1 Verknüpfung von Tabellen

Wenn mehrere Tabellen in einem Zusammenhang stehen, werden sie in einem Relationalen Datenmodell über Fremdschlüsselbeziehungen verbunden:

Kunde

KID	VorN	NachN
10	Peter	Kurz
21	Rita	Müller
34	Hans	Lang

Bestellung

BID	Datum	KID
31	2021-02-14	21
21	2021-04-21	10
34	2021-09-04	10

An den Daten erkennen wir, dass die Kundin „Müller“ eine Bestellung am 14.02.2021 getätigt hat und der Kunde „Kurz“ je eine am 21.04.2021 und am 04.09.2021. Wenn wir diese Information mit unseren herkömmlichen „Mitteln“ herausfinden wollen, müssen wir jedoch pro Kunde zwei SELECT Statements ausführen. Für Frau Müller wäre dies:

```
mysql> SELECT KID FROM Kunde WHERE NachN = "Müller";
+-----+
| KID |
+-----+
| 21  |
+-----+
1 row in set (0.10 sec)

mysql> SELECT Datum FROM Bestellung WHERE KID = 21;
+------------+
| Datum      |
+------------+
| 2021-02-14 |
+------------+
1 row in set (0.10 sec)
```

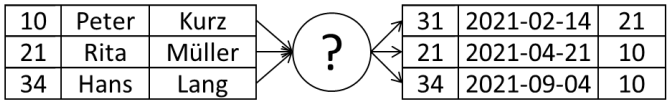
Wenn wir diese Information jedoch in einem einzigen Statement herausfinden wollen, müssen wir in einem SELECT zwei Tabellen ansprechen – wir müssen sie Verknüpfen.

2 CROSS JOIN

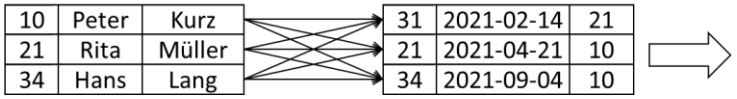
Ein naiver Ansatz wäre, einfach in einem SELECT beide Tabellen zu berücksichtigen. Der Einfachheit halber wollen wir die Zusammenhänge nicht für einen Kunden, sondern für alle sehen:

```
mysql> SELECT * FROM Kunde, Bestellung;
+-----+-------+--------+-----+------------+------+
| KID | VorN  | NachN  | BID | Datum      | KID  |
+-----+-------+--------+-----+------------+------+
|  10 | Peter | Kurz   |  21 | 2021-04-21 |   10 |
|  21 | Rita  | Müller |  21 | 2021-04-21 |   10 |
|  34 | Hans  | Lang   |  21 | 2021-04-21 |   10 |
|  10 | Peter | Kurz   |  31 | 2021-02-14 |   21 |
|  21 | Rita  | Müller |  31 | 2021-02-14 |   21 |
|  34 | Hans  | Lang   |  31 | 2021-02-14 |   21 |
|  10 | Peter | Kurz   |  34 | 2021-09-04 |   10 |
|  21 | Rita  | Müller |  34 | 2021-09-04 |   10 |
|  34 | Hans  | Lang   |  34 | 2021-09-04 |   10 |
+-----+-------+--------+-----+------------+------+
9 rows in set (0.01 sec)
```

Das Ergebnis ist aber erstmal nicht zufriedenstellend. Es handelt sich hier um einen „**Cross Join**“, welche immer dann entsteht, wenn wir der Datenbank nicht genügend Informationen über die Verknüpfung der involvierten Tabellen mitgeben. MySQL weiß schlichtweg nicht, wie Kunde und Bestellung zusammenhängen:



Die Lösung für MySQL ist dann, jeden Kunden mit jeder Bestellung zu verbinden:



KID	VorN	NachN	BID	Datum	KID
10	Peter	Kurz	21	2021-04-21	10
21	Rita	Müller	21	2021-04-21	10
34	Hans	Lang	21	2021-04-21	10
10	Peter	Kurz	31	2021-02-14	21
21	Rita	Müller	31	2021-02-14	21
34	Hans	Lang	31	2021-02-14	21
10	Peter	Kurz	34	2021-09-04	10
21	Rita	Müller	34	2021-09-04	10
34	Hans	Lang	34	2021-09-04	10

Hier sehen wir aber, dass nur die Datensätze korrekt sind, bei denen sowohl aus der Kunden- als auch aus der Bestellungstabelle die KID Werte identisch sind. Insofern können wir unser `SELECT` mit einem Filter dazu bringen, nur diese Datensätze auch auszugeben:

```
mysql> SELECT * FROM Kunde k, Bestellung b
     > WHERE k.KID = b.KID;
+-----+-------+--------+-----+------------+------+
| KID | VorN  | NachN  | BID | Datum      | KID  |
+-----+-------+--------+-----+------------+------+
|  10 | Peter | Kurz   |  21 | 2021-04-21 |   10 |
|  21 | Rita  | Müller |  31 | 2021-02-14 |   21 |
|  10 | Peter | Kurz   |  34 | 2021-09-04 |   10 |
+-----+-------+--------+-----+------------+------+
3 rows in set (0.01 sec)
```

