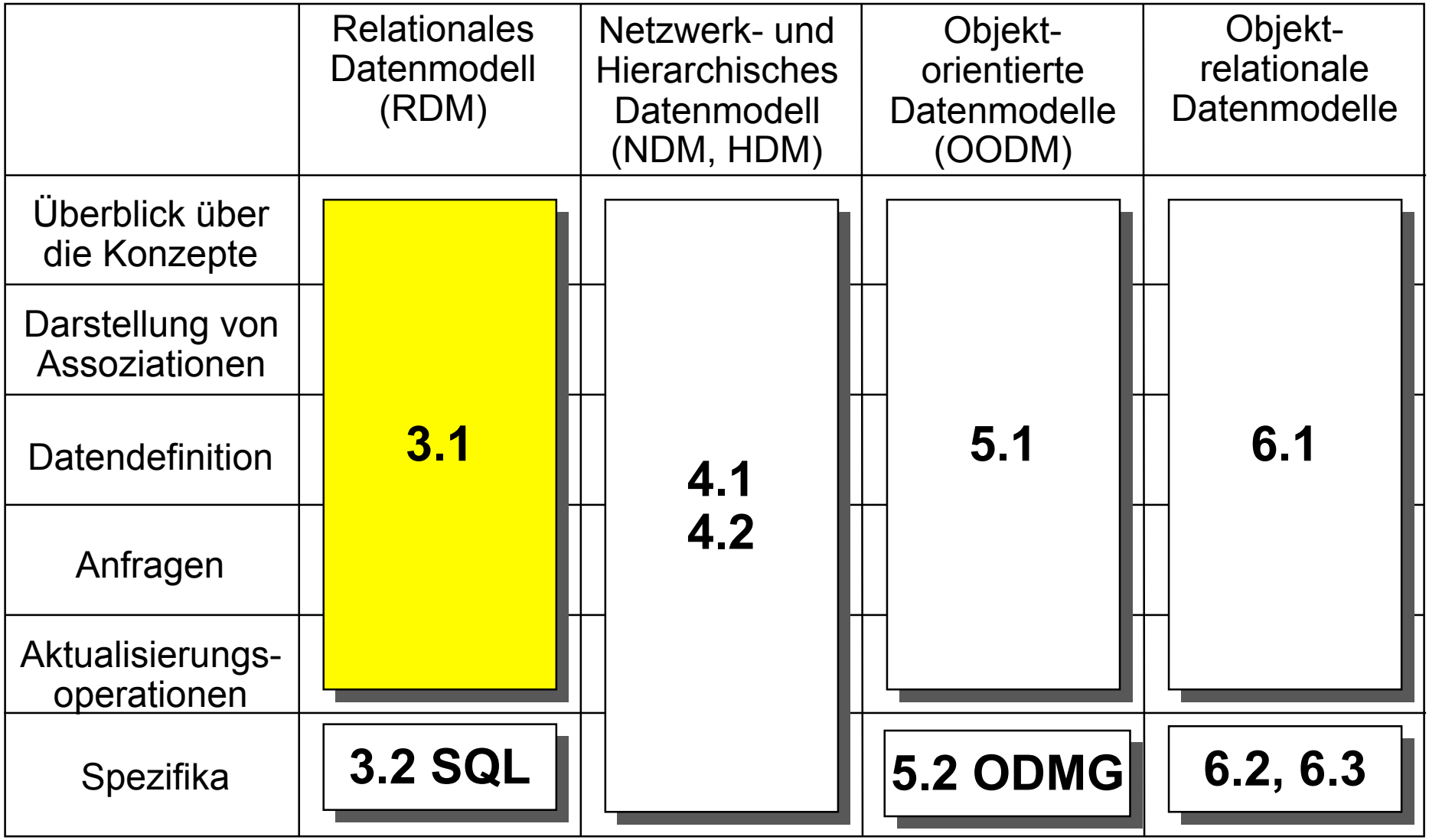
Kapitel 3: Das relationale DB-Modell & SQL



3.1 RDM: Überblick über die Konzepte (1)

- Eine Datenbank ist ein Aggregat benannter *Relationen*.
- Eine Relation ist eine Menge von Elementen
  - deren Struktur durch Attribute definiert,
  - deren Identität durch Schlüssel realisiert und
  - deren Werte durch Domänen kontrolliert werden.
- Ein Relationenelement ist das Aggregat seiner Attribute.
- Relationen werden meist durch *Tabellen* dargestellt, wobei jede Tabelle aus Zeilen und Spalten besteht.
- Jede Zeile repräsentiert ein Element der Relation und wird auch als *Tupel* bezeichnet.
- Die Zahl der Zeilen ist variabel und wird *Kardinalität* der Relation genannt.
- Die Spalten der Tabellen enthalten die *Attribute* der Relation.

