Einführung

- Begriffe
- Datenbanketnwurf mit
  - Entity Relationship Modell
- Relationales Datenmodell
  - Schlüssel, Fremdschlüssel

Einführung

- Datenbanksystem (DBS):
  - System zur Beschreibung, Speicherung und Wiedergewinnung umfangreicher Datenmengen, die von verschiedenen Anwendungsprogrammen benutzt werden.

- Ein DBS besteht aus:
  - Datenbank und
  - Datenbank-Management-System

- Datenbank: “Sammlung aller gespeicherten Daten”

- Datenbank-Management-System
  - Programmsystem, das die DB verwaltet, fortschreibt und Zugriffe darauf regelt

Abstraktionsebenen eines DB Systems

Externe Sicht

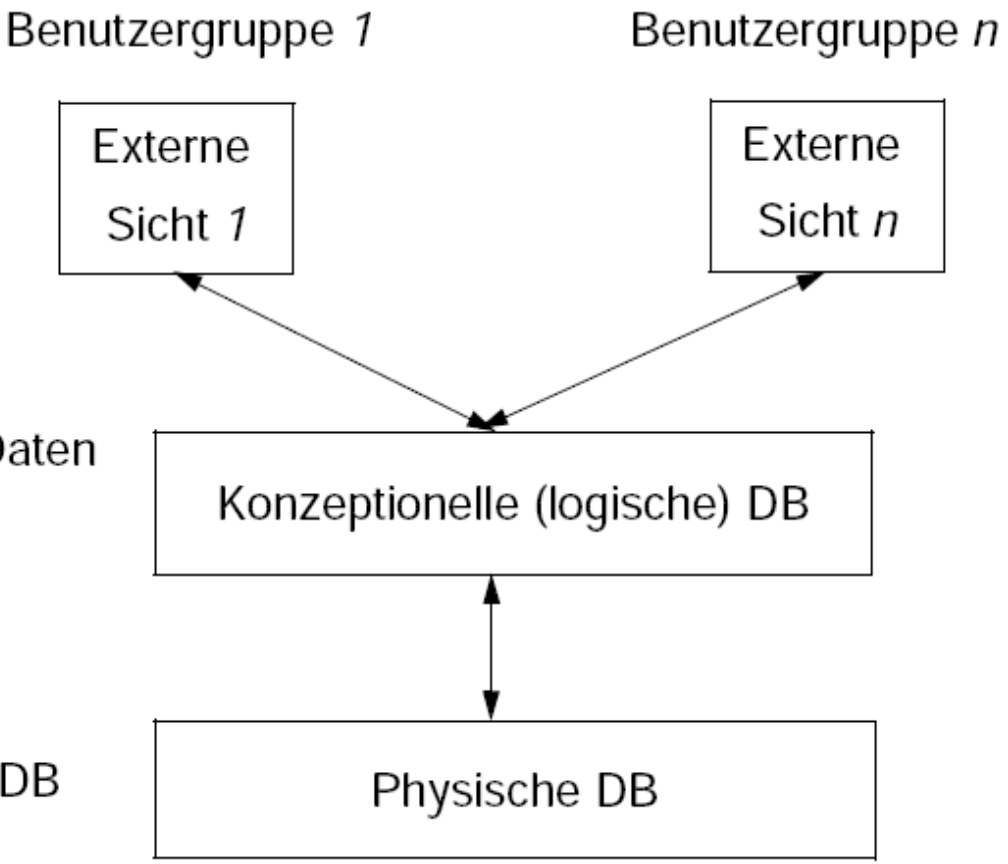
- Views
- Ausschnitte aus dem konzeptionellen Schema

Konzeptionelle Sicht

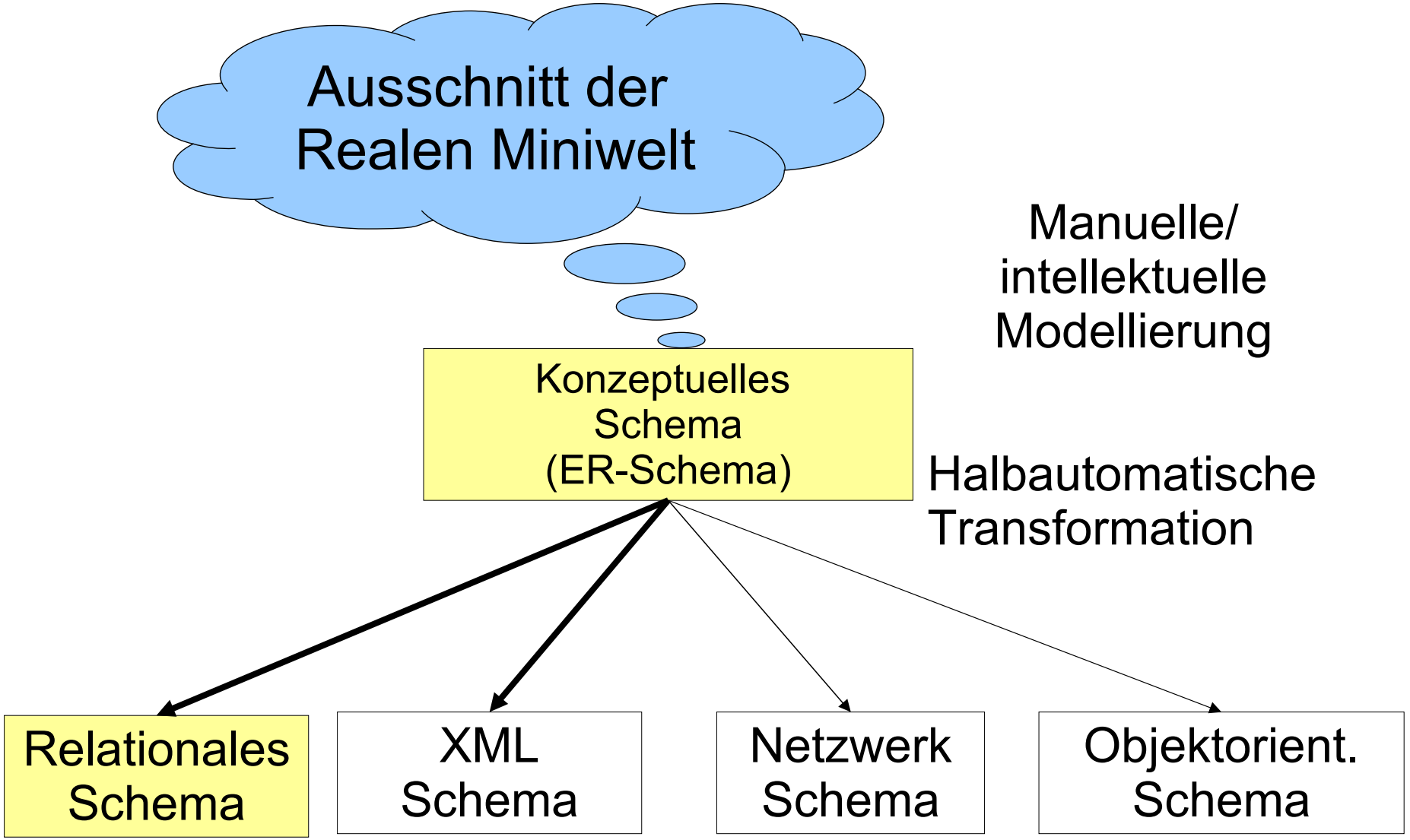
- einheitliche Darstellung aller Daten
- DBS bietet dazu Datenmodell mit entsprechender DDL

Interne Sicht

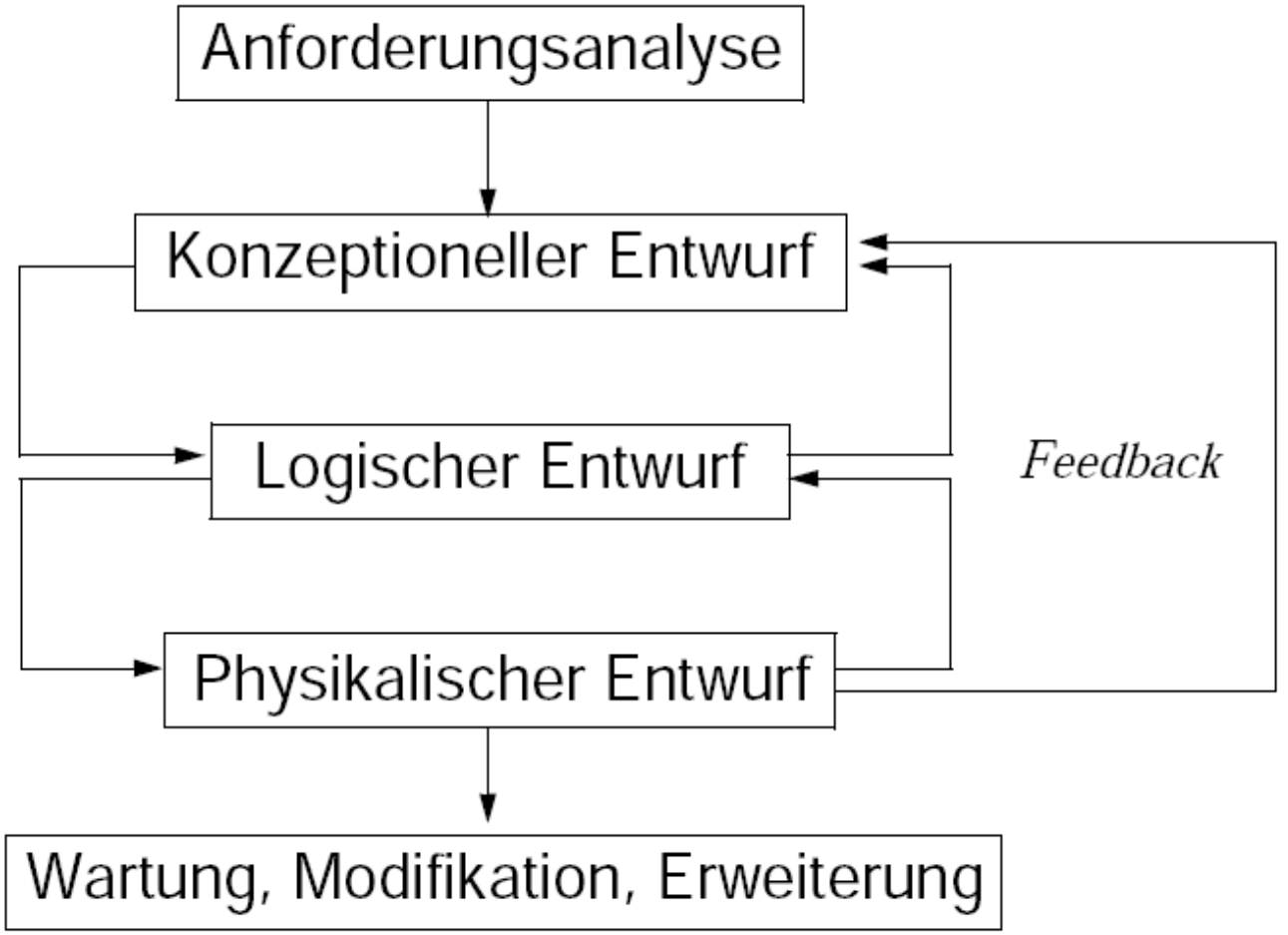
- Implementierung der konzep. DB
- Dateien, Zugriffspfade, etc.



Lebenszyklus eines DB Systems



Lebenszyklus eines DB Systems



Lebenszyklus eines DB Systems

Anforderungsanalyse

- Analyse und Spezifikation von
  - Daten
  - Datenbeziehungen
  - Transaktionen (Funktionen auf den Daten, die von den Applikationen benötigt werden)
- Randbedingungen
  - Leistungsanforderungen (Antwortzeiten etc.)
  - Sicherheit
  - HW/SW-Plattformen
  - Anwendungsrichtlinien
- Ansätze
  - *Knowledge Acquisition Design*: Der DB-Experte interviewt die zukünftigen Anwender und versucht deren Anforderungen festzustellen.
  - *Participatory Design*: Der DB-Experte und der zukünftige Anwender entwickeln das Design als Team.

