das Attribut `ORT_PLZ` identisch mit dem Attribut `PLZ` in der Tabelle Ort ist.

Wenn wir das nun in SQL übersetzen, lautet die Abfrage:

```
SELECT * FROM PERSONEN,ORT
WHERE PERSONEN.ORT_PLZ = ORT.PLZ
```

Hier sehen sie auch, dass man vor die Attributbezeichnung den Namen der Tabelle setzen kann, aus der das Attribut stammt. Dies ist in diesem Beispiel nicht unbedingt erforderlich, fördert aber die Lesbarkeit. Zwingend ist dies aber dann, wenn in verschiedenen Relationen, auf die Bezug genommen wird, gleichnamige Attribute existieren!

Was liefert uns nun diese Abfrage?

PERSONEN EQUI JOIN ORT ON PERSONEN.ORT_PLZ = ORT.PLZ

| ID | VORNAME | NAME | STRASSE | ORT_PLZ | PLZ | STADT |
|----|------------|----------|-----------------|---------|------|-----------|
| 1 | Isaac | Newton | Rotdornweg 12 | 6000 | 6000 | Frankfurt |
| 2 | Albert | Einstein | Blumenstraße 14 | 5000 | 5000 | Köln |
| 3 | Richard P. | Feynmann | Nelkenallee 23 | 6000 | 6000 | Frankfurt |

Nun sehen Sie die zu den jeweiligen Postleitzahlen gehörigen Städtenamen.

Alternativ lässt sich der Equi Join mit dem Befehl **Inner Join** folgendermaßen formulieren:

```
SELECT * FROM PERSONEN INNER JOIN ORT
ON PERSONEN.ORT_PLZ = ORT.PLZ
```

Das Ergebnis der Abfrage ist das Gleiche.



| | | |
|------------------|------------------------------------|---------------------|
| B S G G | Lehrgang: Datenbanken | Arbeitsblatt Nr. 18 |
| | Thema: Verbinden mehrerer Tabellen | Datum: |
| | Name: | Seite 3 von 6 |

Lassen Sie uns die Möglichkeiten weiter ausloten. Stellen Sie sich vor, sie wollen nun nur noch die Personen sehen, die in Frankfurt wohnen. Der zugehörige Equi Join muss noch um eine weitere Bedingung ergänzt werden.

```
SELECT * FROM PERSONEN,ORT
WHERE PERSONEN.ORT_PLZ = ORT.PLZ
AND ORT.STADT = 'Frankfurt'
```

Die Abfrage liefert und nun

PERSONEN EQUI JOIN ORT ON PERSONEN.ORT_PLZ = ORT.PLZ AND ORT.STADT = 'Frankfurt'

| ID | VORNAME | NAME | STRASSE | ORT_PLZ | PLZ | STADT |
|----|------------|----------|----------------|---------|------|-----------|
| 1 | Isaac | Newton | Rotdornweg 12 | 6000 | 6000 | Frankfurt |
| 3 | Richard P. | Feynmann | Nelkenallee 23 | 6000 | 6000 | Frankfurt |

Als **Inner Join** formuliert, liest sich das wie folgt:

```
SELECT * FROM PERSONEN INNER JOIN ORT
ON PERSONEN.ORT_PLZ = ORT.PLZ
AND ORT.STADT = 'Frankfurt'
```

Zwei links, zwei rechts...

Eine weitere Variante eines Joins ist der sogenannte **Left Outer Join** oder kurz **Left Join**. Auf der Grundlage eines in beiden Tabellen identischen Attributwertes wird die Ergebnisrelation gebildet, wobei auch die Tupel der "linken" Tabelle (die zuerst genannte Tabelle) verwendet werden, für die keine Tupel in der "rechten" Tabelle (die zweite genannte Tabelle) existieren. Man nennt dies daher auch "linke Inklusionsverknüpfung".

Auch hier ein Beispiel. Nehmen wir an es gibt folgende Tabellen:

LEHRER(ID, Name)

FUNKTION(ID,Name,Funktion)

LEHRER

| <u>ID</u> | Name |
|-----------|------------|
| 1 | Homm |
| 2 | Ernst |
| 3 | Konrad |
| 4 | Eißing |
| 5 | Vogel |
| 6 | Roggendorf |
| 7 | Schneider |

FUNKTION

| <u>ID</u> | Name | Funktion |
|-----------|-----------|---------------------------------|
| 1 | Schneider | Schulleiter |
| 2 | Eißing | stellv. Schulleiter |
| 3 | Laun | Abteilungsleiter Metalltechnik |
| 4 | Konrad | Abteilungsleiter Elektrotechnik |
| 5 | Schmidt | Abteilungsleiter Bautechnik |



Formulieren wir einen Left Join bezogen auf das gemeinsame Feld Name:

```
SELECT * FROM LEHRER LEFT JOIN FUNKTION
ON LEHRER.NAME = FUNKTION.NAME
```

Als Ergebnis erhalten wir folgende Tabelle

LEHRER Left Join Funktion ON Lehrer.Name = Funktion.Name

| LEHRER.ID | LEHRER.Name | Funktion.ID | Funktion.Name | Funktion |
|-----------|-------------|-------------|---------------|---------------------------------|
| 1 | Homm | | | |
| 2 | Ernst | | | |
| 3 | Konrad | 4 | Konrad | Abteilungsleiter Elektrotechnik |
| 4 | Eißing | 2 | Eißing | stellv. Schulleiter |
| 5 | Vogel | | | |
| 6 | Roggendorf | | | |
| 7 | Schneider | 1 | Schneider | Schulleiter |

Es werden also auch diejenigen Tupel in der "linken" Tabelle, für die in der rechten Tabelle keine Einträge existieren (im gemeinsam genannten Attribut), gelistet.

