JOIN Abfragen in Datenbanken

Prof. Dr. Christian Bockermann

23. November 2022

In diesem Artikel geht es um ein paar Hintergrundinformationen zur Selektion von Daten aus mehreren Tabellen.

1 Einführung

Wir betrachten in diesem Artikel das Beispiel der Firma *PlanetExpress* und der zugehörigen Datenbank. Dabei sind die folgenden beiden Tabellen wichtig:

Tabelle: *Mitarbeiter*

nr	name	raum
1	Hubert J. Farnsworth	1
2	Turanga Leela	2
3	Philip J. Fry	2
4	Hermes Conrad	3
5	Amy Wong	1
6	Bender	NULL

Tabelle: *Raum*

nr	etage
1	Ganz oben
2	Hangar
3	Mittlere Etage
4	Keller

Tabelle: *Aufgabe*

nr	name
1	Wartung
2	Logistik
3	Einkauf

Die Tabellen beschreiben die Entitäten *Mitarbeiter* und *Raum*, wobei einem Mitarbeiter ein Raum zugeordnet ist. Einem Raum können jedoch mehrere Mitarbeiter zugeordnet werden. Wie aus der Tabelle *Mitarbeiter* ersichtlich ist, hat nicht jeder Mitarbeiter notwendigerweise einen zugeordneten Raum.

Die Entität *Aufgabe* wird ebenfalls in einer eigenen Tabelle abgebildet. Hier handelt es sich um eine N:M-Beziehung, da ein Mitarbeiter mehrere Aufgaben bearbeiten kann und eine Aufgabe auch von mehreren Mitarbeiter ausgeführt werden kann. Dafür gibt es dann noch die Verbindungstabelle *MitarbeiterAufgabe*:

Tabelle: *MitarbeiterAufgabe*

mitarbeiter	aufgabe
4	3
5	3
6	1
6	2

Die Abbildung 1 zeigt das ER-Diagramm für die Firma *Planet Express*. Darin ist die 1:N und die N:M Beziehung zu sehen.

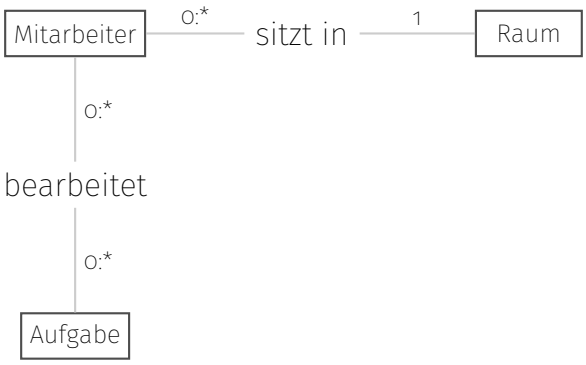

Abbildung 1: ER-Diagramm zu der Firma *Planet Express*.

2 Selektion aus mehreren Tabellen (JOIN)

Mit Hilfe des `SELECT` Befehls lassen sich auch Daten aus mehreren Tabellen auswählen. Mathematisch wird dabei das Kreuzprodukt der entsprechenden Tabellen gebildet, d.h. für zwei Mengen A und B liefert

```
SELECT * FROM A, B
```

alle Tupel, die durch Kombination der Elemente aus A und B gebildet werden.

Für das Beispiel der Tabellen *Mitarbeiter* und *Raum* mit dem Befehl

```
SELECT * FROM mitarbeiter, raum
```

ergeben sich damit alle Paarungen von Mitarbeiter und Raum wie in der folgenden Abbildung gezeigt.

