

DB: Fortgeschrittenes Datenbankdesign

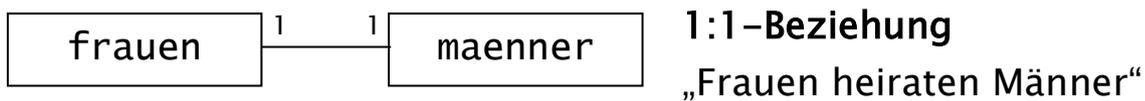
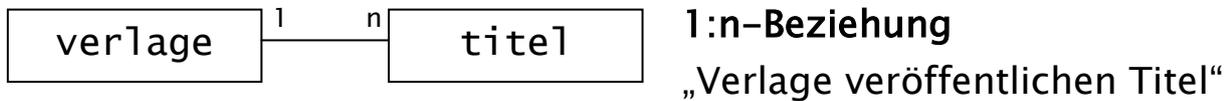
Ziele

- Die n:m-Beziehung in einer relationalen Datenbank darstellen können.
- Die referentielle Integrität bei n:m-Beziehungen richtig einsetzen können.
- Fortgeschrittene Techniken der Datenmodellierung einsetzen können.
- Insbesondere das ER-Diagramm in Chen-Notation erstellen können.
- Beim Feinentwurf das ER-Diagramm in angepasster UML-Notation erstellen können.
- Die Normalisierungsregeln verstehen und einsetzen können.
- Zu einer Problemstellung das Datenmodell erstellen und in MySQL implementieren können.

Relationales Datenbanksystem

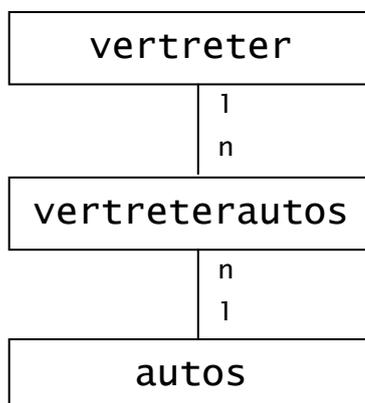
Datenbanksystem welches Daten anhand von Tabellen, die untereinander in Beziehung stehen, abspeichern lässt.

Arten von Beziehungen zwischen Tabellen



Realisierung der n:m-Beziehung durch

- Zwischentabelle vertreterautos
- zwei 1:n-Beziehungen

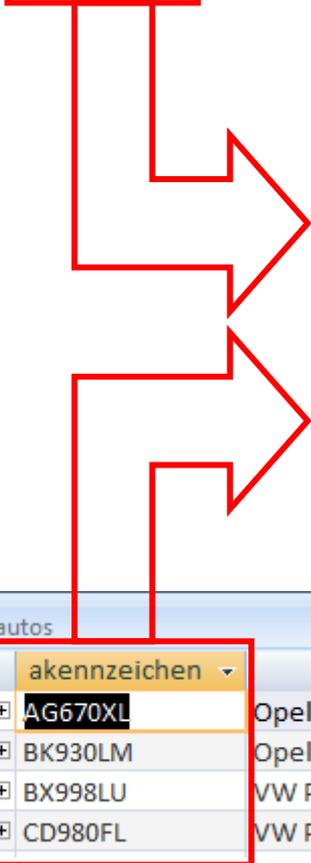


Konkret...

