Handout
Themenfeld: Datenbanken und SQL
Abschnitt: 02.01. Grundlagen des Relationenmodells

Autor: Thomas Krause
Stand: 14.11.2022 12:01:00

Inhalt

1 Relationenmodell: Grundlagen

1.1 Das Relationen-Modell

- Urheber E. F. Codd ca. 1970
- Basis bilden Relationen = Tabellen
- alle Daten werden als Werte in **Relationen** = zweidimensionalen Tabellen dargestellt
    - Spalten = Attribute
    - Zeilen = Tupel
- Benutzer-Zugriff auf die Daten erfolgt über deren Werte = Benutzer muß keine Kenntnisse über die Art und Weise der Speicherung haben, greift also nicht (direkt) auf Speicherort zu
- es existieren Operatoren für das Abrufen und Verarbeiten von Daten aus einzelnen und aus aufeinander verweisenden (**referenzierenden**) Tabellen
- Definition Datenbankschema
- **Datenstrukturierung:**
    - Beschreibung der logischen Aspekte der Daten, **unabhängig von der anschließenden Verarbeitung** in einer Software-Anwendung → Die Software-Anwendung bekommt ihre spezifische Sicht auf die Daten.
    - Definition von Integritätsbedingungen und Zugriffskontrollbedingungen
- **formatierte, feste Datenstrukturen:**
    - Beschreibung der zu speichernden Objekte durch Attribute und Attributwerte
    - Attribut besteht aus Attributbezeichner und einem Wert:
    - Beispiel: Objekt – Attribut – Attributwert
        - name = ‚Krause‘
        - kunden-nr = 101

Prägen Sie sich die erweiterten Eigenschaften eines Relationalen Schemas ein:

- alle Daten werden als Werte in **Relationen** = zweidimensionalen Tabellen dargestellt
    - logisch zusammengehörende Datensätze sind in derselben Tabelle z.B. …
    - Spalten = Attribute (Attribut = Spaltenüberschrift) = Eigenschaft des Datenobjekts
    - Zeile = Datenobjekt = Datensatz = logisch zusammengehörende Daten = werden auch **Tupel** genannt
    - zwischen den Datensätzen verschiedener Tabellen existieren logische Zusammenhänge/ Abhängigkeiten
- **eine** Tabellen-Zeile = **ein** Datensatz = **ein** Tupel = **ein** Datenbankobjekt
    - Tupel werden durch ihre Attributwerte identifiziert.
    - Es sind nur atomare Attributwerte zugelassen.
    - Attribut bezieht seine Werte aus einem unstrukturierten Wertebereich → keine Mengen, Aufzählungstypen, Wiederholungsgruppen, …
    - Die Reihenfolge der Tupel in der Relation ist nicht definiert.
    - Es erfolgt keine Addressierung der Datensätze z.B. durch Positionsangabe.
- **eine** Tabellen-Spalte entspricht **einem** Attribut
    - Die Reihenfolge der Attribute = Spalten ist nicht festgelegt.
    - Tupel werden eineindeutig durch den Attributwert identifiziert, der als Primärschlüssel festgelegt wurde.
    - Der Primärschlüssel wird im Rahmen des Datenbankentwurfs aus den Schlüsselkandidaten ausgewählt.
    - Für alle Schlüsselkandidaten gilt:
        - sie sind eindeutig
        - sie sind minimal

Ein relationales Schema enthält:

- **Domänenregeln**: mögliche Werte für Attribute werden eingeschränkt z.B. durch Festlegung von Datentypen (siehe weitere Infos im Anschluß)
- **Primärschlüssel**: identifizieren die einzelnen Tupel (= Tabellenzeile) eindeutig
- **weitere Schlüssel**: müssen eindeutig sein
- **Fremdschlüssel**: Attribut in einer Tabelle, das in einer anderen Tabelle als Primärschlüssel dient → Herstellen eindeutiger Beziehungen zwischen den Tabellen
- **weitere Geschäftsregeln**: Integritätsregeln, die sich durch logische Bezüge von Daten auf andere Daten ergeben

1.2 Relationales Schema → Datentypen

- Attribute haben unterschiedliche Wertebereiche = Domänen, aus denen sie ihre Werte beziehen
- hier einige allgemeine Beispiele ohne Bezug zu einem konkreten DBMS:
    - CARDINAL = natürliche Zahlen (0, 1, 2, 3, …)
    - INTEGER = ganze Zahlen mit Vorzeichen
    - NUMERIC, FLOAT, DECIMAL = Dezimalzahlen, Komma-Zahlen
    - ZAHL = Zahl (ohne genaue Festlegung)
    - STRING, TEXT, (MEMO) = Zeichenketten
    - CHAR = einzelne Zeichen
    - BOOLEAN, JA/ NEIN = logische Werte, Wahrheitswerte (1|0, ja|nein)
    - DATUM/ UHRZEIT = Datum und Zeit
    - AutoWert = vom DBMS vergebener numerischer Wert
- in Abhängigkeit vom DBMS können weitere individuelle Domänen definiert werden
    - feste Bezeichnungen, Fachbegriffe
    - Statusvariablen
    - Mengen mit festgelegten Werten (z.B. Steuerklassen, Wochentage, andere Kategoriebezeichnungen)