Lebenszyklus eines DB Systems

Konzeptioneller Entwurf

- Basierend auf der Anforderungsanalyse werden die essentiellen Konzepte extrahiert und in einem konzeptionellen Datenmodell (DB-Schema) beschrieben.

- Entity-Relationship Modell
  - Formale Beschreibung der Datenobjekte und ihrer Beziehungen
  - Integration verschiedener Sichten

- Beschreibung der Transaktionen (Eingabe, Änderung, ...)

- Festlegung von Integritätsbedingungen für die Daten und für die Transaktionen

Lebenszyklus eines DB Systems

Logischer Entwurf

- Ausgehend vom DB-Schema (z.B. ER-Modell) wird ein DBS-spezifisches Datenmodell (z.B. Relationales Datenmodell) entwickelt.

- Wichtige Schritte bei der Transformation des DB-Schemas in das Datemodell:
  - Normalisierung und Denormalisierung
  - Primärschlüssel festlegen
  - Formulierung von Integritätsbedingungen und Transaktionen in DBS-spezifischem Datenmodell (relationale Anfragen, rel. Algebra etc.)
  - View-Definitionen (externe Sichten)
  - Zugriffsrechte

Lebenszyklus eines DB Systems

Physikalischer Entwurf (Umsetzung des Datenmodells)

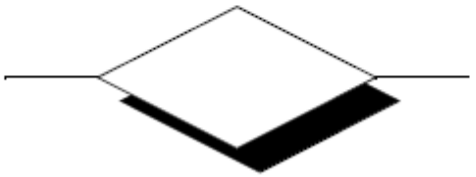
- konkrete Domäne für Attribute, z.B. CHAR(30), VARCHAR(20), NUMBER(4), NUMBER(7,2), DATE, ...
- Erzeugen der Relationen und evtl. Laden mit vorhandenen Daten
- Einrichten von Views, Benutzern, Zugriffsrechten
- Eingabe der Integritätsbedingungen (Anfragen, Prüfprogramme, ...)
- Definition geeigneter Indexe

Wartung, Modifikation und Erweiterung

- Arbeiten mit der Datenbank, Beseitigung von Fehlern
- Erweiterung, Erstellen von Anwendungsprogrammen
- Re-Engineering, Integration

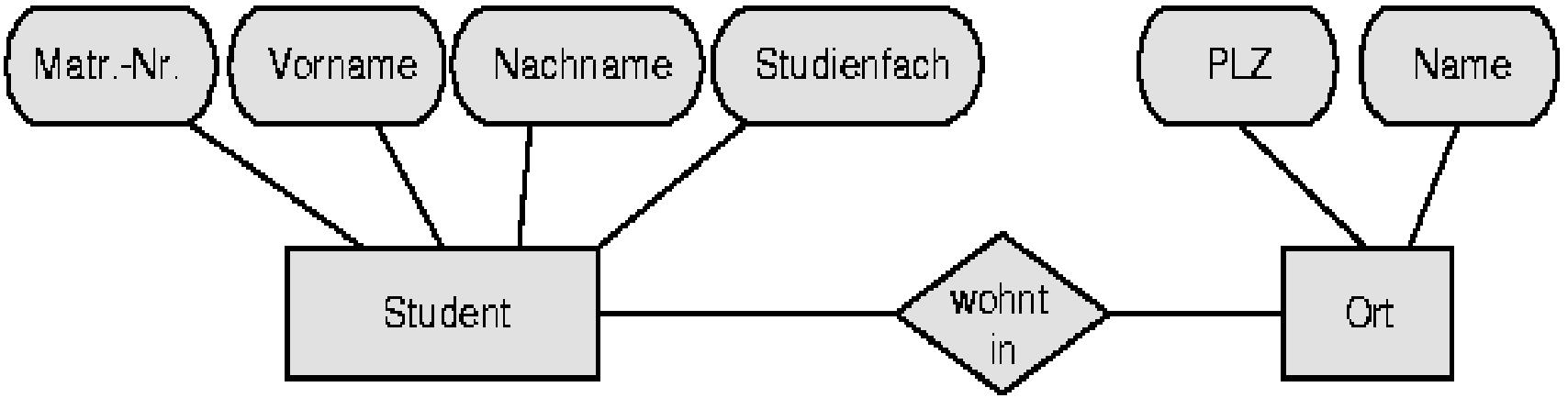
Entity-Relationship Modell (ERM)

Das ER-Modell beschreibt einen Ausschnitt der realen Welt durch:

	Entity	Relationship	Attribut
			
Set	Menge gleichartiger Objekte	Beziehungs-möglichkeit	Merkmal, Eigenschaft
Instanz	bestimmtes Objekt	konkrete Beziehung zwischen zwei Objekten	Eigenschaft eines Objekts

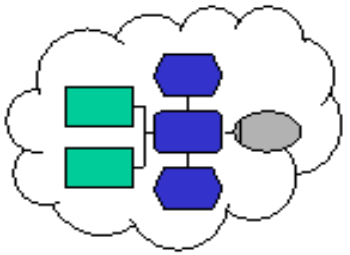
Modellierung (Entity-Relationship-Diagramm)

- Die graphische Darstellung erfolgt durch **Entity-Relationship-Diagramme (ER-Diagramm).**
- Entity-Typen werden durch Rechtecke, Beziehungen durch Rauten und Attribute durch Ovale dargestellt.

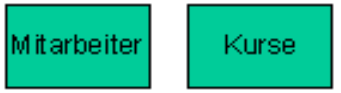


ERM: Vorgehensweise

A) Abstecken des Problemrahmens

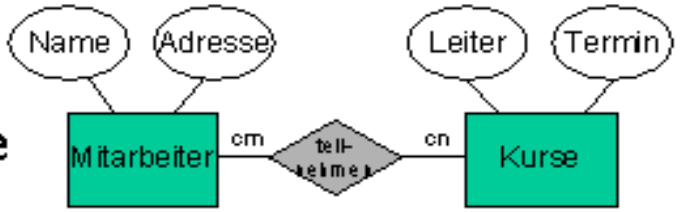


B) Auswahl der Entitätstypen

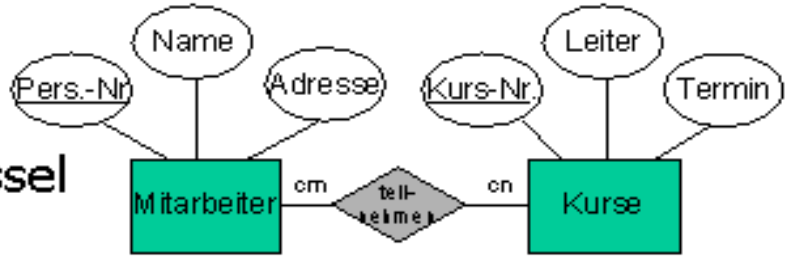


C) Festlegen der Beziehungen

D) Definition der Attribute



E) Definition der Identifikationsschlüssel



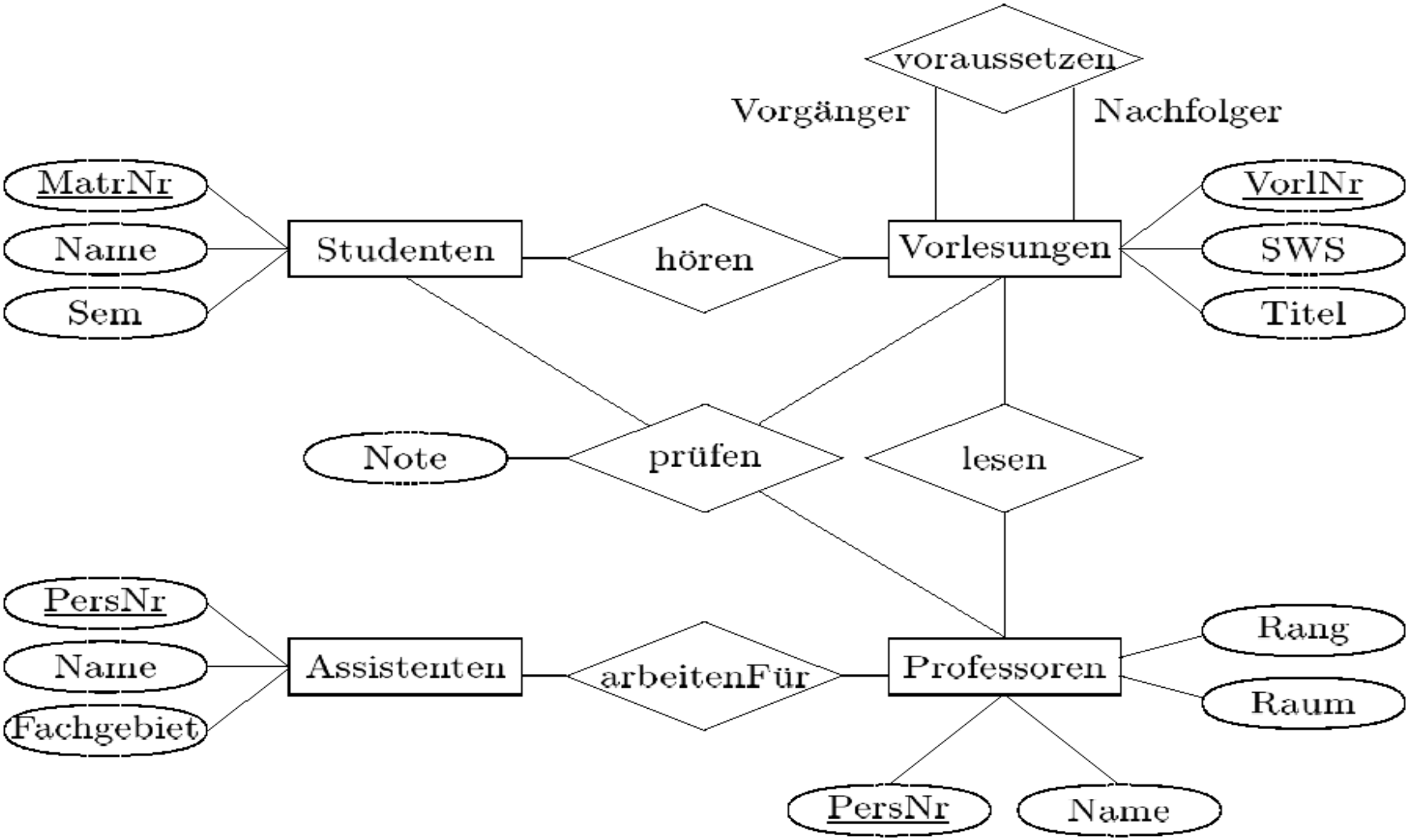
Beispiel: Uni-Verwaltungssystem

- Es soll ein univ. Verwaltungssystem gebaut werden. Dazu interviewt man mehere Personen:
- Professoren:
  Die Studierenden sollen bestimmte Lehrveranstaltungen hören (inskribieren) und über diese dann Prüfungen ablegen können.
- Verwaltungspersonal:
  Wir verwalten Professoren und deren Assistenten, wobei bei Professoren folg. Daten wichtig sind: Persnr, name, raum, rang. Professoren haben je ein eigenes Büro/Raum. Bei Assistenten wird zusätzlich der Fachbereich gespeichert. Nur Professoren halten Vorlesungen ab. Über Vorlesungen werden folg. Daten gespeichert: Vorlnr, titel, sws. Manche Vorlesungen setzen den Besuch anderer Vorlesungen voraus und können ihrerseits wieder Voraussetzung für andere Vorlesungen sein.
  Es werden auch Studentendaten gespeichert: matrnr, name, semester. Zu den Prüfungen muss gespeichert werden, welcher Student, bei welcher Vorlesung und bei welchem Professor welche Note wann gemacht hat.