nr	name	raum	nr	etage
1	Hubert J. Farnsworth	1	1	Ganz oben
1	Hubert J. Farnsworth	1	3	Mittlere Etage
1	Hubert J. Farnsworth	1	2	Hangar
1	Hubert J. Farnsworth	1	4	Keller
2	Turanga Leela	2	1	Ganz oben
2	Turanga Leela	2	3	Mittlere Etage
2	Turanga Leela	2	2	Hangar
2	Turanga Leela	2	4	Keller
3	Philip J. Fry	2	1	Ganz oben
3	Philip J. Fry	2	3	Mittlere Etage
3	Philip J. Fry	2	2	Hangar
3	Philip J. Fry	2	4	Keller
5	Amy Wong	1	1	Ganz oben
5	Amy Wong	1	3	Mittlere Etage
5	Amy Wong	1	2	Hangar
5	Amy Wong	1	4	Keller
4	Hermes Conrad	3	1	Ganz oben
4	Hermes Conrad	3	3	Mittlere Etage
4	Hermes Conrad	3	2	Hangar
4	Hermes Conrad	3	4	Keller
6	Bender	NULL	1	Ganz oben
6	Bender	NULL	3	Mittlere Etage
6	Bender	NULL	2	Hangar
6	Bender	NULL	4	Keller

Abbildung 2: Vollständer Join (Kreuzprodukt) der Tabellen *Mitarbeiter* und *Raum*.

Dabei werden auch Paarungen erzeugt, die mit den Fremdschlüsselbeziehungen garnicht übereinstimmen. Zum Beispiel enthält die Tabelle einen Eintrag für den Mitar-

beiter *Bender* mit dem Raum *Hangar*, obwohl der Mitarbeiter überhaupt keinem Raum zugeordnet ist (`raum` in der Tabelle *Mitarbeiter* für *Bender* ist `NULL`).

Um nur die Paarungen mit den wirklich zugeordneten Räumen zu bekommen, benötigen wir noch die Bedingung, dass der Fremdschlüssel `raum` in der Tabelle *Mitarbeiter* mit dem entsprechenden Primärschlüssel `nr` in der Tabelle *Raum* übereinstimmt:

```
SELECT * FROM mitarbeiter, raum
   WHERE mitarbeiter.raum = raum.nr
```

Die Abfrage mit der `where`-Bedingung wählt aus allen Paaren diejenigen aus, die laut Fremdschlüsselbeziehung gültig sind. Das führt dann zu der Tabelle in Abbildung 3:

nr	name	raum	nr	etage
1	Hubert J. Farnsworth	1	1	Ganz oben
2	Turanga Leela	2	2	Hangar
3	Philip J. Fry	2	2	Hangar
5	Amy Wong	1	1	Ganz oben
4	Hermes Conrad	3	3	Mittlere Etage

Abbildung 3: Verbindung der Tabellen *Mitarbeiter* und *Raum*.

Alternative Syntax mit `JOIN ... ON`

Es gibt dazu eine alternative SQL-Syntax mit dem Schlüsselwort `JOIN`, die zum gleichen Ergebnis führt:

```
SELECT * FROM mitarbeiter
    JOIN raum ON mitarbeiter.raum = raum.nr
```

Dabei wird nach dem Wort `JOIN` die Tabelle, die mit der ersten Tabelle verbunden werden soll angegeben. Dahinter folgt das Wort `ON` und danach die Bedingung für die Verbindung der beiden Tabellen.

Diese JOIN-Syntax hat den Vorteil, dass Bedingungen für die Verbindung der Tabellen ganz klar von anderen Bedingungen getrennt werden. Wählt man z.B. nur den Mitarbeiter und Raum für einen bestimmten Namen aus, kommt man zu folgendem SQL Befehl:

```
SELECT * FROM mitarbeiter
   JOIN raum ON mitarbeiter.raum = raum.nr
   WHERE mitarbeiter.name = 'Amy Wong'
```

2.1 Verbindung von Tabellen mit LEFT JOIN

Bei der Selektion aus den beiden Tabellen Mitarbeiter und Raum fällt jedoch auf, dass nur Mitarbeiter mit einem zugehörigen Raum auch im Ergebnis enthalten sind. Der Mitarbeiter *Bender*, dem kein Raum zugeordnet ist, taucht in der Ergebnistabelle in Abbildung 3 nicht auf.
Das liegt daran, dass der Fremdschlüssel für Mitarbeiter *Bender* den Wert `NULL` enthält und es dafür keinen Primärschlüssel in der Tabelle *Raum* gibt.
(Zur Erinnerung: der Primärschlüssel einer Tabelle darf nie `NULL` sein.)