Wertebereiche = Domänen = Feldtypen in MySQL:

- Überblick über die Datentypen
    - numerische Datentypen
    - Datums- und Zeittypen
    - String-Typen
- siehe auch Quelle: MySQL-Referenzhandbuch 5.1 Kapitel 11. Datentypen

1.2.1 Numerische Datentypen

Datentyp	Beschreibung	Platzbedarf
BIT [(M)]		
TINYINT [(M)] [UNSIGNED] [ZEROFILL]		
BOOL, BOOLEAN		
SMALLINT [(M)] [UNSIGNED] [ZEROFILL]		
MEDIUMINT [(M)] [UNSIGNED] [ZEROFILL]		
INT [(M)] [UNSIGNED] [ZEROFILL]	(auch INTEGER); Parameter M für Darstellungsbreite und ggf. Auffüllung führender Nullen bei der Ausgabe	
BIGINT [(M)] [UNSIGNED] [ZEROFILL]		
FLOAT [(M, D)] [UNSIGNED] [ZEROFILL]		
DOUBLE [(M, D)] [UNSIGNED] [ZEROFILL]		
DECIMAL [(M [,D])] [UNSIGNED] [ZEROFILL]	(auch DEC)	

HINWEIS:

- ZEROFILL → unsigned
- SERIAL → ist ein Alias für BIGINT UNSIGNED NOT NULL AUTO_INCREMENT UNIQUE
- SERIAL DEFAULT VALUE → ist in der Definition einer Integer-Spalte ein Alias für NOT NULL AUTO_INCREMENT UNIQUE

1.2.2 Datums- und Zeittypen

Datentyp	Beschreibung	Platzbedarf
DATE		
DATETIME		
TIMESTAMP [(M)]		
TIME		
YEAR [(2|4)]		

1.2.3 Stringtypen

Datentyp	Beschreibung	Platzbedarf
[NATIONAL] **CHAR** (M) [BINARY|ASCII|UNICODE]	spezieller Fall CHAR	
[NATIONAL] **VARCHAR** (M) [BINARY]		
BINARY (M)		
VARBINARY (M)		
BLOB [(M)]		
TEXT [(M)]		
ENUM ('value1', 'value2', ...)		
SET ('value1', 'value2', ...)		

HINWEIS:

- für die Datentypen BLOB und TEXT existieren jeweils Varianten als TINY~, MEDIUM~, LONG~ mit verschiedenen Kapazitäten

1.2.4 Das Relationen-Modell

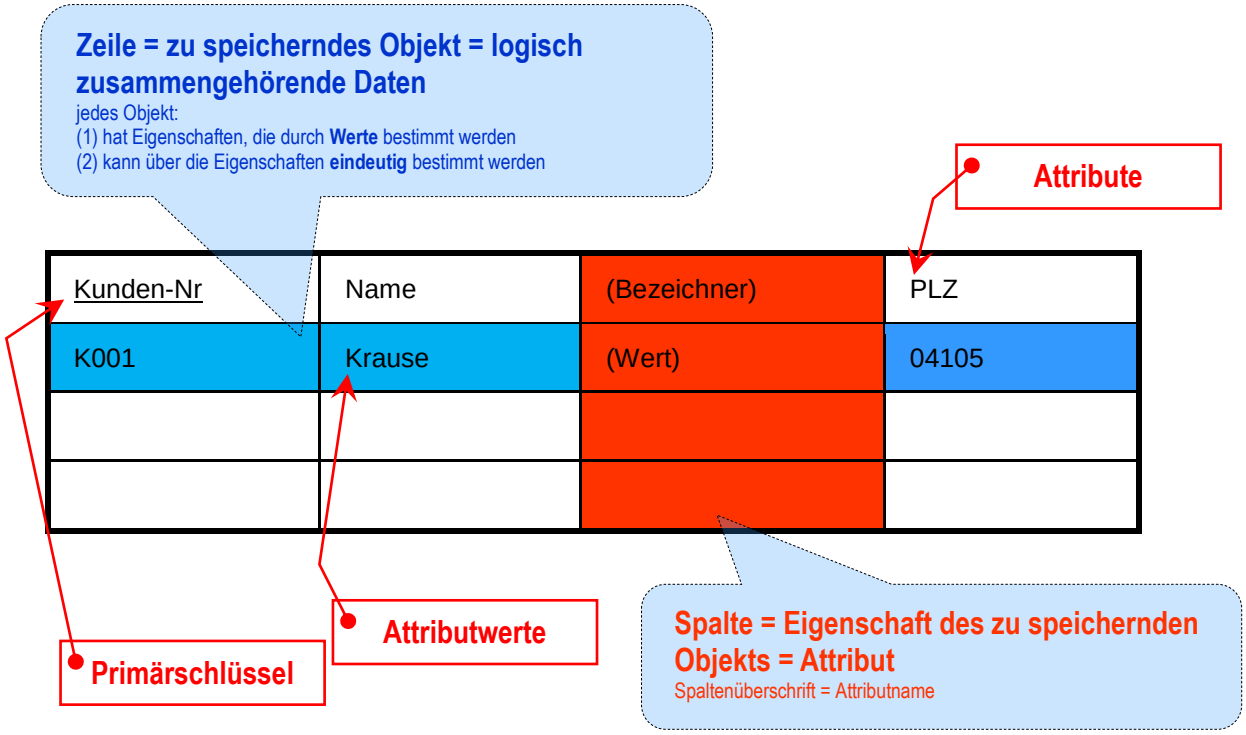
- Relationen-Modell ist eins von vielen, aber das am häufigsten verwendete (welche gibt es noch?)
- alle Daten werden in **Relationen** = zweidimensionalen Tabellen gespeichert: Spalten = Attribute, Zeilen = Tupel; die Tabellen können aufeinander verweisen = referenzieren