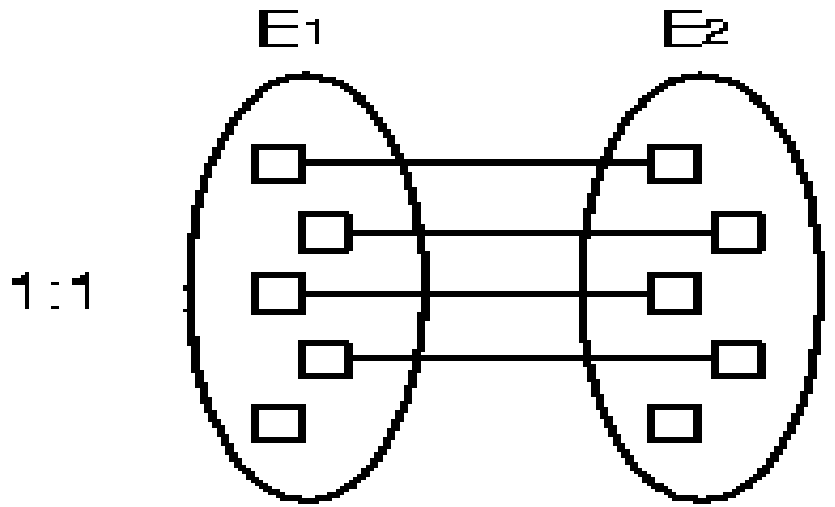
Beispiel: Uni-Verwaltungssystem

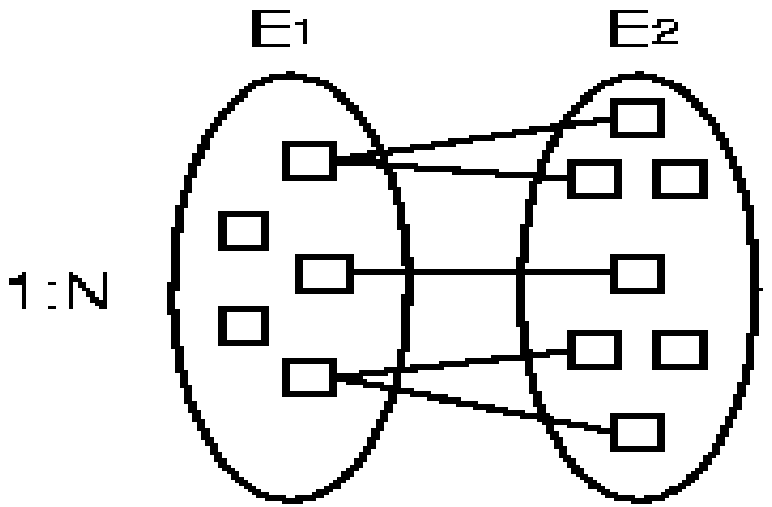
- Ein Vorlesungsverzeichnis soll erstellt werden können.
- Eine Mitarbeiterliste soll erstellt werden können, gereiht nach Professor und seinen Assistenen.
- Ein Inskriptionsverzeichnis: Welcher Student, welche Vorlesung(en) besucht)
- Ein Prüfungsverzeichnis: Welcher Student bei welchem Professor zu welcher Vorlesung wann, welche Note gemacht hat
- Eine Webanbindung (Vorlesungsverzeichnis) soll realisiert werden.
- Eine MS-Access / Openoffice base Anwendung zur Verwaltung der Daten und zur Berichterzeugung soll erstellt werden.

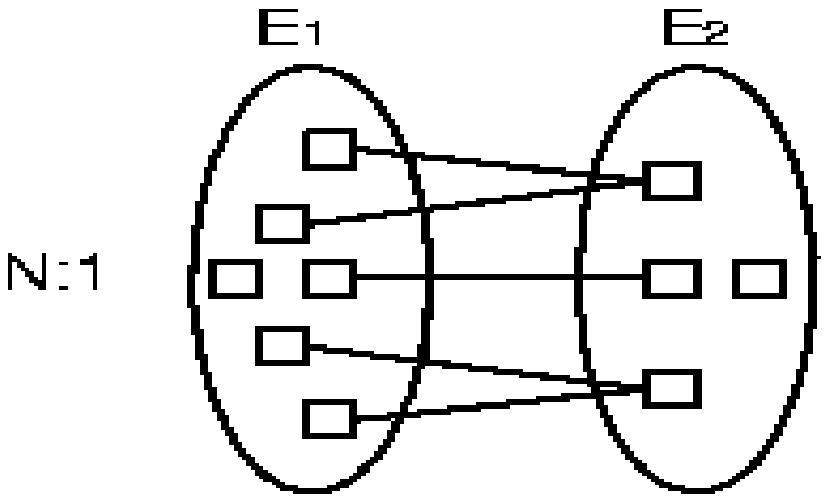
Beispiel: ERD: Universitätsinformationssystem

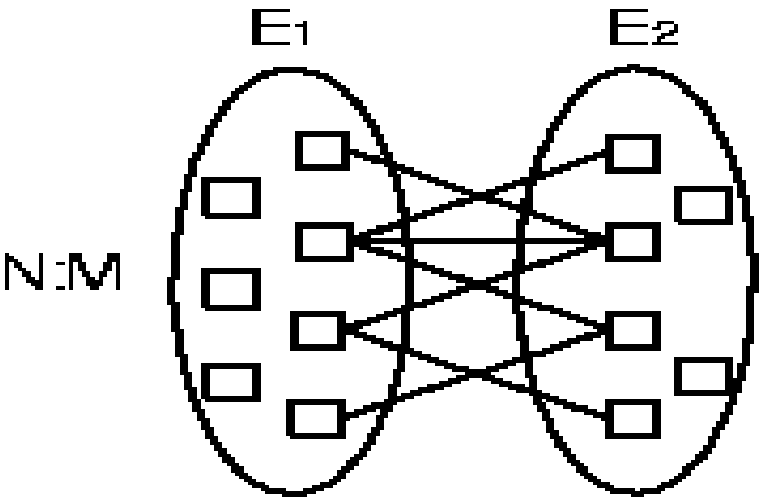


Kardinalität der Beziehung (1:n, n:m)