RDM: Überblick über die Konzepte (2)

- Die Zahl der Spalten einer Tabelle wird im Schema festgelegt und als *Grad* der Relation bezeichnet.
- Jeder Spalte ist eine *Domäne* zugeordnet, welche die zulässigen Werte für das Attribut in allen Zeilen festlegt.
- Jede Tabelle besitzt einen *Primärschlüssel*, der ein einzelnes Attribut oder eine Kombination von Attributen ist, die eine eindeutige Identifikation jedes Tupels innerhalb der Tabelle gestattet.
- Beziehungen zwischen Datenobjekten werden durch Identifikation des referenzierten Objektes über seinen Primärschlüssel repräsentiert (➠ assoziative Identifikation).
- Einen Schlüssel, der in Relation A zur Identifikation eines Tupels in Relation B benutzt wird, bezeichnet man als *Fremdschlüssel*.

RDM: Projektdatenbank

Projekte

Nr	*Titel*	*Budget*
100	DB Fahrpläne	300.000
200	ADAC Kundenstamm	100.000
300	Telekom Statistik	200.000

Projektdurchführung

Nr	*Kurz*
100	MFSW
100	UXSW
100	LTSW
200	UXSW
200	PERS
300	MFSW

Abteilungen

Kurz	*Name*	*Oberabt*
MFSW	Mainframe SW	LTSW
UXSW	Unix SW	LTSW
PCSW	PC SW	LTSW
LTSW	Leitung SW	**NULL**
PERS	Personal	**NULL**

Projektdatenbank

RDM: Tabellen und Schlüssel (1)

Im relationalen Datenmodell unterscheidet man zwei Arten von Schlüsseln:

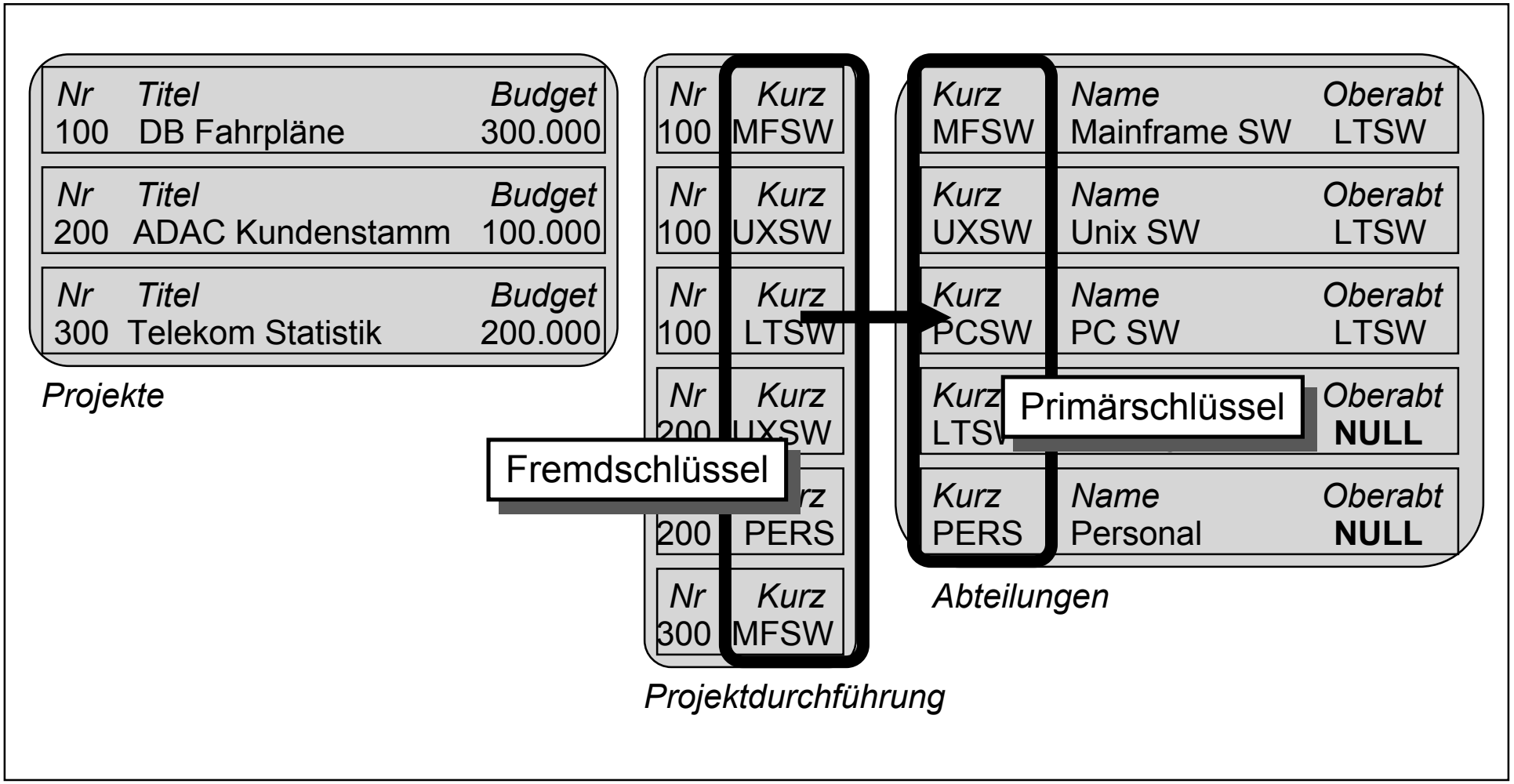
- **Primärschlüssel**: Ist ein Attribut oder eine minimale Attributkombination, das/die für jede Zeile der Tabelle eindeutige Werte bzw. Wertkombinationen enthält. Es kann mehrere Attribute mit dieser Eigenschaft geben (*Schlüsselkandidaten*).
- **Fremdschlüssel**: Ist ein Attribut oder Attributkombination, das/die als Werte Primärschlüsselwerte anderer Tabellen annimmt. Er ist eine Referenz auf ein anderes Objekt.

Duplikate sind in Tabellen nicht erlaubt ➠ die Gesamtheit aller Attribute bildet automatisch einen Schlüsselkandidaten. Oft ist jedoch die Einführung eines künstlichen Schlüssels z.B. einer eindeutigen Nummer (ID) sinnvoll.

Eine Relation mit Primärschlüssel repräsentiert eine Funktion von den Primärschlüsselattributen zu den Nicht-Schlüsselattributen.

Beispiel: Kurz ➠ (Name, Oberabt), Kurz ➠ Name, Kurz ➠ Oberabt

RDM: Tabellen und Schlüssel (2)



Projektdatenbank

RDM: Assoziationen

Bei der Benutzung des Relationenmodells unterscheidet man meist zwei Arten der Verwendung von Tabellen:

- **Entitätentabellen** stellen Objekte (Entitäten) der Anwendung dar (z.B. Projekte, Abteilungen). Jede Zeile einer solchen Tabelle beschreibt ein Objekt.
- **Beziehungstabellen** werden unter Verwendung von Fremdschlüsseln zur Darstellung von Assoziationen zwischen Objekten verwendet (z.B. Projektdurchführung).

1:1 und 1:n Beziehungen lassen sich durch Aufnahme eines Fremdschlüsselattributs in eine Entitätentabelle repräsentieren (z.B. Attribut Oberabteilung in Abteilungen). Falls keine assoziierte Entität existiert, muß der ausgezeichnete Nullwert **NULL** als Fremdschlüsselwert verwendet werden. **NULL** ist kein zulässiger Primärschlüsselwert.

Generell: Bei 1:1 und 1:n Beziehungen sind keine Beziehungstabellen notwendig.

Beachte: Im relationalen Modell existieren zur Verknüpfung von Objekten keine Zeiger oder vergleichbare Strukturen. Das relationale Modell basiert auf assoziativer Identifikation mit Hilfe von Werten.