Kunden-Nr.	Name	(Bezeichner)	Ort
K011	Krause	(Wert)	Berlin
K099	Meier	10117	Berlin
K003	Müller	10120	Berlin

Spalte = Eigenschaft = Attribut

Kunden-Nr.	Name	PLZ	Ort
K001	Krause	10115	Berlin

Zeile = Tupel = zu speicherndes Objekt = Datensatz



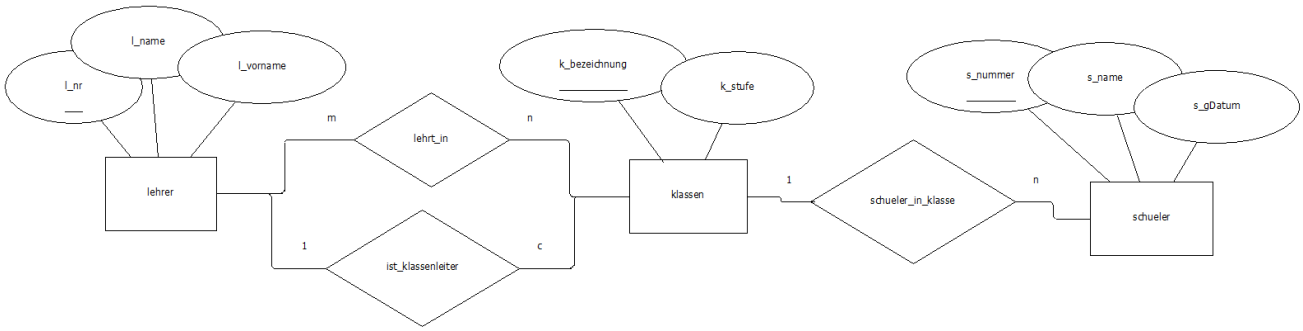
- alle Daten werden als Werte in **Relationen** = zweidimensionalen Tabellen dargestellt
  - logisch zusammengehörende Datensätze werden in derselben Tabelle gespeichert
  - Spalten = Attribute (Attribut = Spaltenüberschrift) = Eigenschaft des Datenobjekts
  - zwischen den Datensätzen verschiedener Tabellen existieren logische Zusammenhänge/ Abhängigkeiten
- **eine** Tabellen-Zeile = **ein** Datensatz
  - Datensätze werden **nur** durch ihre Attributwerte identifiziert.
  - Es sind nur atomare Attributwerte zugelassen. (1. Regel der Normalisierung)
  - die Reihenfolge der Datensätze wird vom DBMS intern verwaltet und ist nach außen hin nicht sichtbar
- **eine** Tabellen-Spalte entspricht **einem** Attribut
  - Datensätze werden eindeutig durch den Attributwert identifiziert, der als Primärschlüssel festgelegt wurde.