Frage: Wie selektieren wir dann jetzt alle Mitarbeiter und die Räume?
Beim JOIN von zwei Tabellen gibt es einen linken und einen rechten Teil (1. Tabelle und zweite Tabelle). Im Fall von unserer Abfrage

```
SELECT * FROM mitarbeiter
   JOIN raum ON mitarbeiter.raum = raum.nr
```

wäre die Tabelle *Mitarbeiter* der linke Teil, *Raum* der Rechte.

Mit Hilfe einer sogenannten `LEFT JOIN` Abfrage, können wir der Datenbank sagen, dass wir alle Mitarbeiter und optional noch zugeordnete Räume abfragen wollen:

```
SELECT * FROM mitarbeiter
   LEFT JOIN raum ON mitarbeiter.raum = raum.nr
```

Das Ergebnis dieser Abfrage ist dann die folgende Tabelle aus Abbildung 4. Dabei sieht man insbesondere, dass nun alle Mitarbeiter enthalten sind und die nicht auszufüllenden Raum-Spalten (`raum_nr`, `etage`) im Ergebnis den Wert `NULL` bekommen.

nr	name	raum	nr	etage
5	Amy Wong	1.0	1	Ganz oben
1	Hubert J. Farnsworth	1.0	1	Ganz oben
4	Hermes Conrad	3.0	3	Mittlere Etage
3	Philip J. Fry	2.0	2	Hangar
2	Turanga Leela	2.0	2	Hangar
6	Bender	NULL	NULL	NULL

Abbildung 4: LEFT JOIN der Tabellen *Mitarbeiter* und *Raum*.

2.2 Überblick über weitere JOIN Abfragen

Neben dem normalen JOIN (auch *inner join* genannt) gibt es weitere Möglichkeiten, Tabellen miteinander zu verbinden. Die nachfolgende Abbildung 5 zeigt die verschiedenen JOIN-Arten und deren Bedeutung für das entstehende Resultat.

Die Abbildung verbindet zwei Tabellen, die Spalte mit den Zahlen sind jeweils die Fremd- bzw. Primärschlüssel.