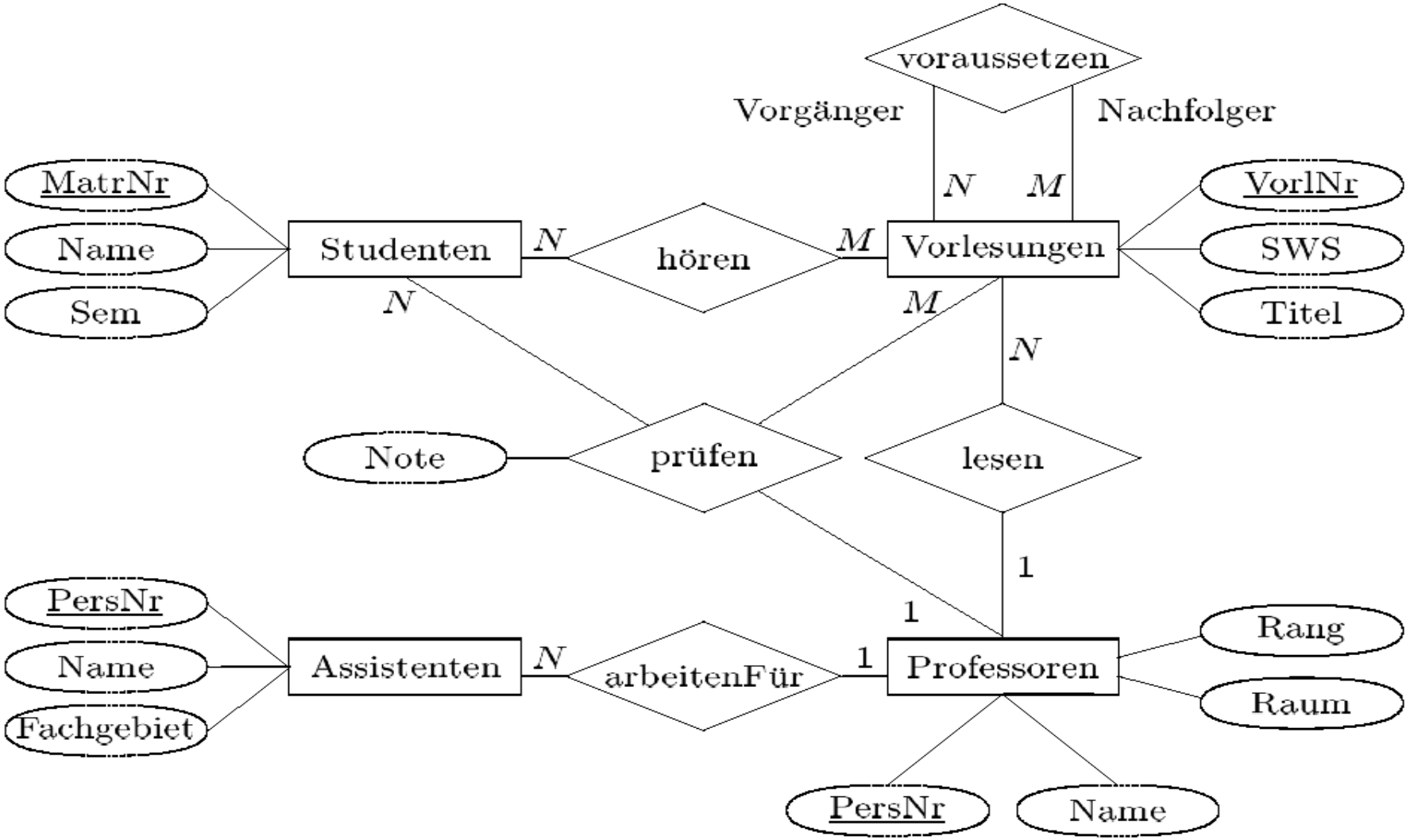




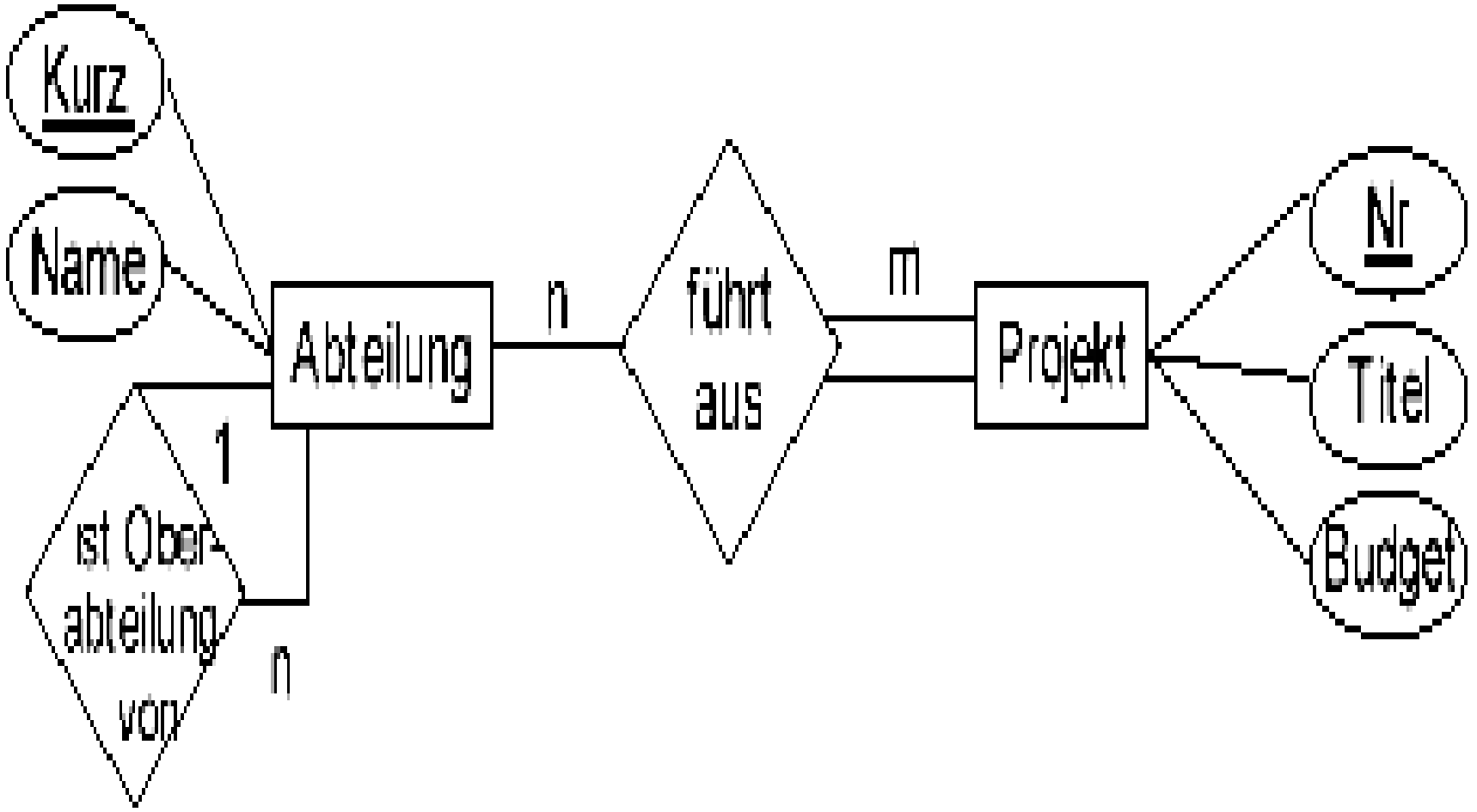




Uni-Beispiel mit Kardinalitäten

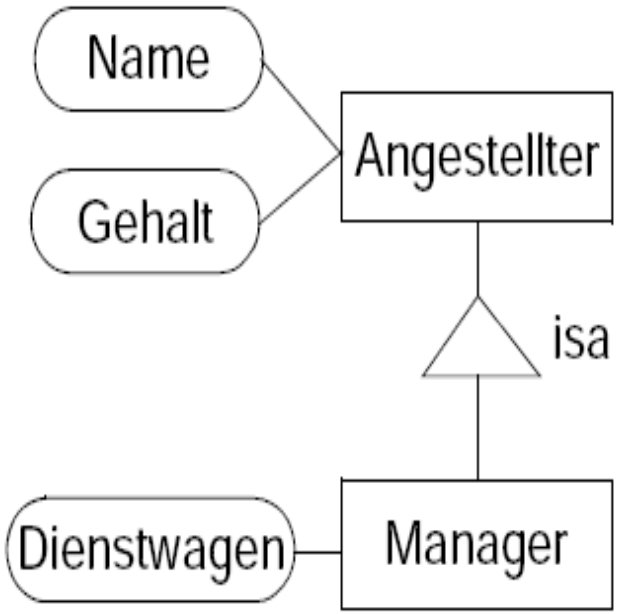


ERD-Beispiel: Projektdatenbank



ERM Erweiterung: Vererbung: IS-A Beziehung

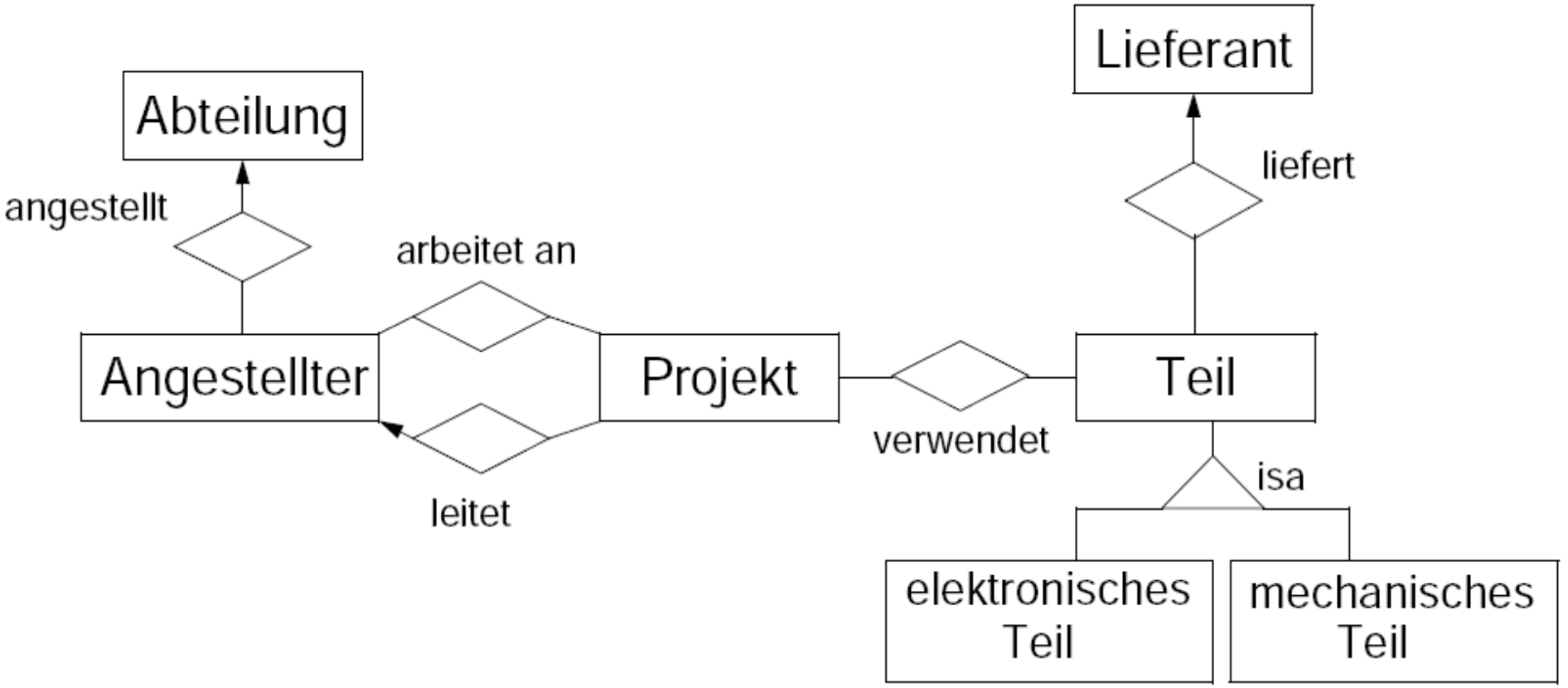
Beispiel:



Semantik dieser ISA-Beziehung:

Der Manager erbt alle Attribute und Beziehungen vom Angestellten. Der Manager erbt also auch den Schlüssel.

ERM Erweiterung: Vererbung: IS-A Beziehung



Modellierungsentscheidungen

- 1. Attribut oder Entity?
- 2. Attribut zu welchem Entity?
- 3. Schlüsselattribute?
- 4. Entity oder Relationship?
- 5. Relationships nur zwischen zwei Entities?
- 6. Attribut oder Relationship?
- 7. Einsatz von ISA-Beziehungen?

Modellierungsentscheidungen

1. Attribut oder Entity?

Kriterien:

- Ist das Attribut ein atomarer Wert oder aus mehreren Werten zusammengesetzt?
- Kann der Wert des Attributs fehlen?
- Kann das Attribut mehrere Werte gleichzeitig annehmen?
- Ist die Domäne des Attributs wichtig (bzgl. der Korrektheit des Wertes)?

Modellierungsentscheidungen

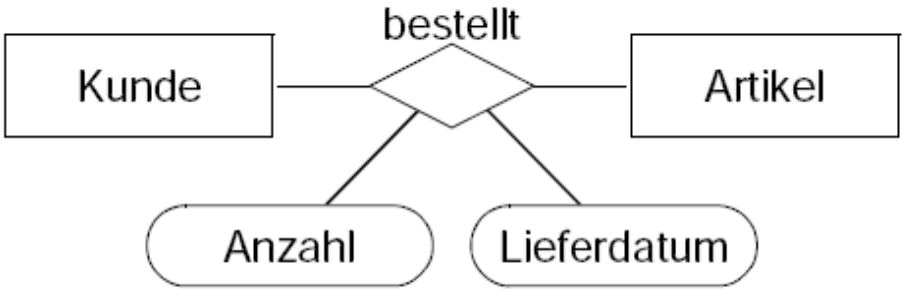
2. Attribut zu welchem Entity?

Kriterien:
- Natürlichkeit,
- Änderungsfreundlichkeit

3. Schlüssel-Attribute
- Dürfen auch mehrere sein (zusammengesetzter Schlüssel)
- Künstliche IDs (*Surrogate*) nur dann einsetzen, wenn es keinen konstanten Schlüssel innerhalb des Entity-Typs gibt oder der Schlüssel aus sehr vielen Attributen zusammengesetzt wäre

Modellierungsentscheidungen

4. Entity oder Relationship?



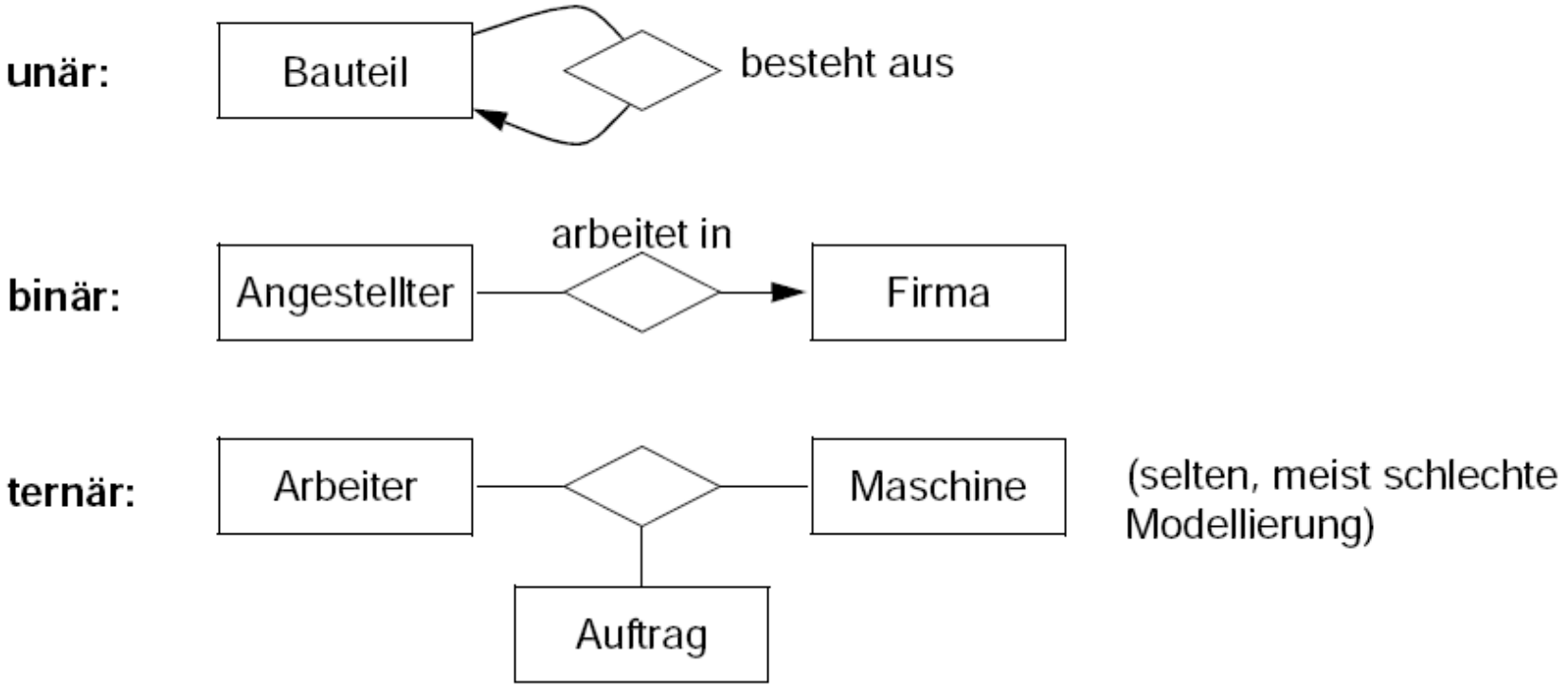
ODER

Kriterien:

- Beziehung benötigt Schlüsselattribute -> modelliere Entity
- Beziehung hat viele Attribute -> modelliere Entity

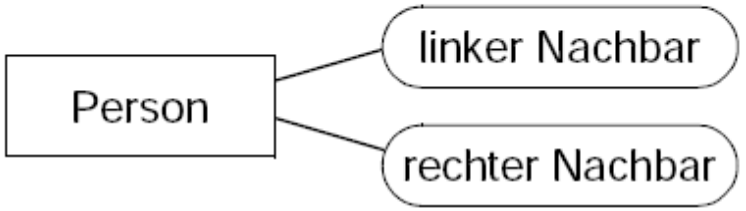
Modellierungsentscheidungen

5. Relationships nur zwischen zwei Entities?

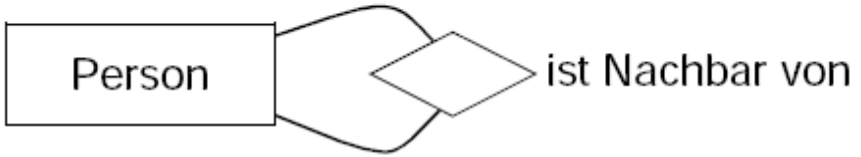


Modellierungsentscheidungen

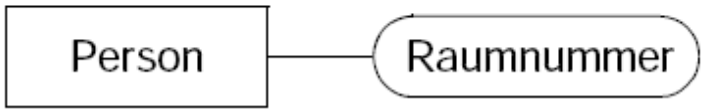
6. Attribut oder Relationship?



schlechte Modellierung im ER-Modell:
"Fremdschlüssel" vermeiden



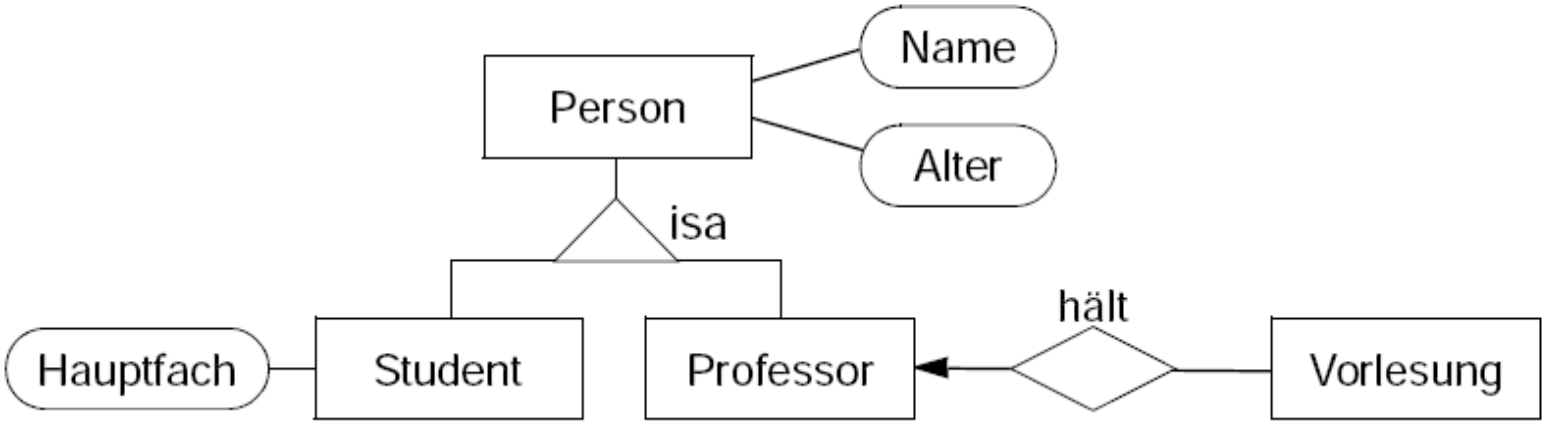
besser als Relationship modelliert



aus anderen Attributen berechenbar:
keine Relationship verwenden

Modellierungsentscheidungen

7. Einsatz von ISA-Beziehungen?



Kriterien:

- Hat die Oberklasse gemeinsame Beziehungen / Attribute
- Haben die Untersklassen unterschiedliche Beziehungen / Attribute
- Ist modellierte Information nützlich für die Applikation

Aufgaben: ERM

- Bibliothek
- Projektverwaltung
- Schulunterrichtsverwaltung
- Adventuregameverwaltung
- HTL-DA-Verwaltung
- Inventarverwaltung
-

Aufgaben: ERM

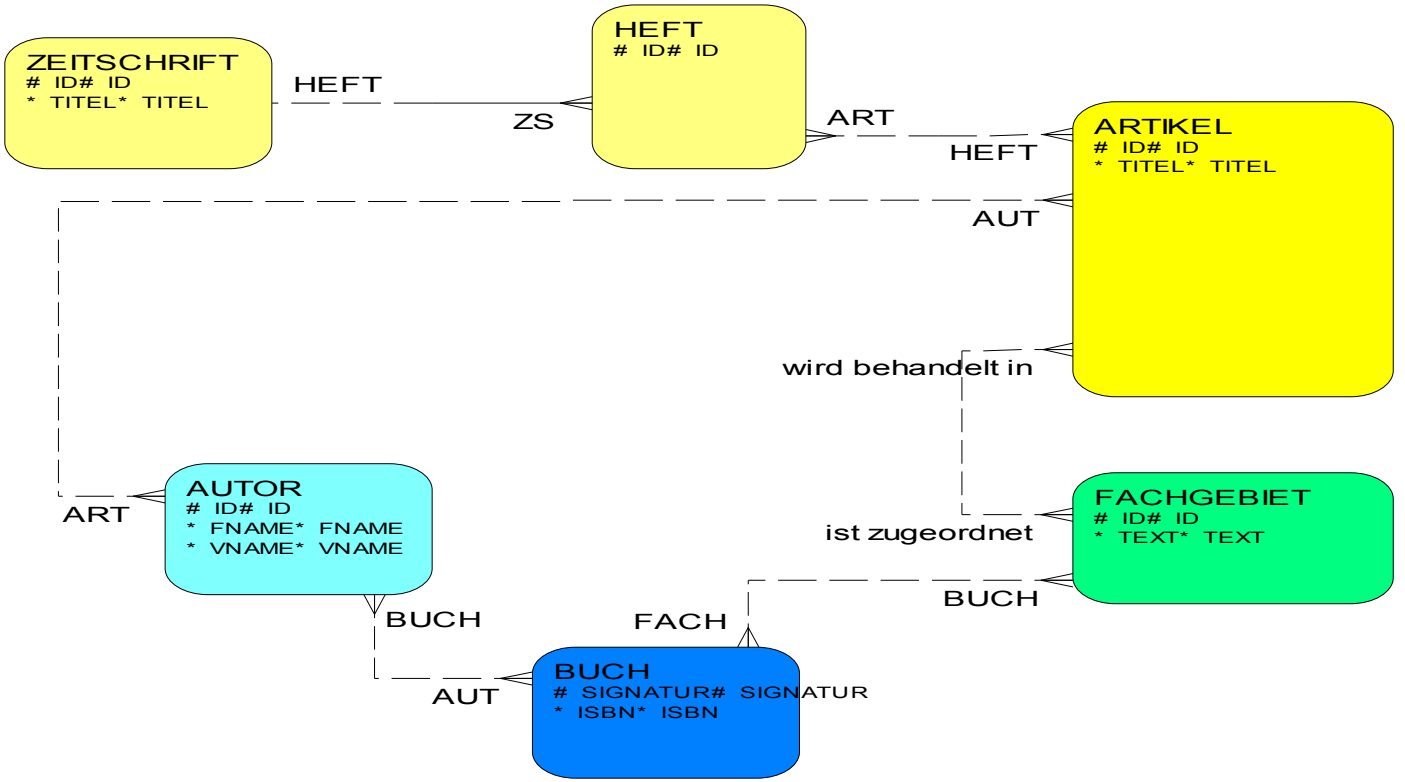
- Erstellen Sie ein ERD: (Projekt)
  - Mitarbeiter arbeiten an Projekten mit und sind gleichzeitig Abteilungen zugeordnet.
  - Es gibt Angestellte, die einen echten Dienstvertrag besitzen und andere, die einen freien Dienstnehmervertrag besitzen.
  - Die Angestellten haben bestimmte Qualifikationen, die bei der Projektarbeit notwendige Voraussetzung sind.

- Erstellen Sie ein ERD: (Autorenschaft)
  - Bücher werden von Autoren erstellt. Dabei ist die Reihenfolge der Autorenauflistung für die Entlohnung wichtig.

Aufgaben: ERM – Bibliothek

- Eine Bibliothek besteht aus Büchern und Zeitschriften.
  - Jedes Buch kann ggf. mehrere Autoren haben und ist eindeutig durch seine ISBN gekennzeichnet.
  - Die Bibliothek besitzt teilweise mehrere Exemplare eines Buches.
  - Zeitschriften dagegen sind jeweils nur einmal vorhanden. Sie erscheinen in einzelnen Heften und werden jahrgangsweise gebunden.
  - Die in Zeitschriften publizierten Artikel sind ebenso wie Bücher einem oder mehreren Fachgebieten (z.B. Betriebssysteme, Datenbanksysteme, Programmiersprachen) zugeordnet.
  - Ausgeliehen werden können nur Bücher (keine Zeitschriften).