Die alternative Variante ist der **Right Outer Join** oder kurz **Right Join**. Hierbei werden alle Tupel der "rechten" Tabelle aufgeführt; unabhängig, ob Tupel in der linken Tabelle mit identischem Wert im Bezugsattribut existieren.

LEHRER Right Join Funktion ON Lehrer.Name = Funktion.Name

| LEHRER.ID | LEHRER.Na- me | Funktion.ID | Funktion.Name | Funktion |
|-----------|------------------|-------------|---------------|---------------------------------|
| 7 | Schneider | 1 | Schneider | Schulleiter |
| 4 | Eißing | 2 | Eißing | stellv. Schulleiter |
| | | 3 | Laun | Abteilungsleiter Metalltechnik |
| 3 | Konrad | 4 | Konrad | Abteilungsleiter Elektrotechnik |
| | | 5 | Schmidt | Abteilungsleiter Bautechnik |

In der Welt der relationalen Datenbanken existieren noch weitere Joins; u.a. der sogenannte Theta Join (Non-Equi-Join) oder der Self Join.

Diese haben jedoch für unsere Zwecke keine Bedeutung und sollen daher auch nicht näher betrachtet werden.



| | | |
|---|------------------------------------|---------------------|
| B | Lehrgang: Datenbanken | Arbeitsblatt Nr. 18 |
| S | Thema: Verbinden mehrerer Tabellen | Datum: |
| G | Name: | Seite 5 von 6 |

Alias-Bezeichnungen

Da die SELECT-Abfragen aufgrund der Tabellennamen teilweise sehr "länglich" werden können, ist es möglich, für die Tabellenbezeichnungen Aliase zu verwenden, die dann zur Angabe der Tabelle verwendet werden können. Hierzu das vorige Beispiel unter Verwendung von Aliase:

```
SELECT * FROM LEHRER L LEFT JOIN FUNKTION F
ON L.NAME = F.NAME
```

Alias L
für die Tabelle
LEHRER

Alias F
für die Tabelle
FUNKTION

Diese Aliase können dann im jeweiligen SQL-Befehl an jeglicher Stelle anstatt des Tabellennamens verwendet werden; z.B.:

```
SELECT P.NAME FROM PERSONEN P, ORT O
WHERE P.ORT_PLZ = O.PLZ
AND O.STADT = 'Frankfurt'
```

Für den Alias-Namen verwendet man üblicherweise ein Kürzel, das aussagekräftig ist und zu weniger Schreibaufwand führt.

Geschachtelte Joins

Joins lassen sich schachteln. Sehen Sie sich mit mir das folgende fiktive Beispiel an:

```
SELECT Fa, Fb FROM Tabelle1 T1 LEFT JOIN
(Tabelle2 T2 LEFT JOIN Tabelle3 T3 ON T2.F2=T3.F3)
ON T1.F1 = F3
```

In den Klammern wird eine linke Inklusionsverknüpfung für Tabelle 2 (Alias T2) mit der Tabelle 3 (Alias T3) durchgeführt, wobei die Felder für den Attributwertvergleich F2 in Tabelle 2 und F3 in Tabelle 3 lauten. Mit dieser Ergebnisrelation wird eine weitere Inklusionsverknüpfung durchgeführt. Die "linke" Tabelle ist hierbei Tabelle 1, die rechte die Ergebnistabelle aus dem Klammerausdruck. Der Attributwertvergleich erfolgt anhand des Feldes F1 aus der Tabelle 1 und dem Feld F3 aus der Ergebnisrelation des Klammerausdruckes. Gelistet werden in der endgültigen Ergebnisrelation lediglich die Felder Fa und Fb.

B
S
G
G

Lehrgang: Datenbanken

Thema: Verbinden mehrerer Tabellen

Name:

Arbeitsblatt Nr. 18

Datum:

Seite 6 von 6

Übungen zu Joins

Für die nachfolgenden Übungen verwenden sie die zur Verfügung gestellten Datenbanken JOIN.MDB und BUCH-JOIN.MDB.

1. Führen Sie alle Beispiele dieses Arbeitsblattes aus und kontrollieren Sie, ob ihre Ergebnisse den dargestellten entsprechen. Nutzen Sie hierzu die Datenbank JOIN.MDB
2. Nutzen Sie für die folgenden Übungen die Datenbank BUCH-JOIN.MDB
Erstellen Sie ein kartesisches Produkt für die Tabellen BUCHAUTOR und BUCHTITEL.
Wie viele Datensätze mit wie vielen Attributen ergeben sich?

3. Ermitteln Sie für den Autor Eco alle Titel. Verwenden Sie Aliase! Es soll der Name und der Titel in der Ergebnisrelation erscheinen.

4. Erstellen sie eine Liste aller Verlage mit den jeweilig erschienenen Büchern (Verlagsname und Titel)

5. Erstellen sie eine Liste aller Bücher mit dem zugehörigen Verlag und Autor; alphabetisch nach Autor sortiert unter Verwendung von Aliasen.

6. Listen Sie **alle** Autoren mit Namen und den veröffentlichten Büchern mit den Titeln. Listen Sie auch diejenigen Autoren, zu denen kein Buchtitel enthalten ist.

7. Ergänzen Sie die Tabelle BUCHVERLAG um einen weiteren Verlag (mittels INSERT INTO)

8. Listen Sie **alle** Verlage sowie die jeweils im Verlag erschienenen Bücher (auch Verlage ohne veröffentlichte Bücher).

9. Listen Sie die Namen aller Autoren sowie die Titel der erschienenen Bücher und der jeweilige Verlagsname. Verwenden Sie Aliase!