Die Bedingung „`WHERE k.KID = b.KID`“ bezeichnen wir als JOIN Bedingung. Nun ist diese JOIN Bedingung im `WHERE` eher unübersichtlich, da wir `WHERE` im Regelfall für die Auswahl von Datensätzen und nicht für das Entfernen von fehlerhaften Zuordnungen verwenden wollen:

```
mysql> SELECT * FROM Kunde k, Bestellung b
     > WHERE k.KID = b.KID
     > AND k.NachN = "Müller";
+-----+-------+--------+-----+------------+------+
| KID | VorN  | NachN  | BID | Datum      | KID  |
+-----+-------+--------+-----+------------+------+
|  21 | Rita  | Müller |  31 | 2021-02-14 |   21 |
+-----+-------+--------+-----+------------+------+
1 row in set (0.01 sec)
```

Hierfür gibt es einen anderen Syntax, um diese „Vermischung“ von Auswahlkriterien zu vermeiden.

3 Der INNER JOIN

```
mysql> SELECT * FROM Kunde k
     > JOIN Bestellung b ON k.KID = b.KID
     > WHERE k.NachN = "Müller";
+-----+-------+--------+-----+------------+------+
| KID | VorN  | NachN  | BID | Datum      | KID  |
+-----+-------+--------+-----+------------+------+
|  21 | Rita  | Müller |  31 | 2021-02-14 |   21 |
+-----+-------+--------+-----+------------+------+
1 row in set (0.01 sec)
```

Dies ist die empfohlene Syntaxvariante für den JOIN zwischen Tabellen. Jede Verbindung einer Tabelle zu einer anderen über eine Fremdschlüsselbeziehung führt zu einem weiteren JOIN Eintrag. Beim JOIN über mehrere Tabellen können die einzelnen Tabellen nun nacheinander „hinzugejoined“ werden. Hier ein erweitertes Datenmodell:

Kunde

KID	VorN	NachN
10	Peter	Kurz
21	Rita	Müller
34	Hans	Lang

Bestellung

BID	Datum	KID
31	2021-02-14	21
21	2021-04-21	10
34	2021-09-04	10

Bestellpos

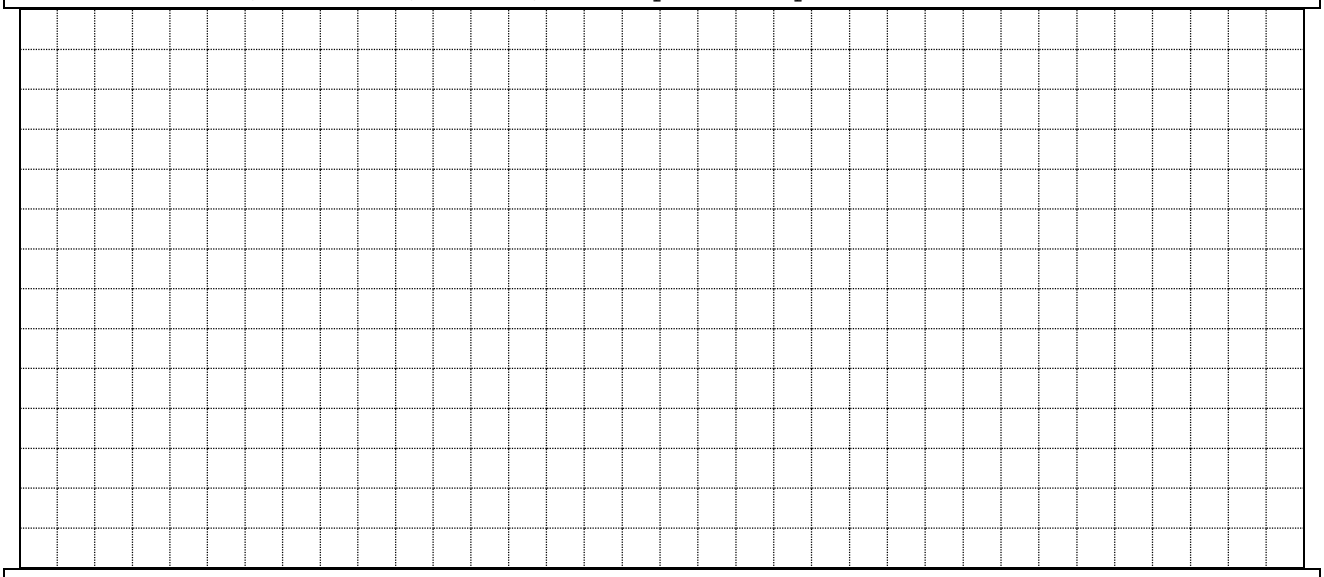
BPID	BID	PID	Anz
102	31	67	1
103	21	19	2
104	21	67	1
105	34	67	1