vnummer	vnachname	vvorname	vprovision	vgebdatum	vteilzeit
103	Sandri	Sonja	0,025	26.02.1979	<input type="checkbox"/>
104	Roner	Elisabeth	0,15	04.02.1979	<input checked="" type="checkbox"/>
105	Kalser	Sabine	0,1	22.09.1976	<input checked="" type="checkbox"/>
106	Linger	Thomas	0,05	10.03.1977	<input checked="" type="checkbox"/>
107	Thaler	Paul	0,175	02.12.1979	<input type="checkbox"/>
108	Ramoser	Mirco	0,15	14.11.1978	<input type="checkbox"/>
109	Steinegger	Kerstin	0,2	27.09.1979	<input checked="" type="checkbox"/>
110	Sparer	Achim	0,15	16.07.1979	<input checked="" type="checkbox"/>
111	Suma	Manuel	0,15	11.11.1979	<input checked="" type="checkbox"/>

vnummer	akennzeichen	vakilometer	vaunfallfrei
104	AG670XL	5300	<input checked="" type="checkbox"/>
108	CD980FL	13700	<input type="checkbox"/>
104	BX998LU	830	<input checked="" type="checkbox"/>
109	BX998LU	6500	<input checked="" type="checkbox"/>
104	CD980FL	150	<input checked="" type="checkbox"/>
110	AG670XL	2500	<input checked="" type="checkbox"/>
109	BK930LM	300	<input checked="" type="checkbox"/>
107	BK930LM	18000	<input type="checkbox"/>

akennzeichen	aname
AG670XL	Opel Corsa 1.5TD
BK930LM	Opel Astra 2.0 DTL
BX998LU	VW Passat 1.9TDI
CD980FL	VW Polo 1.2I



n:m-Beziehung mit *referentieller Integrität*

- Ein Auto kann mehreren Vertretern zugewiesen werden.
- Ein Vertreter kann mehrere Autos benutzen.
- Ein Auto kann einem Vertreter nicht mehrmals zugewiesen werden.
- Ein Vertreter kann auch kein Auto benutzen.
- Ein Auto kann auch keinem Vertreter zugewiesen werden.
- Was passiert, wenn Auto in autos gelöscht wird in vertreterautos (und in vertreter)?
- Was passiert, wenn Auto in autos sein Kennzeichen ändert?
- Was passiert, wenn Vertreter in vertreter gelöscht wird in vertreterautos (und in autos)?

```
CREATE TABLE vertreterautos(  
  vnummer INTEGER NOT NULL,  
  akennzeichen VARCHAR(7) NOT NULL,  
  ...  
  FOREIGN KEY (vnummer)  
    REFERENCES vertreter(vnummer)  
    ON DELETE CASCADE ON UPDATE CASCADE,  
  FOREIGN KEY (akennzeichen)  
    REFERENCES autos(akennzeichen)  
    ON DELETE CASCADE ON UPDATE CASCADE,  
  PRIMARY KEY (vnummer, akennzeichen)) ENGINE=InnoDB;
```

ACHTUNG: Es muss kein Index definiert werden, weil Primärschlüsseldefinition bereits Indexierung vornimmt.

```
SELECT v.vnachname, v.vvorname, va.akennzeichen,  
       a.aname, va.vakilometer  
FROM vertreter v, vertreterautos va, autos a  
WHERE v.vnummer = va.vnummer AND  
       va.akennzeichen = a.akennzeichen  
ORDER BY v.vnachname, v.vvorname;
```



vnachname	vvorname	akennzeichen	aname	vakilometer
Ramoser	Mirco	CD980FL	VW Polo 1.2I	13700
Roner	Elisabeth	CD980FL	VW Polo 1.2I	150
Roner	Elisabeth	BX998LU	VW Passat 1.9TDI	830
Roner	Elisabeth	AG670XL	Opel Corsa 1.5TD	5300
Sparer	Achim	AG670XL	Opel Corsa 1.5TD	2500
Steinegger	Kerstin	BK930LM	Opel Astra 2.0 DTL	300
Steinegger	Kerstin	BX998LU	VW Passat 1.9TDI	6500
Thaler	Paul	BK930LM	Opel Astra 2.0 DTL	18000

FRAGE: Wie müsste vertreterautos definiert werden, wenn abgespeichert werden sollte, wann welcher Vertreter welches Auto benutzt hat und ein Vertreter mehrmals dasselbe Auto benutzen kann (**ACHTUNG:** Primärschlüssel und Indexe!!!)