RDM: Datendefinition

Schemadefinition der Projektdatenbank:

```
create table Projekte
( Nr integer not null,
  Titel char(30) not null,
  Budget decimal(10,2) not null,
  primary key(Nr) );
```

```
create table Projektdurchfuehrung
( Nr integer not null,
  Kurz char(4) not null,
  primary key(Nr, Kurz) );
```

```
create table Abteilungen
( Kurz char(4) not null,
  Name char(30) not null,
  Oberabt char(4),
  primary key(Kurz) );
```

Referentielle Integrität in SQL: Kapitel 3.2

RDM: Referentielle Integrität

Referentielle Integrität: Zu jedem benutzten Fremdschlüssel existiert ein Tupel mit einem entsprechenden Primärschlüsselwert in der referenzierten Tabelle.

Überprüfung der referentiellen Integrität ist notwendig beim

- Einfügen eines neuen Fremdschlüsselwertes in eine Beziehungstabelle. Das referenzierte Objekt mit diesem Wert als Primärschlüssel muß existieren.
- Löschen eines Tupels aus einer Entitätentabelle. Auf dieses Tupel dürfen keine Referenzen bestehen. Gibt es noch Referenzen, bieten sich mehrere Möglichkeiten an:
  - Eine Fehlermeldung wird erzeugt.
  - Propagierung der Löschoperation, das referenzierende Tupel wird ebenfalls gelöscht (➠ *kaskadiertes Löschen*).
  - Die Referenzen können durch Setzen des Fremdschlüssels auf einen Nullwert ungültig gemacht werden, sofern dieser nicht Bestandteil des Schlüssels ist.

RDM: Domänen

- Domänen legen zulässige Wertebereiche für Attribute fest. Sie sind mit Typen vergleichbar und können
  - mit vordefinierten Typen übereinstimmen,
  - spezielle Wertmengen festlegen.

```
Int
```

```
"Yes", "No", "Don't know"
```

- Operationen auf Attributen, wie z.B. der Vergleich zwischen Budget und Nummer, können auf ihre Zulässigkeit überprüft werden.

RDM: Entwurf relationaler Schemata

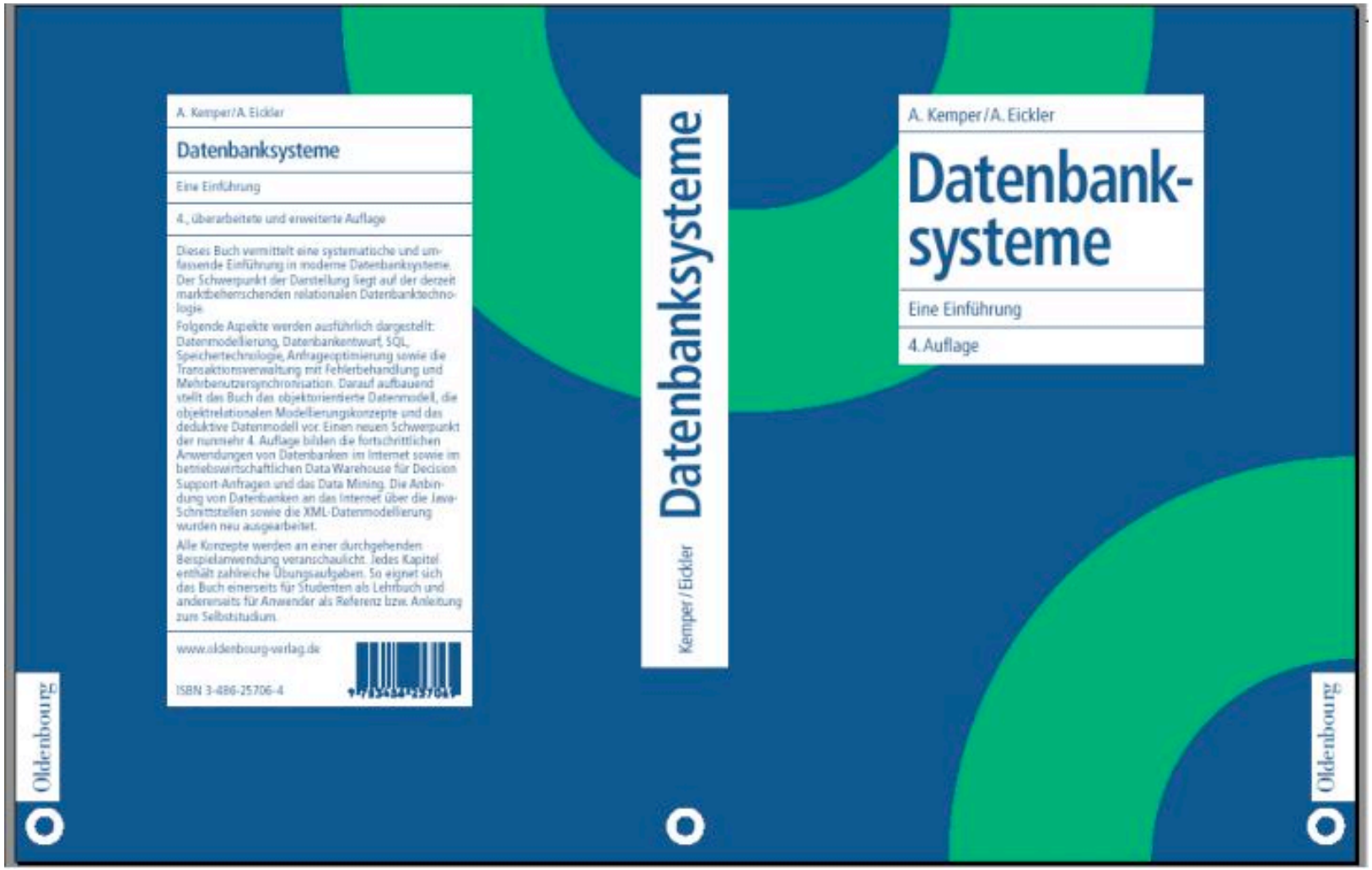
Zwei alternative Methoden:

- Entwickle zunächst ein ER-Diagramm, leite daraus ein relationales Schema mit Entitäten- und Beziehungstabellen ab
  (vgl. C. Batini, S. Ceri, S.B. Navathe. Conceptual Database Design - An Entity Relationship Approach, Benjamin/Cummings, Redwood City, Kalifornien, 1992).

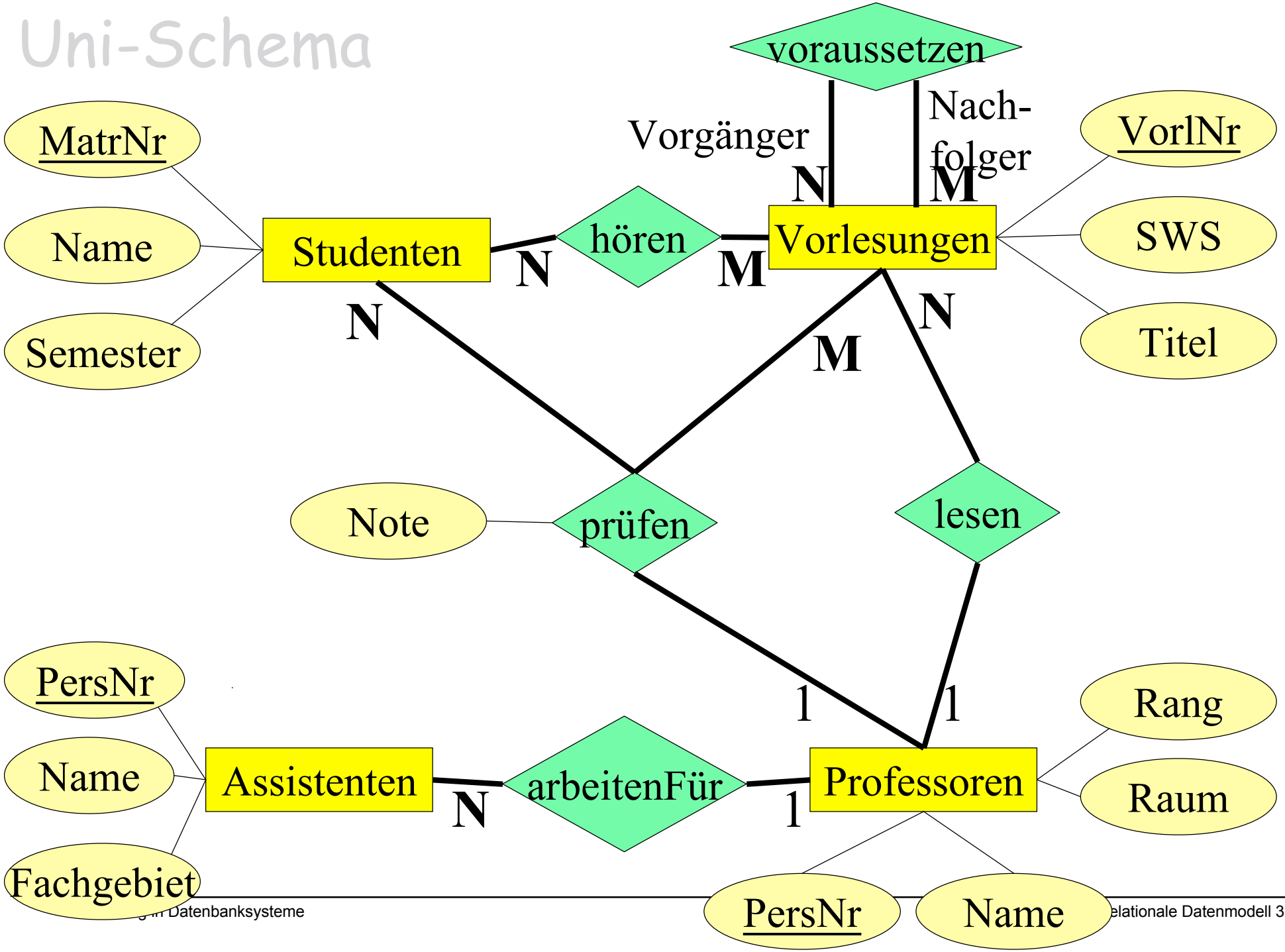
  bevorzugt

- Sammle funktionale Abhängigkeiten aus der Anforderungsdefinition und erzeuge daraus ein relationales Schema in Normalform
  (Im Trend 1970...80). Ausführlich in der Literatur beschrieben
  (vgl. S.M. Lang, P.C. Lockemann. Datenbankeinsatz. Springer, Berlin u.a., 1995).

Acknowledgments



Uni-Schema


voraussetzen
Vorgänger
Nach-
folger
N
M
MatrNr
Name
Semester
Studenten
N
hören
M
Vorlesungen
VorlNr
SWS
Titel
N
M
N
Note
prüfen
lesen
1
1
PersNr
Name
Fachgebiet
Assistenten
N
arbeitenFür
1
Professoren
Rang
Raum
PersNr
Name