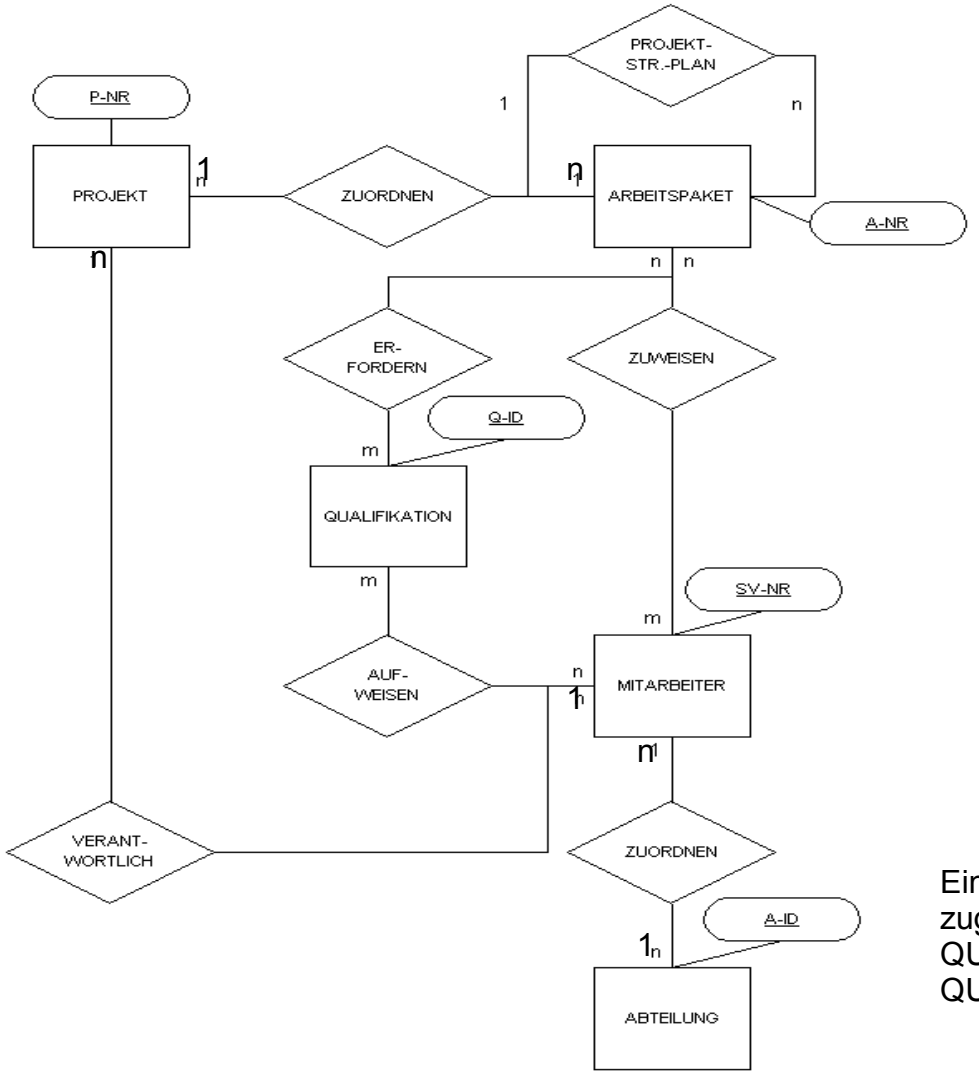
Lösung: ERM Bibliothek



Aufgabe: ERM - Projektmanagement

- **Stellen Sie folgende Zusammenhänge in einem Projektmanagementsystem als ER-Diagramm dar, und geben Sie an, ob es sich bei den Beziehungen jeweils um eine 1 : n- oder n : m-Beziehung handelt (ursprüngliche Entitytypen sind im Text bei ihrer ersten Nennung kursiv angegeben, in Klammern die Primärschlüssel).**
  - Ein *Projekt* (P-NR) besteht aus n *Arbeitspaketen* (A-NR), die jeweils einem Projekt eindeutig zugeordnet sind. Die Arbeitspakete sind untereinander zu einem hierarchischen Projektstrukturplan verbunden, wobei ein Arbeitspaket in mehrere weitere Arbeitspakete untergliedert werden kann, während jedes Arbeitspaket genau einem, hierarchisch übergeordneten Arbeitspaket untergeordnet ist.
  - Jedem Arbeitspaket werden ein oder mehrere *Mitarbeiter* (SV-NR) zugewiesen, wobei Mitarbeiter auch in mehreren Arbeitspaketen beschäftigt sein können. Mitarbeiter weisen bestimmte *Qualifikationen* (Q-ID) auf und werden *Abteilungen* (A-ID) zugewiesen.
  - Ein Arbeitspaket kann eine oder mehrere Qualifikationen erfordern.
  - Jedes Projekt hat einen Mitarbeiter als Verantwortlichen, wobei allerdings ein Mitarbeiter auch für mehrere Projekte Verantwortung tragen kann.
  - **Führen Sie obiges ER-Modell in eine relationale Tabellendefinition über, wobei Sie zu jeder Tabelle den Primärschlüssel angeben.**
  - **Kann ein Informationssystem, das auf diesem ER-Modell basiert, die Information liefern, welche Qualifikationen in Summe für die Abarbeitung eines Projektes**

Lösung: ERM Projektmanagement



PROJEKT

P-NR	SV-NR	...
...	...	...
...	...	...
...	...	...

ARBEITSPAKET

A-NR	ÜA-NR	P-NR	...
...	...	...	...
...	...	...	...
...	...	...	...

MITARBEITER

SV-NR	A-ID	...
...	...	...
...	...	...
...	...	...

ZUWEISEN

A-NR	SV-NR	...
...	...	...
...	...	...
...	...	...

ABTEILUNG

A-ID	...
...	...
...	...
...	...

QUALIFIKATION

Q-ID	...
...	...
...	...
...	...

ERFORDERN

A-NR	Q-ID	...
...	...	...
...	...	...
...	...	...

AUFWEISEN

SV-NR	Q-ID	...
...	...	...
...	...	...
...	...	...

Ein ARBEITSPAKET ist eindeutig einem PROJEKT zugeordnet.Einem ARBEITSPAKET sind entsprechende QUALIFIKATIONEN zugeordnet.Die Zuordnung PROJEKT - QUALIFIKATIONEN ist möglich.1 n

Relationales Datenmodell

- Einziger Grundbaustein: Relation (Tabelle)
  - wird beschrieben durch Relationenschema: $R(A_1 : D_1, \ldots, A_k : D_k)$
  - Attribut A_i: Spalte, Name eindeutig
  - Domäne D_i: Wertebereich, Datentyp des Attributs
  - Tupel: Zeile, Element einer Relation
- Relationales Datenbankschema: Menge aller Relationenschemata
- Relationale Datenbank: Menge aller Relationen = Relationales DB-Schema + Werte
- Relationen sind Mengen
  - keine Duplikate
  - Reihenfolge der Tupel belanglos

RM - Grundbegriffe

FAK

InstNr	InstName	Vorstand
123	Informatik	1000
456	Mathematik	2000

PROF

PersNr	PersName	InstNr	Gebiet
1234	Clausen	123	Vert. Systeme
4545	Hagenauer	123	Simulation
3535	Bauer	456	Lin.Algebra
2125	Zinterhof	456	Numerik

- Begriffe:
  - Tabelle(Relation): zB.: FAK, PROF,
  - Spalten(Attribute): zB.: InstNr, InstName, Gebiet,
  - Attributwerte(Domain): zB.: Clausen, Informatik,
  - Datensatz(Tupel): zB.: 123,Informatik,1000,
  - Primärschlüssel(PrimaryKey): zB.: FAK.InstNr: 123
  - Fremdschlüssel(ForeignKey): zB.: PROF.*InstNR*: 123

RM – Grundregeln

- Jede Zeile (Tupel) ist **eindeutig** und beschreibt ein Objekt (Entity) der Miniwelt

- Die Ordnung der Zeilen ist ohne Bedeutung
- Die Ordnung der Spalten ist ohne Bedeutung

- Jeder Datenwert in einer Spalte ist ein atomares Datenelement
- Alle Informationen sind ausschließlich durch Datenwerte ausgedrückt.

RM - Grundbegriffe

- Darstellung "tabellenübergreifender" Information durch Fremdschlüssel

  - **Foreign Key**=
    Attribut(menge), die in **Bezug auf den Primärschlüssel** einer anderen (oder derselben) Tabelle definiert ist.

    (gleicher Definitionsbereich) (*PROF.InstNr* == INST.InstNr)

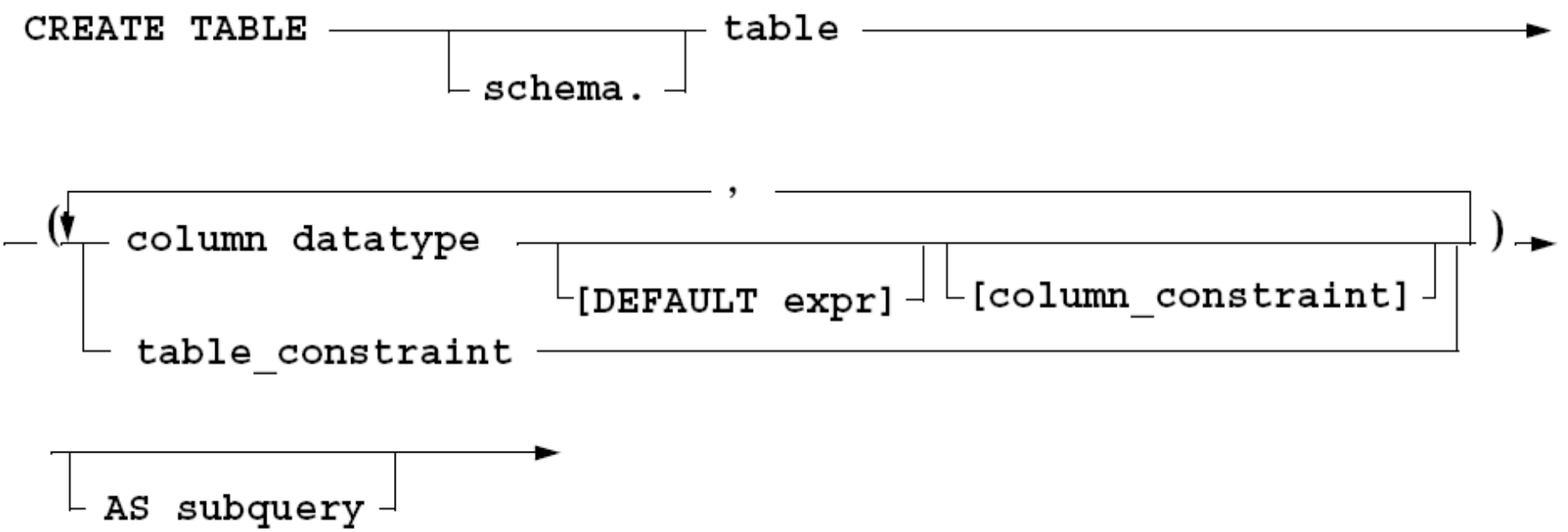
- Beziehungen werden durch

  Fremdschlüssel und zugehörigen **Primärschlüssel**

  dargestellt!