Vorgangsweise bei der Datenmodellierung

1. Schritt: Grobentwurf: Erstellen des Entity-Relationship-Modell

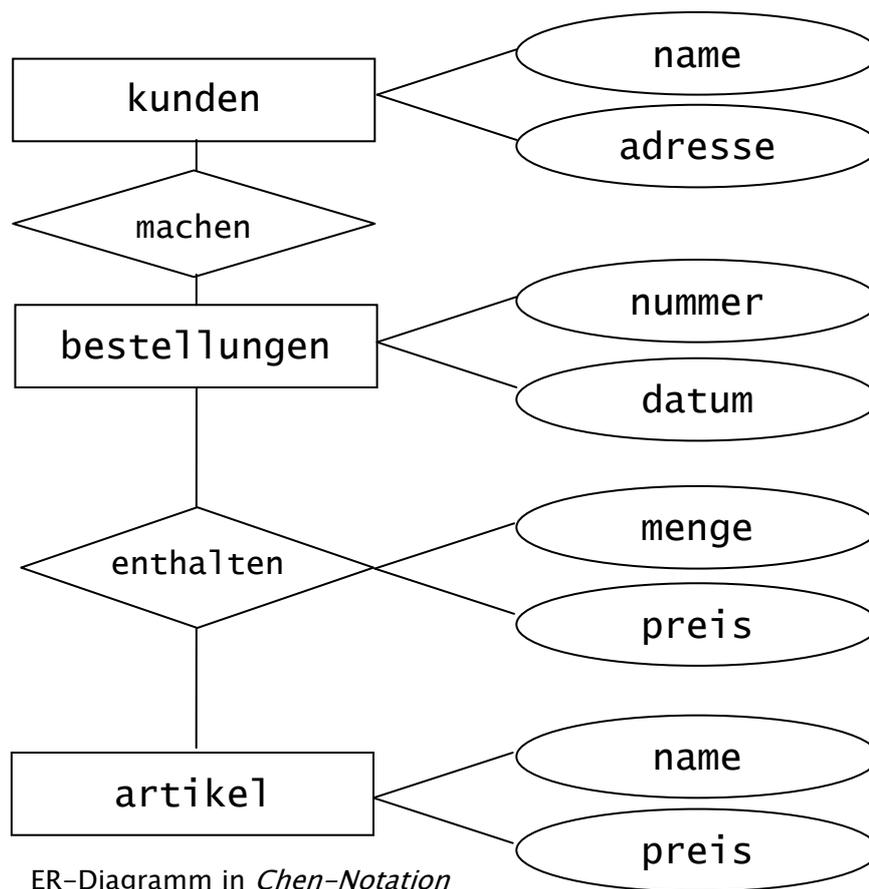
2. Schritt: Feinentwurf: Exakte Beziehungsdefinition, Definition der Primär- und Fremdschlüssel

3. Schritt: Feinstentwurf: Exakte Tabellendefinition

1. Schritt: Grobentwurf, Erstellen des Entity-Relationship-Modells

- Entität** Objekt der Wirklichkeit z. B. *Müller*.
Entitätstyp Typisierung gleichartiger Entitäten z. B. *Kunde*
- Attribut** Eigenschaft eines Entitätstyp z. B. *Name*
- Beziehung** Verknüpfung zwischen Entitäten z. B. *Kunden machen Bestellungen*

Kunden machen Bestellungen an einem bestimmten Datum. Bestellung erhält fortlaufende Nummer. In den Bestellungen werden Artikel in einer bestimmten Menge und zu einem bestimmten Preis bestellt.



- Man erhält einen groben Überblick über die abzuspeichernden Objekte und die herrschenden Beziehungen.
- Die genaue Art der Beziehung wird nicht festgelegt.
- *Primär-* und *Fremdschlüssel* werden nicht definiert.