2 Relationenmodell: Praktische Umsetzung

2.1 Wir setzen fort mit Phase 3: Logischer Entwurf

- **Ziel:** fertiges logisches Datenbankschema
- **(wesentlicher) Inhalt:** Umsetzung des ERM aus Phase 2 in ein sogenanntes **relationales Schema**, das dann praktisch in eine Datenbank-Software (DBMS) eingegeben werden kann
- **Arbeitsschritte:**
    1. aus jeder **Entität** wird eine **Tabelle**, Tabellen erhalten einen **Primärschlüssel** (siehe ERM)
    2. **Beziehungen** zwischen Tabellen werden über **Fremdschlüssel** oder ggf. über zusätzliche Tabellen hergestellt
    3. **Normalisierung** des Schemas: Anpassung des relationalen Schemas so, dass
        - keine Daten mehrfach gespeichert werden (= **Vermeidung von Redundanz**)
        - **keine widersprüchlichen** Daten gespeichert werden
        - die **gegenseitige Abhängigkeit** der Daten voneinander **minimiert** wird
        - (die speziellen Regeln für die Normalisierung werden später besprochen)

Das ist das ERM, das in ein RM umgewandelt werden soll:

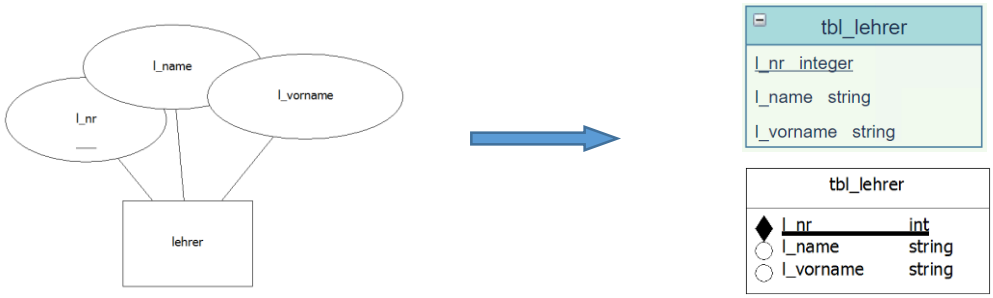


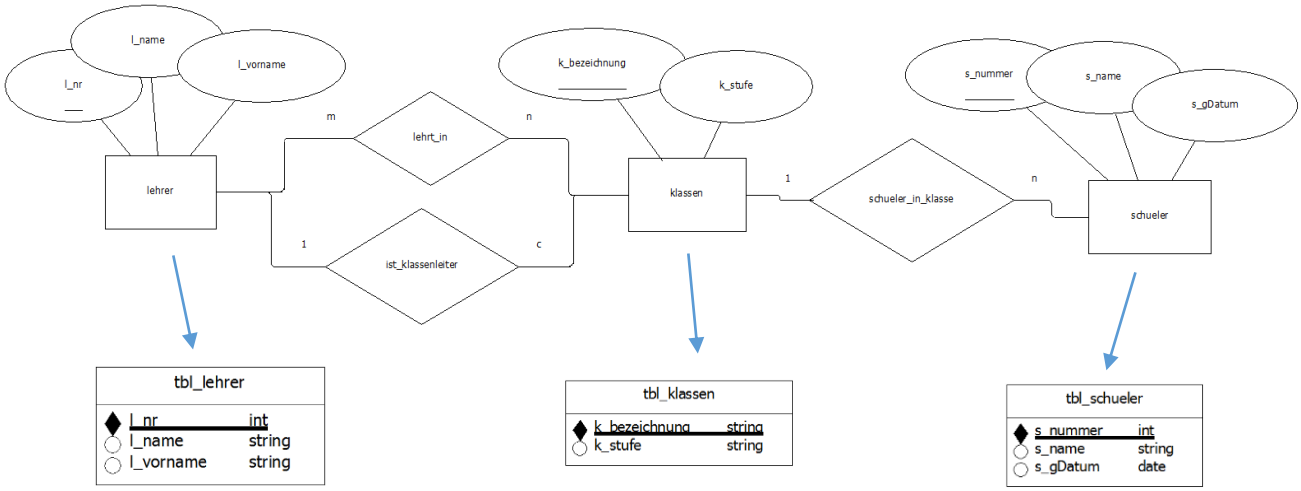
Welche Schritte sind bei der Umwandlung des ERM in ein RM zu erledigen?