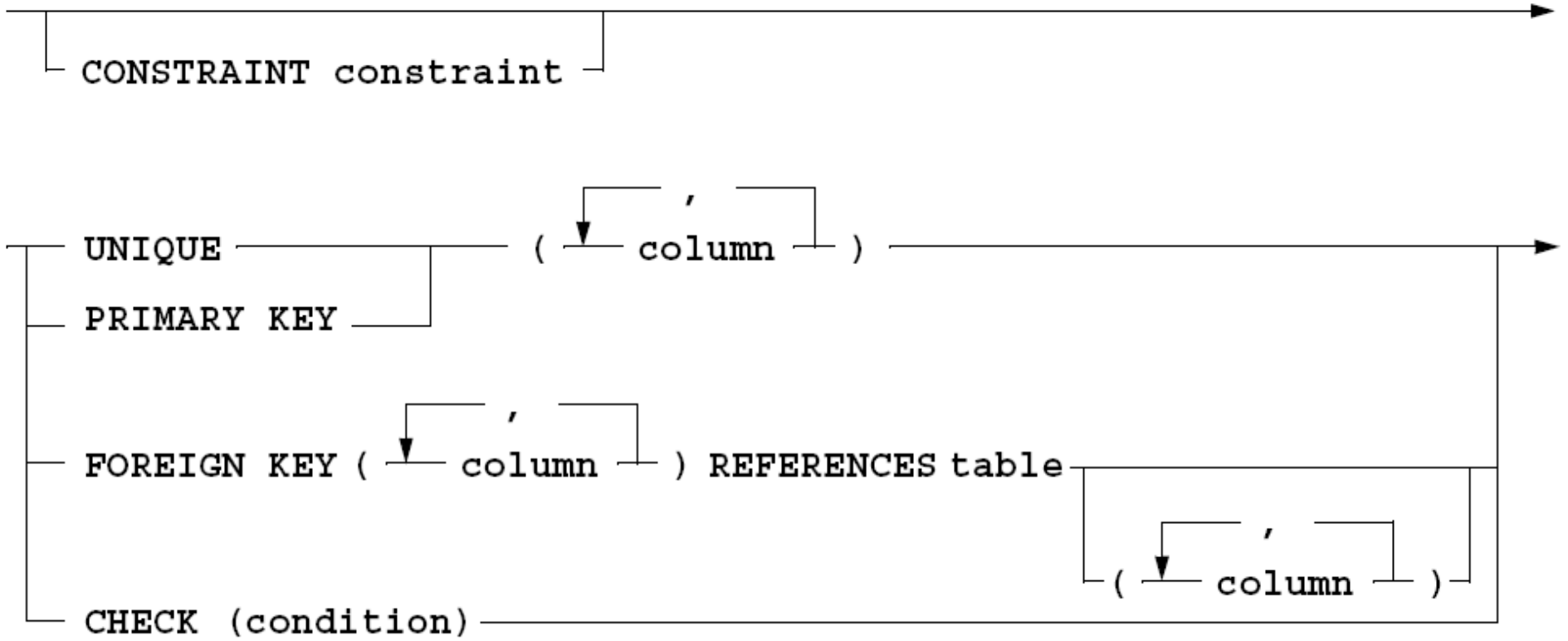
Datendefinition in SQL

Der Befehl create table (teilweise) in Oracle SQL



Datendefinition in SQL

table_constraint ::=



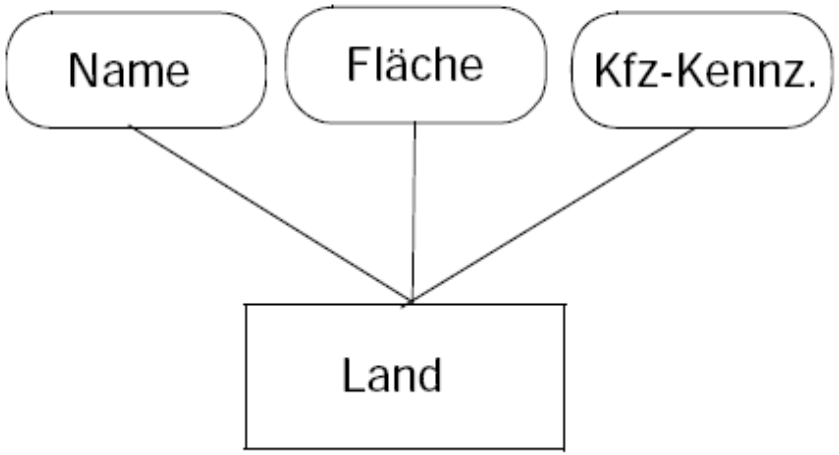
Oracle Datentypen

... auszugsweise ...

Datatype	Description
CHAR(*size*)	Fixed length character date of length *size*. Maximum *size* is 255.
VARCHAR2(*size*)	Variable length character string having maximum length *size* bytes. Maximum size is 4000.
NUMBER(*p*, *s*)	Number having precision *p* and scale *s*.
LONG	Character data of variable length up to 2 gigabytes
DATE	Valid dates range from January 1, 4712 BC to December 31, 4712 AD
RAW(*size*)	Raw binary data of length *size* bytes. Maximum *size* is 2000.
LONG RAW	Raw binary data of variable length up to 2 gigabytes.
BLOB	A binary large object. Maximum size is 4 gigabytes.

Transformation: ERM -> RM

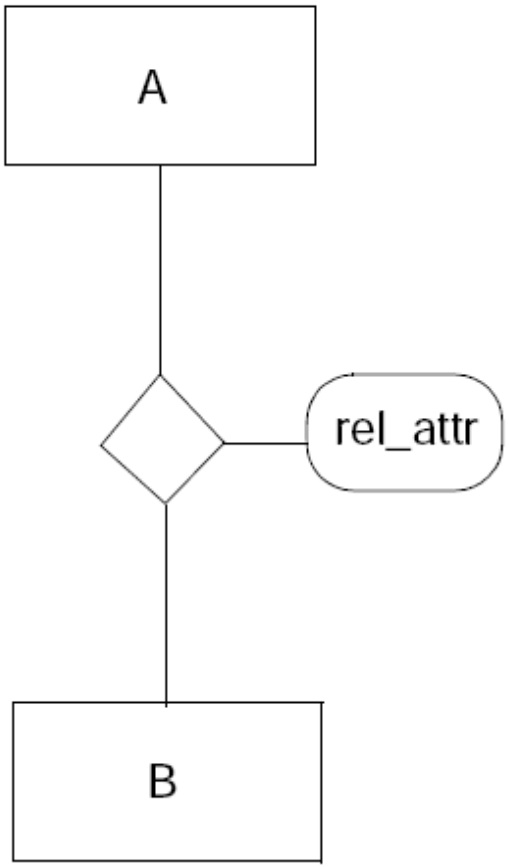
- Transformation von Entities




```
CREATE TABLE land
(
    name varchar2(25) NOT NULL,
    flaeche number(10,2),
    kfz_kennz char(4) NOT NULL,

    CONSTRAINT pk_name
       PRIMARY KEY (name),
    UNIQUE (kfz_kennz)
);
```

Transformation: ERM -> RM

- Transformation von Relationships (n:m)



Entity A und B haben wir schon überführt:

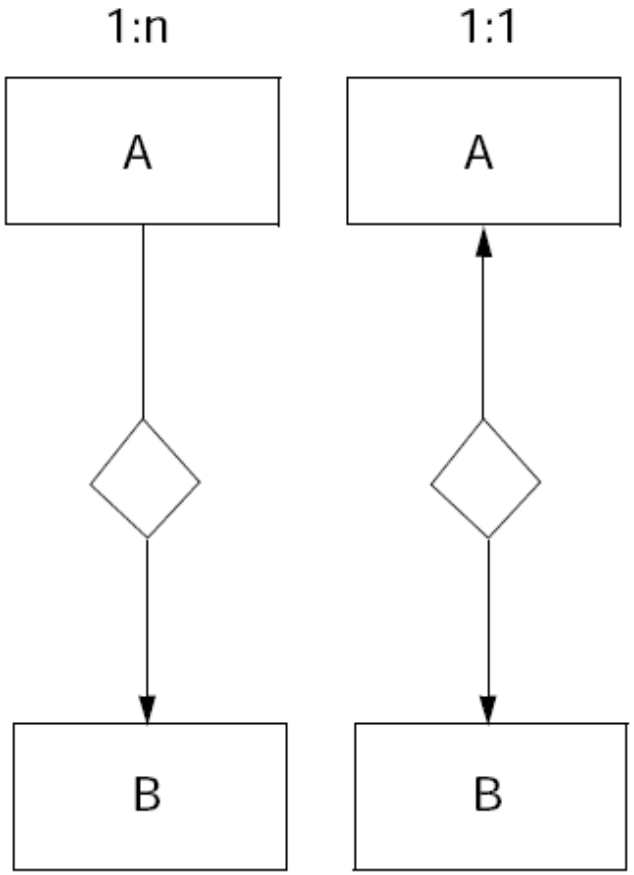
```
CREATE TABLE A ( ... PRIMARY KEY a_id ...)

CREATE TABLE B ( ... PRIMARY KEY b_id ...)
```

Hinzu kommt:

```
CREATE TABLE rel
(
    a_id        ... NOT NULL,
    b_id        ... NOT NULL,
    rel_attr    ... ,
    PRIMARY KEY (a_id, b_id),
    FOREIGN KEY (a_id) REFERENCES A,
    FOREIGN KEY (b_id) REFERENCES B
)
```

Transformation: ERM -> RM

- Transformation von Relationships (1:n, 1:1)



Keine separate Relation für die Beziehung, sondern:

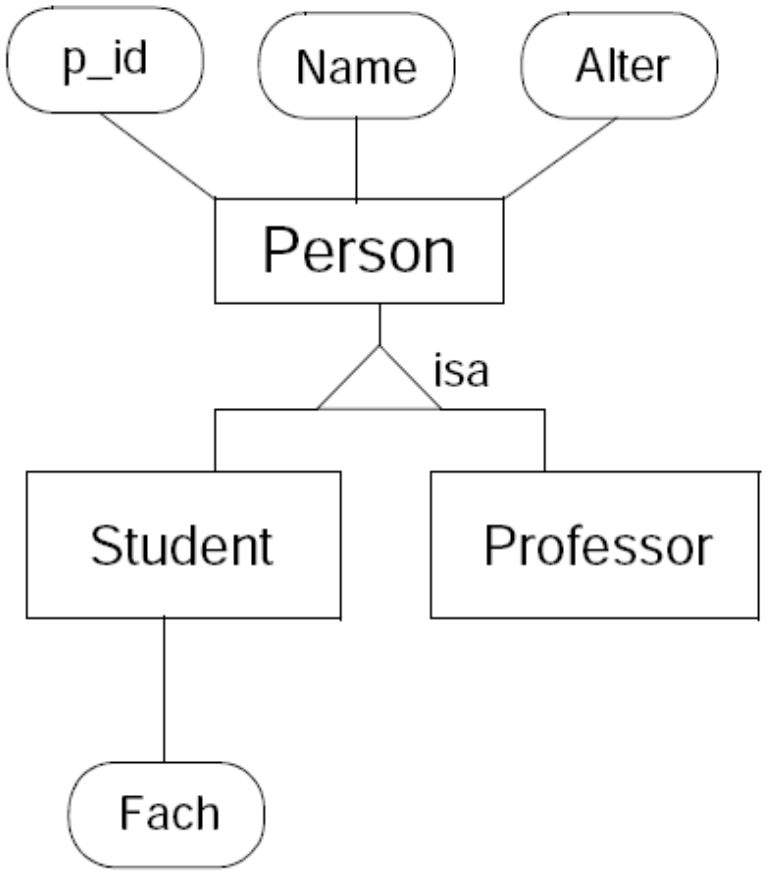
1:n -> Aufnahme des Primärschlüssels von B in die Entity-Relation A

1:1 -> Aufnahme des Primärschlüssels von B in die Entity-Relation A oder umgekehrt

```
CREATE TABLE A
(
    a_id          ... NOT NULL,
    a_rest        ... ,
    b_id          ... NOT NULL,
    PRIMARY KEY (a_id),
    FOREIGN KEY (b_id) REFERENCES B
)
```

Transformation: ERM -> RM

- Transformation von ISA-Beziehungen




```
CREATE TABLE person
(
    p_id char(8) NOT NULL,
    name        ... ,
    alter       ... ,
    PRIMARY KEY (p_id),
)

CREATE TABLE student
(
    p_id char(8) NOT NULL,
    fach        ... ,
    PRIMARY KEY (p_id),
    FOREIGN KEY (p_id) REFERENCES person
)

CREATE TABLE professor
(
    p_id char(8) NOT NULL,
    ... ,
    PRIMARY KEY (p_id),
    FOREIGN KEY (p_id) REFERENCES person
)
```

Transformation: ERM -> RM

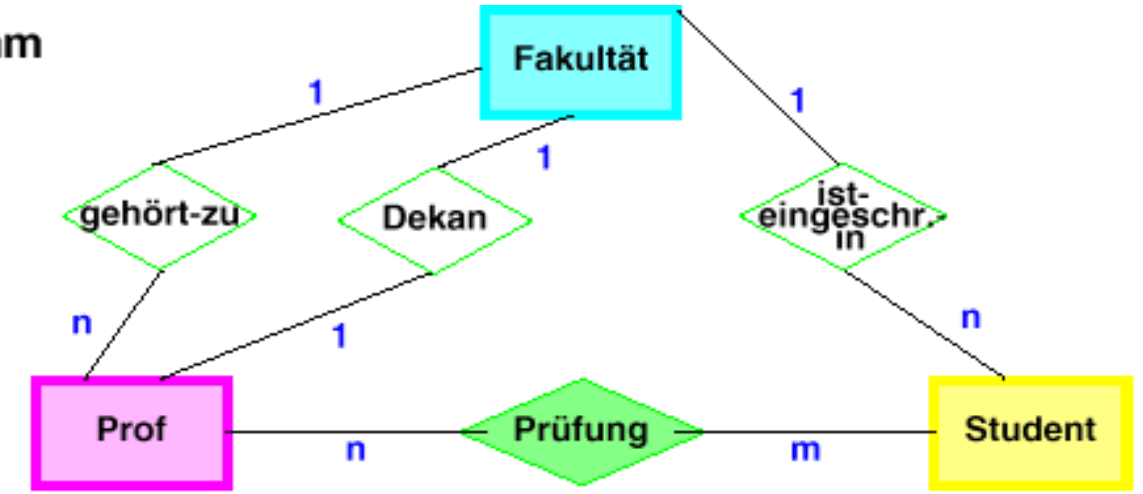
- Alternative:

```
CREATE TABLE person_x
(
    p_id char(8) NOT NULL,
    name         ... ,
    alter        ... ,
    PRIMARY KEY (p_id)
)

CREATE TABLE student
(
    p_id char(8) NOT NULL,
    name         ... ,
    alter        ... ,
    fach         ... ,
    PRIMARY KEY (p_id)
)

CREATE TABLE professor
( ... )
```

```
CREATE VIEW person (p_id, name, alter) AS
(
     (SELECT p_id, name, alter FROM peron_x)
     UNION
     (SELECT p_id, name, alter FROM student)
     UNION
     (SELECT p_id, name, alter FROM professor)
)
```