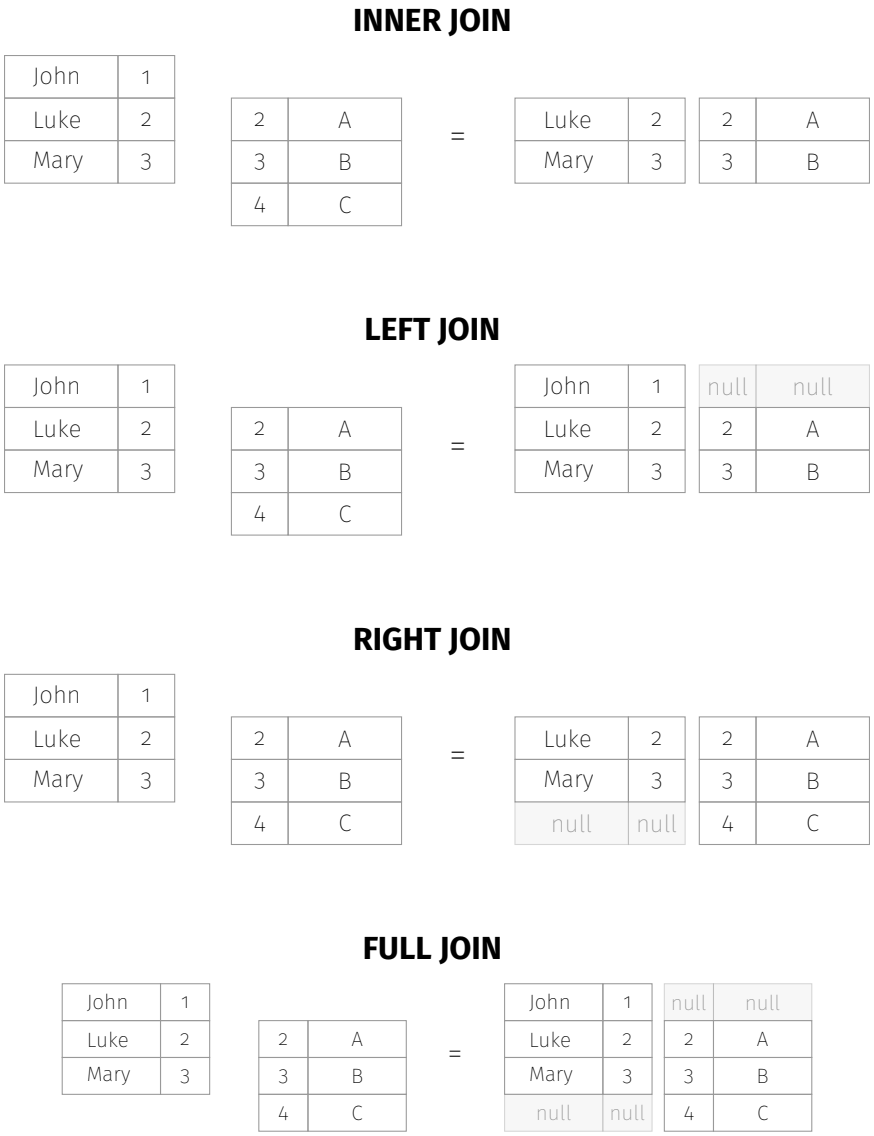

Abbildung 5: Übersicht über die gebräuchlichsten JOIN Abfragen.

Recht häufig finden LEFT JOIN Abfragen Verwendung – inbesondere, wenn die Daten in der 3ten oder 4ten Normalform vorliegen. Dann besteht eine Vielzahl von Tabellen aus Fremdschlüsseln und einige Tabellen enthalten die zugehörigen Bezeichnungen bzw. Attribute.

2.3 JOIN von Tabellen einer N:M Beziehung

Für N:M Beziehungen im ER-Diagramm wird für jede der beteiligten Entitäten eine Tabelle angelegt und eine zusätzliche Verbindungstabelle für die Darstellung der Beziehungen. Wir betrachten dazu die folgende Beziehung:

Ein Mitarbeiter kann also mehrere Aufgaben bearbeiten und eine Aufgabe kann von mehreren Mitarbeitern bearbeitet werden. Die zugehörigen Tabellen könnten aussehen wie in Abbildung 6 dargestellt:

Mitarbeiter

nr	name
4	Hermes Conrad
5	Amy Wong
6	Bender

MitarbeiterAufgabe

mitarbeiter	aufgabe
4	3
5	3
6	1
6	2

Aufgabe

nr	name
1	Wartung
2	Logistik
3	Einkauf

Abbildung 6: Tabellen für die N:M Beziehung von Mitarbeiter und Aufgabe.

Dabei ist zu beachten, dass die zentrale Verbindungstabelle *MitarbeiterAufgabe* zu den beiden anderen Tabellen jeweils eine Verbindung mit möglichen Mehrfachverweisen hat. Als Beispiel taucht in der Spalte `mitarbeiter` zweimal der Wert 6 auf.
Daher lässt sich die Tabelle *MitarbeiterAufgabe* schlecht mit einem LEFT JOIN an die *Mitarbeiter* Tabelle anfügen. Stattdessen nutzen wir in diesem Fall die *MitarbeiterAufgabe* Tabelle als Basis-Tabelle und verbinden die beiden anderen Tabellen per LEFT JOIN:

```
SELECT mitarbeiter.nr, mitarbeiter.name AS mitarbeiter,
       aufgabe.name AS aufgabe

    FROM MitarbeiterAufgabe

    LEFT JOIN Mitarbeiter
           ON MitarbeiterAufgabe.mitarbeiter = Mitarbeiter.nr

    LEFT JOIN Aufgabe
           ON MitarbeiterAufgabe.aufgabe = Aufgabe.nr
```

Das Ergebnis der Abfrage findet sich in der folgenden Abbildung 7. Dabei wurde die Spalte *name* der Tabelle *Mitarbeiter* als Ergebnisspalte *mitarbeiter* selektiert und *name* aus der Tabelle *Aufgabe* als Spalte *aufgabe*.

nr	mitarbeiter	aufgabe
5	Amy Wong	Einkauf
6	Bender	Wartung
6	Bender	Logistik
4	Hermes Conrad	Einkauf

Abbildung 7: Mitarbeiter und Aufgabe über die N:M Verbindungstabelle.