Produkt

PID	Bez	Preis
19	SenseSound	349.00
67	Mobil Blu	229.00
99	SlimSound	187.00

Ergänzen Sie nun die JOIN Bedingung für folgendes Statement:

```
SELECT k.VorN, k.NachN, k.KID, SUM(bp.Anz * p.Preis) FROM Kunde k
```



```
GROUP BY k.KID
WHERE b.Datum > "2021-02-01";
```

Dieser JOIN heißt „INNER JOIN“, weil nur die Werte jeder Tabelle angezeigt werden, welche tatsächlich einen „Joinpartner“ haben. Folgendes SELECT würde nur die inneren umrahmten Datensätze anzeigen:

```
SELECT * FROM Kunde k
JOIN Bestellung b ON b.KID = k.KID
JOIN Bestellpos bp ON bp.BID = b.BID
JOIN Produkt p ON p.PID = bp.PID;
```

KID	VorN	NachN	BID	Datum	KID	ZID	BID	PID	Anz	PID	Bez	Preis
34	Hans	Lang										
21	Rita	Müller	31	2021-02-14	21	102	31	67	1	67	Mobil Blu	229.00
10	Peter	Kurz	21	2021-04-21	10	103	21	19	2	19	SenseSound	349.00
10	Peter	Kurz	21	2021-04-21	10	104	21	67	1	67	Mobil Blu	229.00
10	Peter	Kurz	34	2021-09-04	10	105	34	67	1	67	Mobil Blu	229.00
										99	SlimSound	187.00

4 OUTER JOIN

Neben dem INNER JOIN existiert noch der OUTER JOIN, bei dem auch Daten ausgegeben werden, welche keinen Joinpartner aufweisen. Hierbei gibt es einen LEFT und einen RIGHT OUTER JOIN:

```
SELECT * FROM Kunde k
LEFT OUTER JOIN Bestellung b ON b.KID = k.KID
LEFT OUTER JOIN Bestellpos bp ON bp.BID = b.BID
LEFT OUTER JOIN Produkt p ON p.PID = bp.PID;
```

KID	VorN	NachN	BID	Datum	KID	ZID	BID	PID	Anz	PID	Bez	Preis
34	Hans	Lang	null	null	null	null	null	null	null	null	null	null
21	Rita	Müller	31	2021-02-14	21	102	31	67	1	67	Mobil Blu	229.00
10	Peter	Kurz	21	2021-04-21	10	103	21	19	2	19	SenseSound	349.00
10	Peter	Kurz	21	2021-04-21	10	104	21	67	1	67	Mobil Blu	229.00
10	Peter	Kurz	34	2021-09-04	10	105	34	67	1	67	Mobil Blu	229.00
										99	SlimSound	187.00

Bei mehreren Joins müssen – je nach Datenlage – alle Tabellen mit OUTER JOIN zusammengefasst werden. Die Felder, welche aufgrund fehlender Joinpartner keine Werte haben, werden mit `null` aufgefüllt.

```
SELECT * FROM Kunde k
JOIN Bestellung b ON b.KID = k.KID
JOIN Bestellpos bp ON bp.BID = b.BID
RIGHT OUTER JOIN Produkt p ON p.PID = bp.PID;
```

KID	VorN	NachN	BID	Datum	KID	ZID	BID	PID	Anz	PID	Bez	Preis
34	Hans	Lang										
21	Rita	Müller	31	2021-02-14	21	102	31	67	1	67	Mobil Blu	229.00
10	Peter	Kurz	21	2021-04-21	10	103	21	19	2	19	SenseSound	349.00
10	Peter	Kurz	21	2021-04-21	10	104	21	67	1	67	Mobil Blu	229.00
10	Peter	Kurz	34	2021-09-04	10	105	34	67	1	67	Mobil Blu	229.00
null	null	null	null	null	null	null	null	null	null	99	SlimSound	187.00

Kommerzielle Datenbanken wie Oracle oder DB2 beherrschen auch den FULL OUTER JOIN:

```
SELECT * FROM Kunde k
LEFT OUTER JOIN Bestellung b ON b.KID = k.KID
LEFT OUTER JOIN Bestellpos bp ON bp.BID = b.BID
FULL OUTER JOIN Produkt p ON p.PID = bp.PID;
```

KID	VorN	NachN	BID	Datum	KID	ZID	BID	PID	Anz	PID	Bez	Preis
34	Hans	Lang	null	null	null	null	null	null	null	null	null	null
21	Rita	Müller	31	2021-02-14	21	102	31	67	1	67	Mobil Blu	229.00
10	Peter	Kurz	21	2021-04-21	10	103	21	19	2	19	SenseSound	349.00
10	Peter	Kurz	21	2021-04-21	10	104	21	67	1	67	Mobil Blu	229.00
10	Peter	Kurz	34	2021-09-04	10	105	34	67	1	67	Mobil Blu	229.00
null	null	null	null	null	null	null	null	null	null	99	SlimSound	187.00