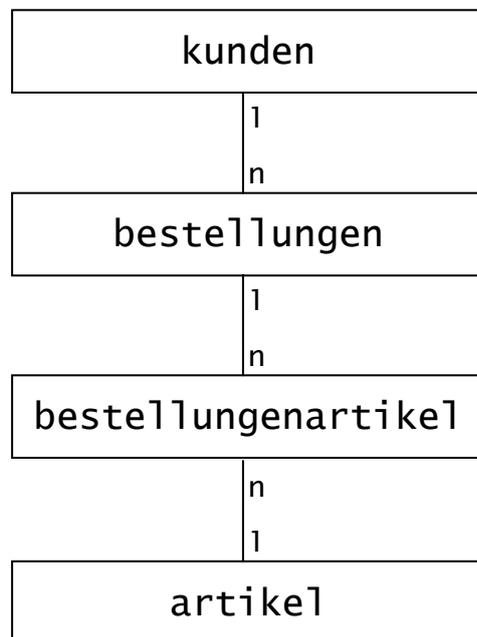
2. Schritt: Feinentwurf, exakte Beziehungsdefinition, PS und FS

kunden(knummer, knachname, kvorname, kstrassenr, kplz,
kort)

bestellungen(bnummer, knummer, bdatum)

artikel(anummer, aname, apreis)

bestellungenartikel(bnummer, anummer, bamenge, bapreis)



ER-Diagramm in angepasster
UML-Notation

- Exakte Datenfeldbenennung.
- Rückführung der Beziehungen auf 1:1- und 1:n-Beziehungen.
- Definition der Primär- und Fremdschlüssel.
- Exakte Datenfelddefinition nicht vorhanden.

3. Schritt: Feinstentwurf, exakte Datenfelddefinition

```
CREATE TABLE kunden(  
  knummer INTEGER NOT NULL AUTO_INCREMENT,  
  knachname VARCHAR(100) NOT NULL,  
  kvorname VARCHAR(100) NOT NULL,  
  kstrassenr VARCHAR(100) NOT NULL,  
  kplz VARCHAR(7) NOT NULL DEFAULT "39100",  
  kort VARCHAR(100) NOT NULL DEFAULT "Bozen",  
  PRIMARY KEY (knnummer)) ENGINE=InnoDB;  
  
CREATE TABLE bestellungen(  
  bnummer INTEGER NOT NULL AUTO_INCREMENT,  
  knnummer INTEGER NOT NULL,  
  bdatum DATETIME NOT NULL,  
  KEY (knnummer),  
  PRIMARY KEY (bnummer),  
  FOREIGN KEY (knnummer) REFERENCES kunden(knummer)  
    ON DELETE RESTRICT ON UPDATE CASCADE)  
  ENGINE=InnoDB;  
  
CREATE TABLE artikel(  
  anummer INTEGER NOT NULL AUTO_INCREMENT,  
  aname VARCHAR(100) NOT NULL,  
  apreis NUMERIC(10,2) NOT NULL,  
  PRIMARY KEY (anummer)) ENGINE=InnoDB;  
  
CREATE TABLE bestellungenartikel(  
  bnummer INTEGER NOT NULL,  
  anummer INTEGER NOT NULL,  
  bamenge INTEGER NOT NULL,  
  bapreis NUMERIC(10,2) NOT NULL,  
  PRIMARY KEY (bnummer, anummer),  
  FOREIGN KEY (bnummer) REFERENCES bestellungen(bnummer)  
    ON DELETE CASCADE ON UPDATE CASCADE,  
  FOREIGN KEY (anummer) REFERENCES artikel(anummer)  
    ON DELETE RESTRICT ON UPDATE CASCADE)  
  ENGINE=InnoDB;
```

- Datentypen werden exakt festgelegt.
- Eingabe erforderlich (NOT NULL).
- Standardwert (DEFAULT).
- Eindeutig (UNIQUE).
- Lösch- und Änderungsweitergabe (ON DELETE/UPDATE).

