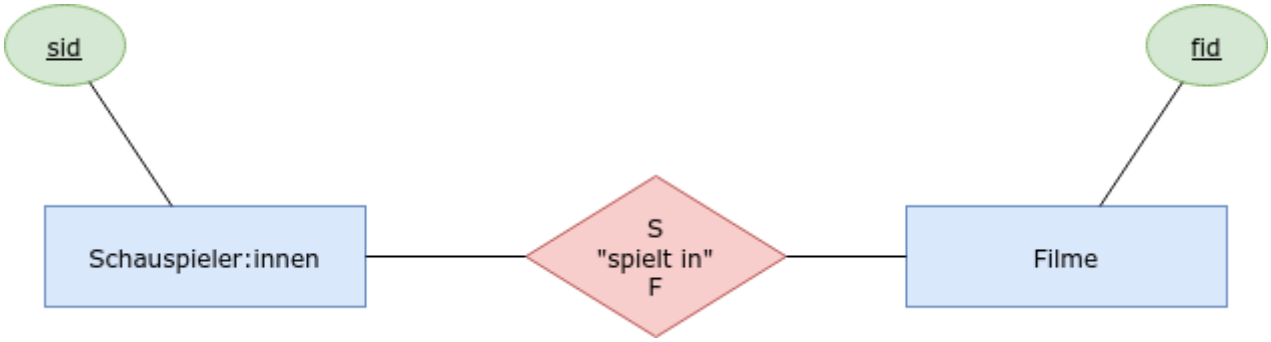
n-m Beziehungen

In normalisierten Datenbanken kommen häufig "Beziehungstabellen" zum Einsatz, welche die Verbindung zwischen den Entitäten herstellen. **n-m-Beziehungen benötigen immer eine solche Beziehungstabelle**.

Beispiel



Schauspieler

sid	Snachname	Svorname
1	DeVito	Danny
2	Crystal	Billy
3	Reiner	Rob
4	Ryan	Meg
5	Fisher	Carry
6	Belushi	John

Filme

fid	Titel
1	Blues Brothers
2	Harry And Sally
3	Throw Mama From The Train
4	The Wolf of Wall Street

Die Struktur der Beziehungstabelle sieht so aus:

`schauspieler_film(sfid, sid↑ , fid↑)`

Der Surrogatschlüssel `sfid` ist hier optional, man kann auch die Kombination der beiden "kpünstlichen" Schlüssel `sid` und `fid` als Primärschlüssel verwenden:

`schauspieler_film(sid↑ , fid↑)`

Das vollständige Schema sieht also so aus:

```
schauspieler_film(sfid, sid↑ , fid↑)
schauspieler(sid, SVorname, SNachname)
filme(fid, titel)
```

Eine kurze Recherche ergibt (unter anderem), dass Meg Ryan in "Harry And Sally" mitspielt. Um diese Beziehung abzubilden, muss in unserer schauspieler_film-Tabelle eine Zeile der Form

sfid	sid	fid
1	4	2

eingefügt werden. Der Umstand, dass John Belushi eine Rolle in Blues Brothers spielt, führt zu einer weiteren Zeile:

sfid	sid	fid
1	4	2
2	6	1

(A1)

Ergänze die Beziehungstabelle unter Verwendung entsprechender Ressourcen (Suchmaschine).

In manchen Fällen macht es Sinn, dass Beziehungstabellen neben den eigentlichen Beziehungen **zusätzliche Informationen speichern, die zu einer bestimmten Beziehung gehören**. Im Fall der Filmdatenbank könnte das z.B. die Gage sein oder ob eine Schauspielerin für die Roll in diesem Film einen Oskar erhalten hat.

Ergänze das Scheme der Beziehungstabelle um die beiden im Beispiel genannten Felder.

Importiere für die folgenden Übungen die Tabellen der normalisierten Zahnarztbedarfsdatenbank in deine Übungsdatenbank. (

zahnarztbedarf_2nf.zip

)

(A2)

Erstelle ein ER-Diagramm für die Datenbank. Überführe das ER Modell in ein relationales Datenbankschema.

(A3)

1. Gib ein SQL-Statement an, das alle Produkte der Firma mit Hilfe des Filters WHERE `hersteller.firma = 'Eisen Karl'` auflistet[1)].
2. Gib ein SQL-Statement an, das alle Bestellungen (Produkt und Anzahl) von Viktoria auflistet.
3. Gib ein SQL-Statement an, das den Rechnungsbetrag von Dr. Blutgesicht ausgibt.
4. Gib ein SQL-Statement an, das alle Doktoren ausgibt, die Zement gekauft haben
5. Gib ein SQL-Statement an, das alle Doktoren ausgibt, deren Rechnungsbetrag über 100EUR liegt

Lösung 1

```
SELECT produkte FROM hersteller, produkte, bestellungen
WHERE bestellungen.produkt_id = produkte.id
AND bestellungen.hersteller_id = hersteller.id
AND hersteller.firma = "Eisen-Karl"
```

Lösung 2

```
SELECT produkt, anzahl FROM hersteller, doktoren, produkte, bestellungen
WHERE bestellungen.produkt_id = produkte.id
AND bestellungen.doktor_id = doktoren.id
AND bestellungen.hersteller_id = hersteller.id
AND doktoren.vorname = "Viktoria"
```

Lösung 3

```
SELECT SUM(preis*anzahl) AS rechnung, doktoren.name FROM
hersteller,doktoren, produkte, bestellungen
WHERE bestellungen.produkt_id = produkte.id
AND bestellungen.doktor_id = doktoren.id
AND bestellungen.hersteller_id = hersteller.id
AND doktoren.name = "Blutgesicht"
```

Lösung 4

```
SELECT vorname, name FROM bestellungen, doktoren, produkte
WHERE bestellungen.produkt_id=produkte.id
AND bestellungen.doktor_id=doktoren.id
AND produkte.produkt="Zement";
```

Lösung 5

```
SELECT SUM(preis*anzahl) AS rechnung, doktoren.name FROM
hersteller,doktoren, produkte, bestellungen
WHERE bestellungen.produkt_id = produkte.id
AND bestellungen.doktor_id = doktoren.id
AND bestellungen.hersteller_id = hersteller.id
GROUP BY doktoren.name
HAVING rechnung >100
```

(Bonus 1)

Teste das folgende SQL Statement:

```
SELECT DISTINCT produkt, firma FROM produkte p
INNER JOIN bestellungen b ON p.id=b.produkt_id
INNER JOIN hersteller h ON h.id=b.hersteller_id
WHERE h.firma = "Eisen-Karl"
```

Was wird hier abgefragt? Experimentiere mit der WHERE Bedingung und mit den angezeigten Feldern.

1)

Du sollst also nicht "von Hand" zuerst die Hersteller ID nachschauen…

From:
https://www.info-bw.de/ -

Permanent link:
https://www.info-bw.de/faecher:informatik:oberstufe:datenbanken:nm_beziehungen:start

Last update: **06.05.2025 06:24**