Zusammenfassung: ERD --> RM

ER-Diagramm



Relationales Schema

FAK

FNR	FNAME	DEKAN

STUDENT

MATNR	SNAME	FNR	STUDBEG

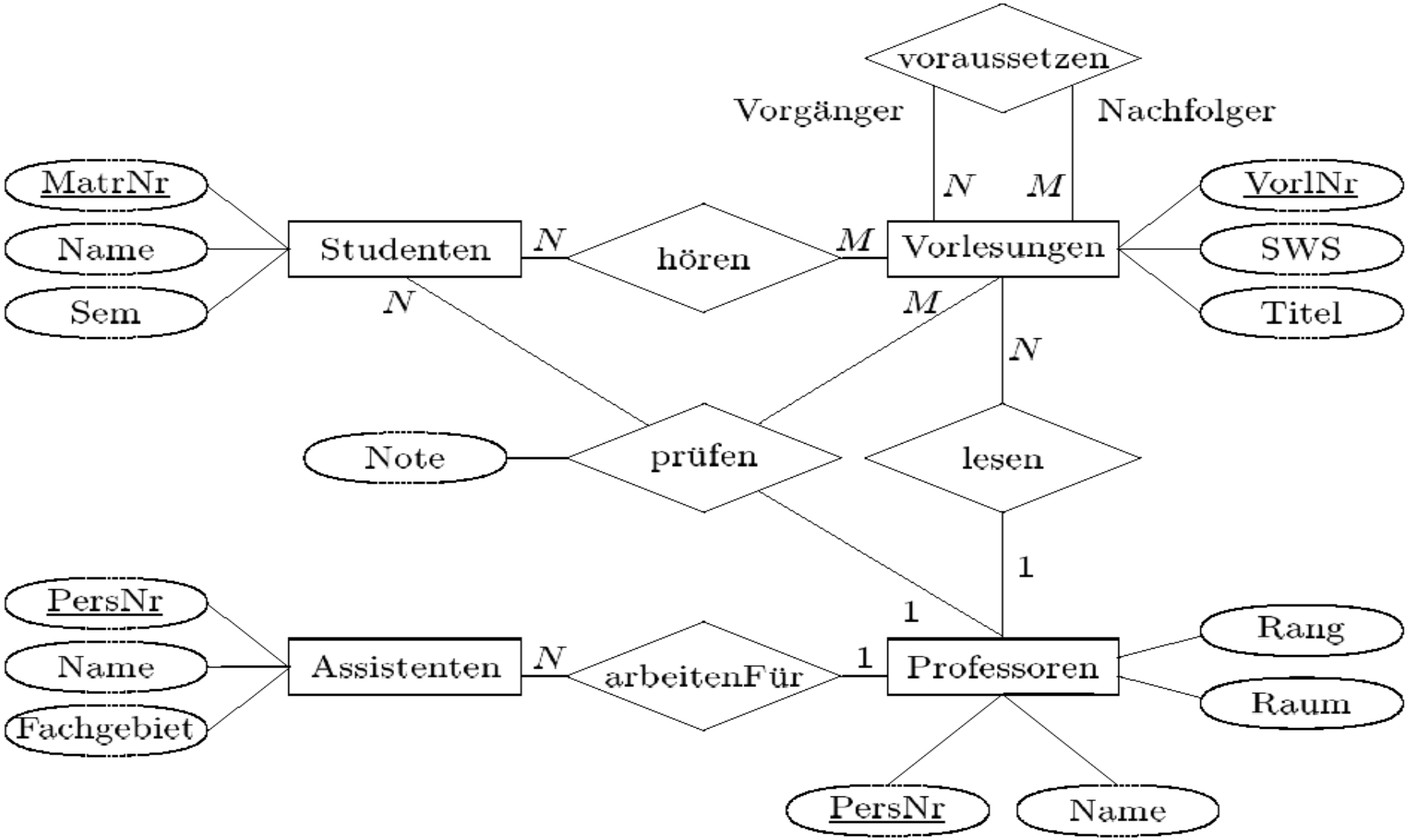
PROF

PNR	PNAME	FNR	FACHGEB

PRUEFUNG

PNR	MATNR	FACH	DATUM	NOTE

Beispiel: Uni-Beispiel mit Kardinalitäten



Uni-Beispiel: Tabellen

- Studenten :
  - {[MatrNr : integer; Name : string; Semester : integer] }
- Vorlesungen :
  - { [VorlNr : integer; Titel : string; SWS : integer] }
- Professoren :
  - { [PersNr : integer; Name : string; Rang : string; Raum : integer] }
- Assistenten :
  - { [PersNr : integer; Name : string; Fachgebiet : string] }

Uni-Beispiel: Beziehungen

- Beziehung: hören : (Studenten N:M Vorlesungen)
  - Tabelle: [*MatrNr* : integer; *VorlNr* : integer] }

- Beziehung: lesen (Professoren 1:N Vorlesungen.)
  - KEINE Tabelle: sondern Primar-Fremdschlüssel Constraint (s.u)

- Beziehung: arbeitenFür : (Professoren 1:N Assistenten)
  - KEINE Tabelle: sondern Primar-Fremdschlüssel Constraint (s.u)

- Beziehung: voraussetzen : (Vorlesungen N:M Vorlesungen)
  - Tabelle: {[*Vorgänger* : integer; *Nachfolger* : integer] }

- Beziehung: prüfen : (Studenten, Vorlesungen, Professoren)
  - Tabelle: { [*MatrNr* : integer; *VorlNr* : integer; *PersNr* : integer; Note : decimal] }

Uni-Beispiel: *hören* (N:M Beziehungen)

Studenten	
MatrNr	...
26120	...
27550	...
...	...

hören	
MatrNr	VorlNr
26120	5001
27550	5001
27550	4052
28106	5041
28106	5052
28106	5216
28106	5259
29120	5001
29120	5041
29120	5049
29555	5022
25403	5022
29555	5001

Vorlesungen	
VorlNr	...
5001	...
4052	...
...	...

Uni-Beispiel: Professoren 1:N Vorlesungen

- Vorlesungen : { [VorlNr; Titel; SWS; ***gelesenVon***] }Professoren : { [PersNr; Name; Rang; Raum] }

Professoren			
PersNr	Name	Rang	Raum
2125	Sokrates	C4	226
2126	Russel	C4	232
2127	Kopernikus	C3	310
2133	Popper	C3	52
2134	Augustinus	C3	309
2136	Curie	C4	36
2137	Kant	C4	7

Vorlesungen			
VorlNr	Titel	SWS	gelesenVon
5001	Grundzüge	4	2137
5041	Ethik	4	2125
5043	Erkenntnistheorie	3	2126
5049	Mäeutik	2	2125
4052	Logik	4	2125
5052	Wissenschaftstheorie	3	2126
5216	Bioethik	2	2126
5259	Der Wiener Kreis	2	2133
5022	Glaube und Wissen	2	2134
4630	Die 3 Kritiken	4	2137

Uni-Beispiel: Relationale Darstellung

Professoren			
PersNr	Name	Rang	Raum
2125	Sokrates	C4	226
2126	Russel	C4	232
2127	Kopernikus	C3	310
2133	Popper	C3	52
2134	Augustinus	C3	309
2136	Curie	C4	36
2137	Kant	C4	7

Studenten		
MatrNr	Name	Semester
24002	Xenokrates	18
25403	Jonas	12
26120	Fichte	10
26830	Aristoxenos	8
27550	Schopenhauer	6
28106	Carnap	3
29120	Theophrastos	2
29555	Feuerbach	2

Vorlesungen			
VorlNr	Titel	SWS	gelesenVon
5001	Grundzüge	4	2137
5041	Ethik	4	2125
5043	Erkenntnistheorie	3	2126
5049	Mäeutik	2	2125
4052	Logik	4	2125
5052	Wissenschaftstheorie	3	2126
5216	Bioethik	2	2126
5259	Der Wiener Kreis	2	2133
5022	Glaube und Wissen	2	2134
4630	Die 3 Kritiken	4	2137

voraussetzen	
Vorgänger	Nachfolger
5001	5041
5001	5043
5001	5049
5041	5216
5043	5052
5041	5052
5052	5259

hören	
MatrNr	VorlNr
26120	5001
27550	5001
27550	4052
28106	5041
28106	5052
28106	5216
28106	5259
29120	5001
29120	5041
29120	5049
29555	5022
25403	5022
29555	5001

Assistenten			
PersNr	Name	Fachgebiet	Boss
3002	Platon	Ideenlehre	2125
3003	Aristoteles	Syllogistik	2125
3004	Wittgenstein	Sprachtheorie	2126
3005	Rhetikus	Planetenbewegung	2127
3006	Newton	Keplersche Gesetze	2127
3007	Spinoza	Gott und Natur	2134

prüfen			
MatrNr	VorlNr	PersNr	Note
28106	5001	2126	1
25403	5041	2125	2
27550	4630	2137	2

Aufgaben: ERM -> RM

- Reverse Engineering:
- Lade in die Datenbank
  http://dev.mysql.com/doc/world-setup/en/world-setup.html

- Verwende ein ERM- Programm und starte Reverse Engineering

Data Dictionary

Neben den Daten **in** den Relationen benötigt ein Datenbanksystem Informationen **über** die Relationen (und über andere Objekte), sog. "Meta-Daten".

Die Meta-Daten werden im "Data Dictionary" oder "System Katalog" gespeichert.

Relationenverwaltung

- Relationennamen
- Attributsnamen für jede Relation
- Domäne der Attribute
- Viewnamen und Viewdefinitionen
- Integritätsbedingungen

Benutzerverwaltung (Datenschutz)

- Namen berechtigter (autorisierter) Benutzer
- Speicherbereich und Speicherobergrenze für jeden Benutzer
- Zugriffs- und andere Rechte einzelner Benutzer

Data Dictionary

Namensgebung für Oracle Systemrelationen

	USER_	ALL_	DBA_
TABLES	alle Tabellen, die der Benutzer angelegt hat	alle Tabellen, auf die der Benutzer Zugriff hat	alle Tabellen des gesamten Systems
TAB_ COLUMNS	alle Spalten derjenigen Tabellen, die der Benutzer angelegt hat	alle Spalten derjenigen Tabellen, auf die der Benutzer Zugriff hat	alle Spalten aller Tabellen des gesamten Systems
INDEXES	alle Indexe, die der Benutzer angelegt hat	alle Indexe, die über Tabellen erstellt wurden, auf die der Benutzer Zugriff hat	alle Indexe des gesamten Systems
VIEWS	alle Views, die der Benutzer angelegt hat	alle Views, auf die der Benutzer Zugriff hat	alle Views des gesamten Systems
TAB_ PRIVS	Zugriffsrechte auf alle Tabellen, die der Benutzer angelegt hat	Zugriffsrechte auf alle Tabellen, auf die der Benutzer Zugriff hat	Zugriffsrechte auf alle Tabellen des gesamten Systems