- Entitäten umwandeln in Tabellen
- Beziehungen zwischen den Entitäten umwandeln

2.2 Schritt 1: alle Entitäten umwandeln in Tabellen

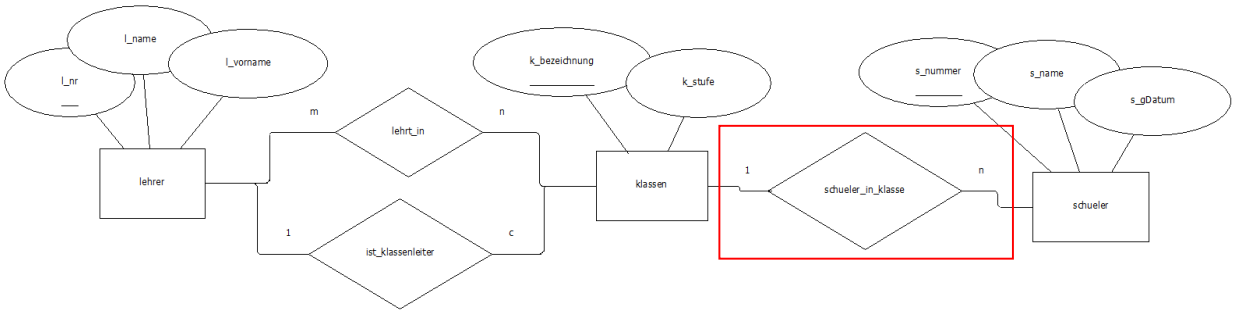
- jede einzelne Entität wird in jeweils eine Tabelle umgesetzt
- jedes Attribut wird zu einer Spalte in der jeweiligen Tabelle:
  - die Spalten erhalten einen eindeutigen Namen
  - jede Spalte erhält einen Wertebereich = Datentyp passend zu ihrer Verwendung
  - der Primärschlüssel steht i.d.R. an erster Stelle und wird unterstrichen
- in den so entstehenden Tabellen werden später die zu speichernden Objekte mit je einer Tabellenzeile eingetragen, also je Objekt ein Datensatz/ Tupel
- Hinweise zur Darstellung des Symbols für die Tabelle:
  - …

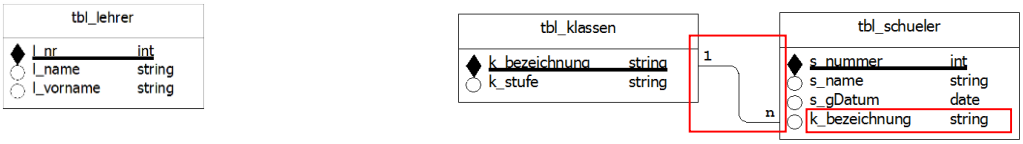


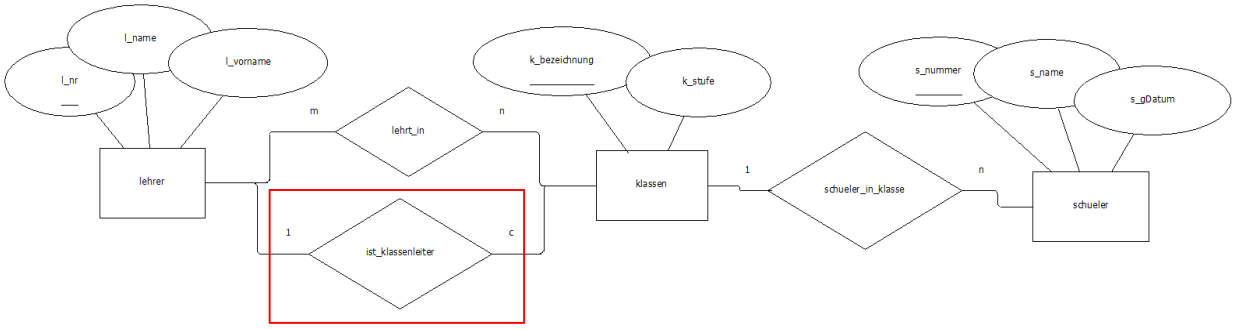


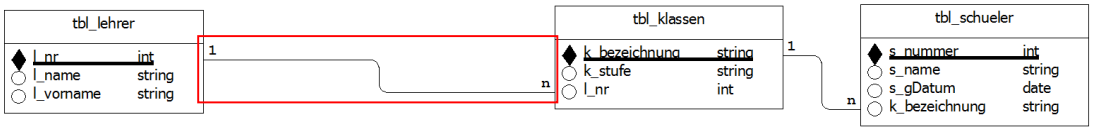
2.3 Schritt 2: alle Beziehungen umwandeln