Relationale Darstellung von Entitytypen

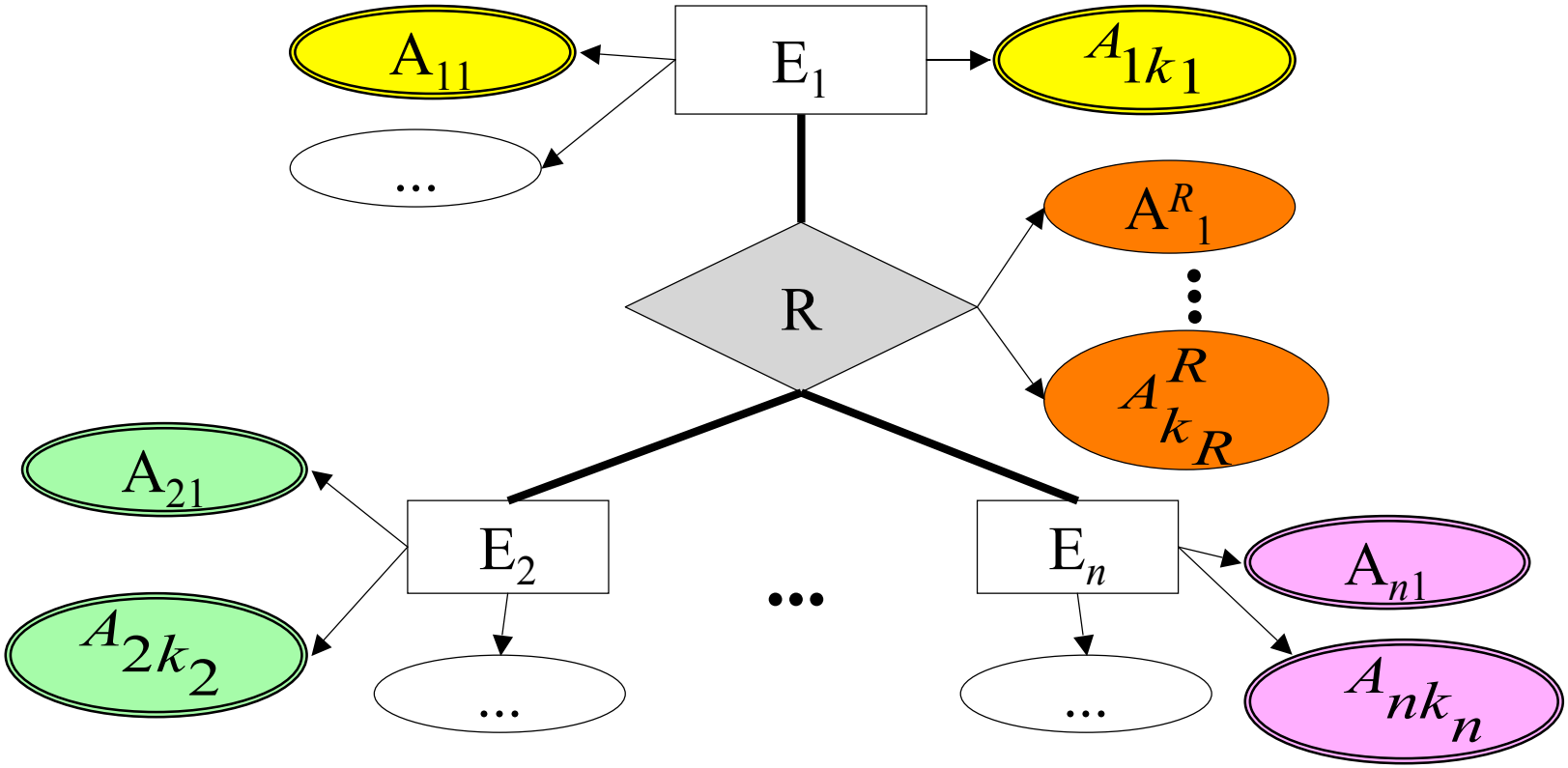
Studenten: {[MatrNr:integer, *Name*: *string, Semester*: *integer*]}

Vorlesungen: {[VorlNr:integer, *Titel*: *string, SWS*: *integer*]}

Professoren: {[PersNr:integer, *Name*: *string, Rang*: *string, Raum*: *integer*]}

Assistenten: {[PersNr:integer, *Name*: *string, Fachgebiet*: *string*]}

Relationale Darstellung von Beziehungen



$$R: \{[\underbrace{A_{11},\ldots,A_{1k_1}}_{\text{Schlüssel von } E_1}, \underbrace{A_{21},\ldots,A_{2k_2}}_{\text{Schlüssel von } E_2}, \ldots, \underbrace{A_{n1},\ldots,A_{nk_n}}_{\text{Schlüssel von } E_n}, \underbrace{A_1^R,\ldots,A_{k_R}^R}_{\text{Attribute von R}}]\}$$

Beziehungen unseres Beispiel-Schemas

hören : {[MatrNr: integer, VorlNr: integer]}

lesen : {[PersNr: integer, VorlNr: integer]}

arbeitenFür : {[AssistentenPersNr: integer, *ProfPersNr: integer*]}

voraussetzen : {[Vorgänger: integer, Nachfolger: integer]}

prüfen : {[MatrNr: integer, VorlNr: integer, PersNr: integer,
Note: decimal]}

Schlüssel der Relationen

hören : {[MatrNr: integer, VorlNr: integer]}

lesen : {[PersNr: integer, VorlNr: integer]}

arbeitenFür : {[AssistentenPersNr: integer, *ProfPersNr: integer*]}

voraussetzen : {[Vorgänger: integer, Nachfolger: integer]}

prüfen : {[MatrNr: integer, VorlNr: integer, PersNr: integer,
Note: decimal]}