Regel zum richtigen Datenbankdesign: Normalisierung

Unter Normalisierung versteht man das richtige Zerlegen von Tabellen, um *Redundanzen* zu vermeiden, die bei Änderung der Daten zu *Inkonsistenzen* führen.

Vom Datenbanktheoretiker *Edgar F. Codd* entwickelt.

1. Normalform (1NF):

Jedes Datenfeld der Tabelle muss *atomaren Wertebereich* haben, und Tabelle darf *keine Wiederholungsgruppen* haben.

mitarbeiter	<u>(mnummer,</u>	<u>mname,</u>	<u>mkind1,</u>	<u>mkind2,</u>	<u>mkind3)</u>
	1	Mair Max	Rudi	Resi	Rupert
	2	Spiss Susi	Evi	NULL	NULL

mitarbeiter	<u>(mnummer,</u>	<u>mnachname,</u>	<u>mvorname)</u>
	1	Mair	Max
	2	Spiss	Susi

kinder	<u>(knummer,</u>	<u>kvorname,</u>	<u>mnummer)</u>
	1	Rudi	1
	2	Resi	1
	3	Rupert	1
	4	Evi	2

Praktischer Nutzen:

SELECT-Abfragen werden einfacher (z. B. Wie viele Kinder hat Mair Max, Sortiere der Mitarbeiter nach Vorname).

2. Normalform (2NF):

Eine Tabelle ist in 2NF, wenn sie in 1NF ist und jedes Datenfeld das nicht zum Primärschlüssel gehört, ist vom ganzen Primärschlüssel abhängig und nicht nur von Teilen des Primärschlüssels.

mitarbeiter	<u>mnummer</u>	<u>knummer</u>	mnachname	mvorname	kbeurteilung
	1	1	Mair	Max	gut
	1	2	Mair	Max	sehr gut
	2	2	Spiss	Susi	genügend
	3	1	Huber	Hugo	mäßig

mnachname und mvorname sind abhängig von mnummer.

mitarbeiterkurse	<u>mnummer</u>	<u>knummer</u>	kbeurteilung
	1	1	gut
	1	2	sehr gut
	2	2	genügend
	3	1	mäßig

mitarbeiter	<u>mnummer</u>	mnachname	mvorname
	1	Mair	Max
	2	Spiss	Susi
	3	Huber	Hugo

Praktischer Nutzen:

Jede Tabelle modelliert nur einen Sachverhalt und nur logisch zusammenhängende Informationen finden sich in einer Tabelle.

3. Normalform (3NF):

Eine Tabelle ist in 3NF, wenn sie in 2NF ist und ein Datenfeld das nicht zum Primärschlüssel gehört, darf nur direkt vom Primärschlüssel abhängen. Ein Datenfeld das nicht zum Primärschlüssel gehört, darf nicht von Datenfeldern abhängig sein, die nicht zum Primärschlüssel gehören.

mitarbeiter	<u>(mnummer,</u>	mnachname,	mvorname,	anummer,	aname)
	1	Mair	Max	1	Fertigung
	2	Spiss	Susi	2	Lager
	3	Obexer	Evi	1	Fertigung
	4	Huber	Hugo	3	Forschung

anummer ist abhängig von mnummer, weil ein Mitarbeiter nur in einer Abteilung arbeiten darf, aber aname ist abhängig von anummer, welche nicht zum Primärschlüssel gehört (*transitive Abhängigkeit*).

mitarbeiter	<u>(mnummer,</u>	mnachname,	mvorname,	<u>anummer.)</u>
	1	Mair	Max	1
	2	Spiss	Susi	2
	3	Obexer	Evi	1
	4	Huber	Hugo	3

abteilungen	<u>(anummer,</u>	aname)
	1	Fertigung
	2	Lager
	3	Forschung

Praktischer Nutzen:

Verbleibende thematische Durchmischungen in den Tabellen werden behoben.