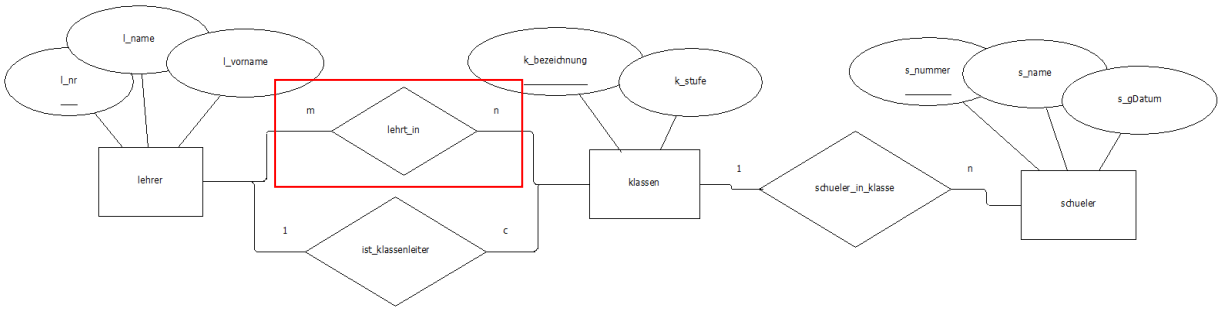
- Hinweis: Wir beschränken uns zunächst auf die Beziehungen mit den Kardinalitäten **1:n** (gleichbedeutend mit 1:m) und **m:n**
- Beziehungen mit Kardinalität 1:n bzw. 1:m können direkt abgebildet werden: zwischen den Tabellen wird eine Fremdschlüsselbeziehung hergestellt
- Beziehungen mit m:n können nicht direkt abgebildet werden → es ist jeweils eine Beziehungsmengen-Relation notwendig → eine m:n – Beziehung wird in 2 Beziehungen mit der Kardinalität 1:n aufgelöst

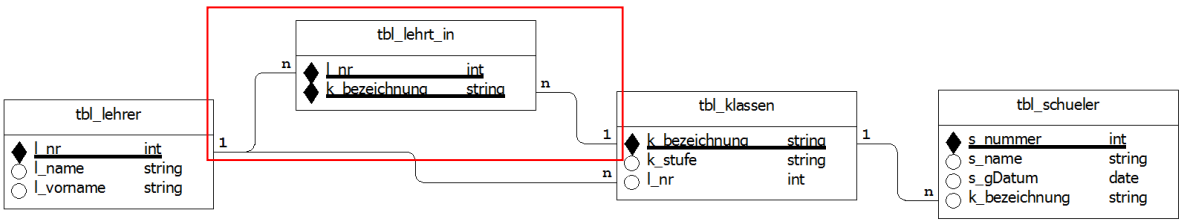





l_name
l_vorname
l_nr
lehrer
m
lehrt_in
n
k_bezeichnung
k_stufe
klassen
1
schueler_in_klasse
n
s_nummer
s_name
s_gDatum
schueler
1
ist_klassenleiter
c

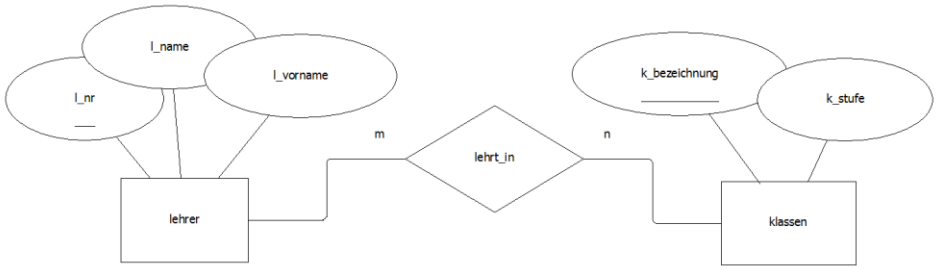

tbl_lehrer
l_nr int
l_name string
l_vorname string
1
n
tbl_klassen
k_bezeichnung string
k_stufe string
l_nr int
1
n
tbl_schueler
s_nummer int
s_name string
s_gDatum date
k_bezeichnung string

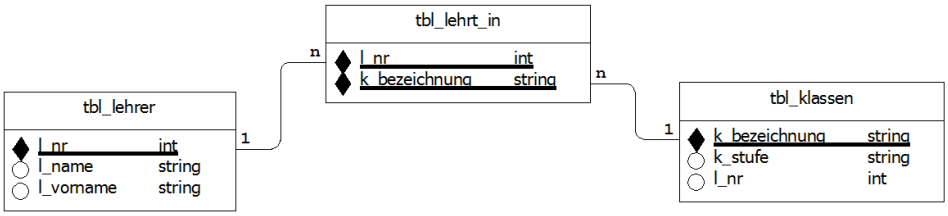

l_name
l_vorname
l_nr
lehrer
m
lehrt_in
n
k_bezeichnung
k_stufe
klassen
1
schueler_in_klasse
n
s_nummer
s_name
s_gDatum
schueler
1
ist_klassenleiter
c


tbl_lehrer
l_nr int
l_name string
l_vorname string
1
n
tbl_lehrt_in
l_nr int
k_bezeichnung string
n
n
tbl_klassen
k_bezeichnung string
k_stufe string
l_nr int
1
n
tbl_schueler
s_nummer int
s_name string
s_gDatum date
k_bezeichnung string

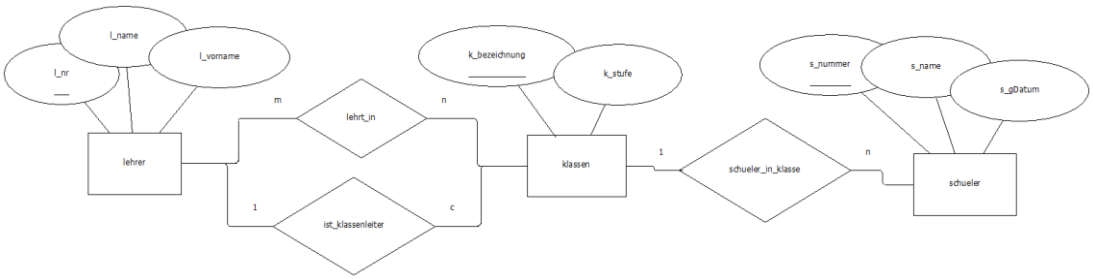
Ausschnitt auf die m:n-Beziehung:

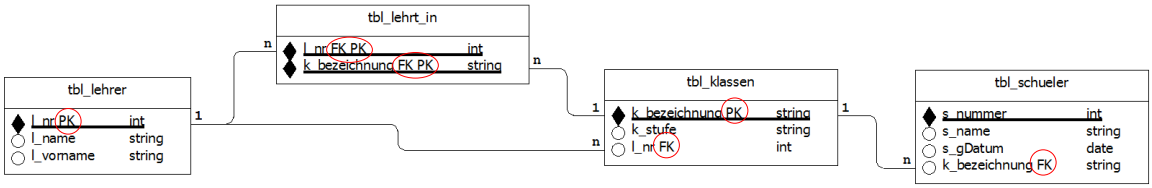
- die im ERM vorhandene m:n-Beziehung wird im RM durch eine Beziehungsmengen-Tabelle und 2 Stück 1:n-Beziehungen dargestellt
- Die Beziehungsmengen-Tabelle bekommt von den Entitäten, die ursprünglich mit der m:n-Beziehung verbunden waren, jeweils die Primärschlüsselfelder als Fremdschlüsselfelder eingetragen. Diese beiden Felder bilden in der Beziehungstabelle einen zusammengesetzten Primärschlüssel.
- hat die m:n-Beziehung eigene Attribute, werden diese in die Beziehungstabelle als zusätzliche Spalten eingetragen
- HINWEIS: Die hier genannte Umsetzung der m:n-Beziehung ist eine mögliche Lösung. Je nach Aufgabenstellung kann es in Zukunft abweichende Lösungen geben.



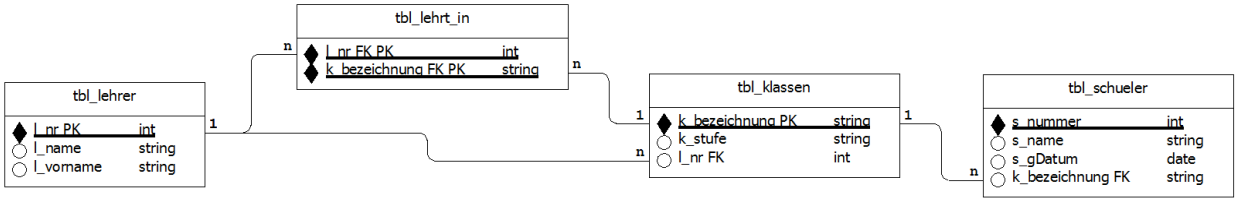


Hinweise zur Darstellung eines RM:





2.4 Beispiel 1:



2	9A
2	9B
1	8B
1	9C

1	Krause	Thomas
2	Meyer	Ricarda
3	Müller	Peter
4	Lehmann	Saskia
5	Schulze	Uwe

8A	8	3
8B	8	1
9A	9	5
9B	9	
9C	9	

122	Steubel	23.03.2010	8B
129	Santer	07.10.2009	8B
130	Maier	12.01.2010	8A
131	Schmidt	19.05.2008	9B
132	Rast	07.04.2008	9C

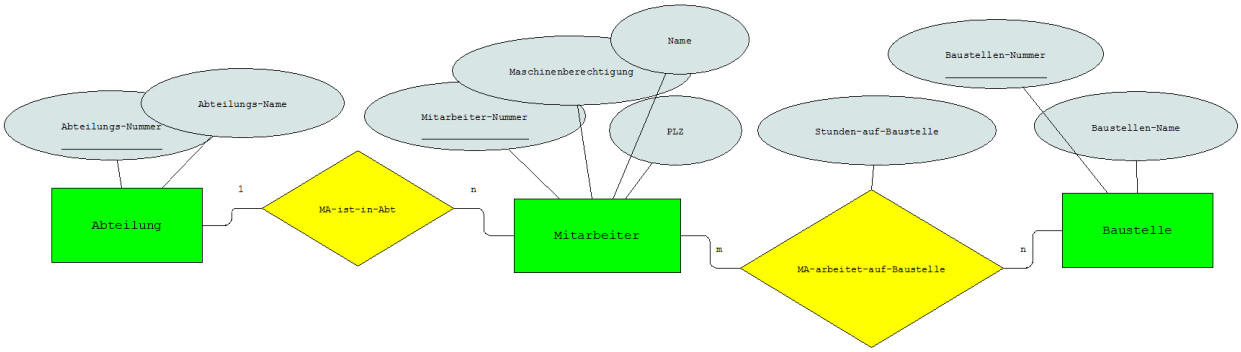
2.5 Beispiel 2:

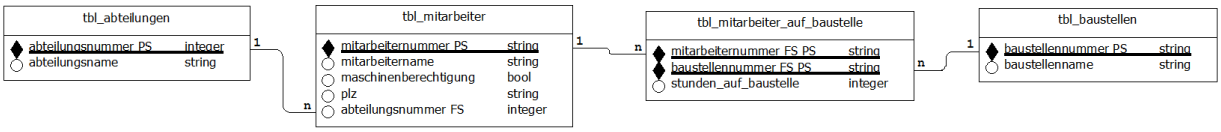
Szenario:

Die Firma "Hochbau" besteht aus mehreren Abteilungen. Die Abteilungen haben jeweils eine eindeutige Abteilungsnummer und einen beliebigen Namen. Alle Mitarbeiter haben eine eindeutige Mitarbeiter-Nummer. Sie gehören genau einer Abteilung an. In jeder Abteilung können mehrere Mitarbeiter sein. Für jeden Mitarbeiter ist zu speichern, ob er über eine Maschinenberechtigung verfügt. Zu jedem Mitarbeiter müssen der Name und die Postleitzahl des Wohnorts gespeichert werden.

Das Unternehmen arbeitet auf verschiedenen Baustellen. Die Baustellen haben eine eindeutige Baustellennummer und einen beliebigen Baustellennamen. Die Mitarbeiter können auf mehreren Baustellen tätig sein. Auf jeder Baustelle können mehrere Mitarbeiter tätig sein. Für jeden Mitarbeiter soll erfaßt werden, wieviel Stunden er auf welcher Baustelle gearbeitet hat.

Baustellen-nummer	Baustellen-name	Baustellen-Stunden	Abteilungs-nummer	Abteilungs-name	Maschinen-berechtigung	MA-Nummer	MA-Name	MA-PLZ
B021 B112	MIDL Kaufstadt	12 23	12	Ausbau	J	M010	Stein	04838
B253	GaleriaX	37	9	Hochbau	N	M009	Örtel	04105
B056 B112 B253	Brutto Kaufstadt GaleriaX	21 24 34	10	Haustechnik	J	M021	Hahn	04509
B056 B253	Brutto GaleriaX	8 24	9	Hochbau	N	M024	Holzer	04119





3 Zusammenfassung aller Handlungsschritte bis zur fertigen Datenbank

(1) Aufgabenstellung analysieren: Entitäten (mit Eigenschaften = Attributen) und deren Beziehungen untereinander bestimmen

(2) Entitäten mit Attributen in ERM eintragen, 1 Attribut muss Primärschlüssel werden

(3) Beziehungen zwischen Entitäten eintragen, Kardinalitäten 1:n bzw m:n bestimmen und eintragen

(4) Relationen-Modell aus dem ERM entwickeln

  (1) jede Entität wird eine Tabelle

  (2) jede 1:n bzw 1:m Beziehung in eine Fremdschlüsselbeziehung umwandeln

  (3) jede m:n Beziehung in eine Beziehungstabelle mit den 2 entsprechenden Fremdschlüsselbeziehungen umwandeln

(5) praktische Umsetzung im DBMS

  (1) neue Datenbank erstellen

  (2) die Tabellen nacheinander erstellen, zuerst die ohne Fremdschlüsselbeziehung (von außen nach innen) mit allen Attributen = Spalten, Primärschlüssel festlegen

  (3) Datenbank-Diagramm erstellen lassen

  (4) mit Drag&Drop die Beziehungen zwischen den Tabellen erstellen

(6) Daten in Datenbank eintragen ACHTUNG: Erst die Daten in die Tabellen OHNE Fremdschlüsselbeziehung eintragen (von außen nach innen)

(7) Datenbank ist fertig