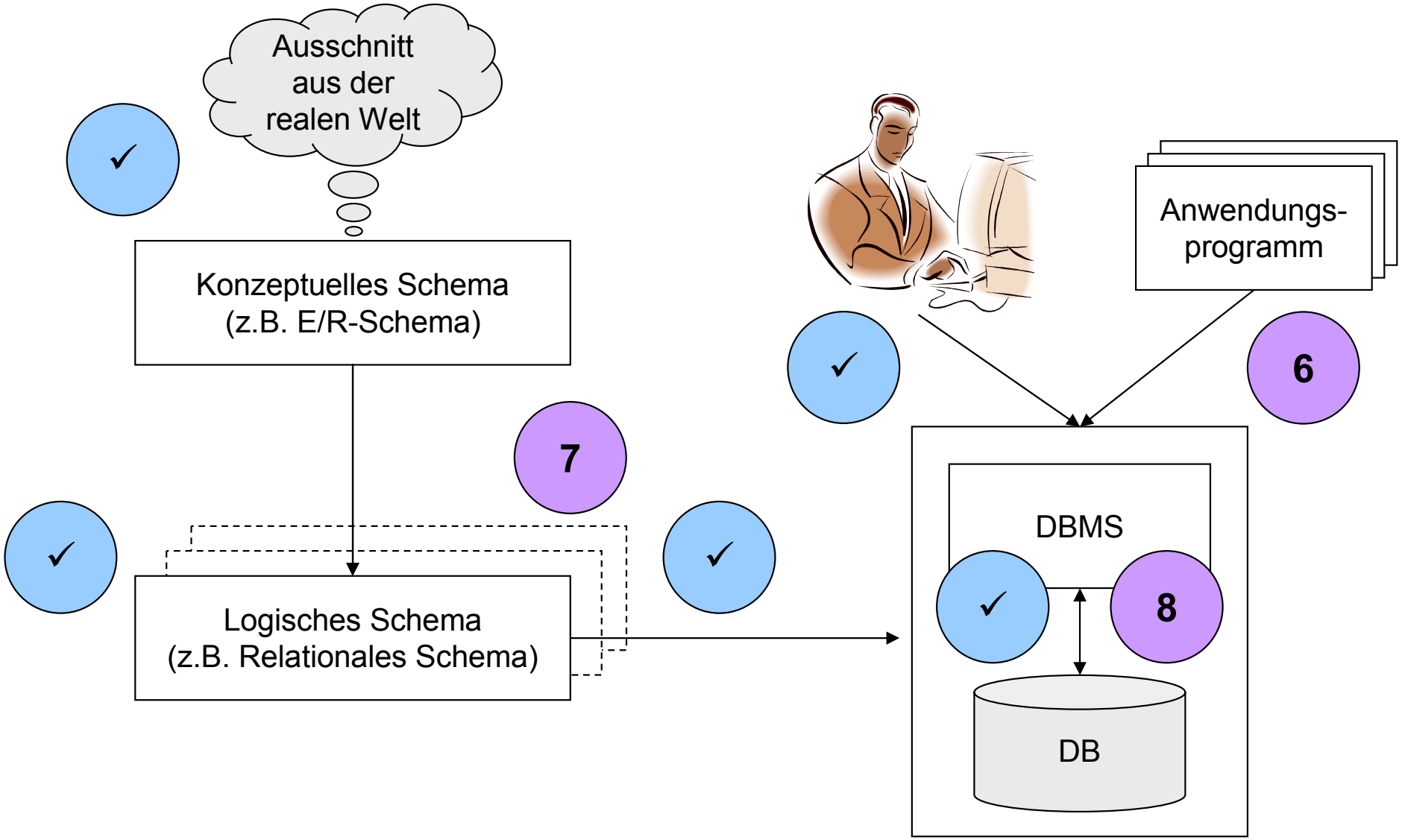
Datenbanken II

Sommersemester 2006

- Materialien zur Vorlesung -

Prof. Dr. Inge Schestag

Hochschule Darmstadt
Fachbereich Informatik


Ausschnitt
aus der
realen Welt
Konzeptuelles Schema
(z.B. E/R-Schema)
Logisches Schema
(z.B. Relationales Schema)
Anwendungs-
programm
6
7
8
DBMS
DB

Inhalt Datenbanken II - Vorlesung

Inhalt Datenbanken II - Praktikum

Aufgabenstellung	verwendete SW
1. Entwurf eines kleinen **ER-Diagrammes**, Ableitung des **relationalen Schemas** sowie die Generierung des **SQL-DDL-Skripts** zur Implementierung der Datenstruktur Die Datenstruktur soll im Rahmen des Praktikums auf dem Oracle-Server generiert werden. **SQL-DML**: Alle Tabellen werden mit Hilfe von interaktivem SQL oder kleinen SQL-Skripts mit Daten gefüllt, die Daten werden modifiziert, gelöscht und anschließend die gesamte Datenstruktur wieder vom DB-Server entfernt.	*SQL*PLUS* (SQL - *Oracle 10g*)
2. Entwurf eines größeren, **konzeptionellen und physischen relationalen Schemas** mit Hilfe des CASE-Tools *Data Architect*	*Data Architect - Power Designer 11*
3. SQL-Abfragen auf der Basis einer vorgegebenen Datenbank	*SQL*PLUS* (SQL - *Oracle 10g*)
4. **Stored Procedures & Trigger**	*SQL*PLUS* (SQL - *Oracle 10g*)
5. Datendeklaration und –manipulation auf einem **objektrelationalen DBMS**	*SQL*PLUS* (SQL - *Oracle 10g*)
6. Java Applikation mit **JDBC-Zugriff** auf eine relationale Datenbank unter Berücksichtigung **konkurrierender Zugriffe** auf die gleiche DB.	*Java (Eclipse), MS Access / MySQL / ...*

Praktikum	freitags 3y	freitags 3x
P1	31.03.2006	07.04.2006
P2	21.04.2006	28.04.2006
P3	05.05.2006	12.05.2006
P4	19.05.2006	26.05.2006
P5	02.06.2006	09.06.2006
P6	16.06.2006	23.06.2006

Vorlesungs- und Praktikumsunterlagen

Online unter http://www.fbi.fh-darmstadt.de/~schestag/ werden im Laufe des Semesters jeweils einige Tage vor der Vorlesung bzw. spätestens eine Woche vor dem ersten Praktikumstermin online bereit gestellt

Klausur

- Termin: 5. Juli 2006, 12:00 -13.30 Uhr, Raum B11 /
- An- bzw. Abmeldung ausschließlich über das OBS, nur angemeldete Studenten dürfen teilnehmen! **Anmeldung + Nichtteilnahme = nicht bestanden (5)**
- erlaubte Hilfsmittel: 1 (beidseitig) beschriebenes / bedrucktes A4-Blatt
- Stil der Klausur: Aufgaben ähnlich zu Hörsaalübungen + Verständnisfragen

Literatur

- Heuer, Andreas, Saake, Gunter. *Datenbanken: Konzepte und Sprachen*. MITP-Verlag, Bonn, 2. Auflage 2000.

- Date, C.J. *An Introduction to Database Systems*. International Edition. Addison-Wesley Systems. 8th ed. 2003.

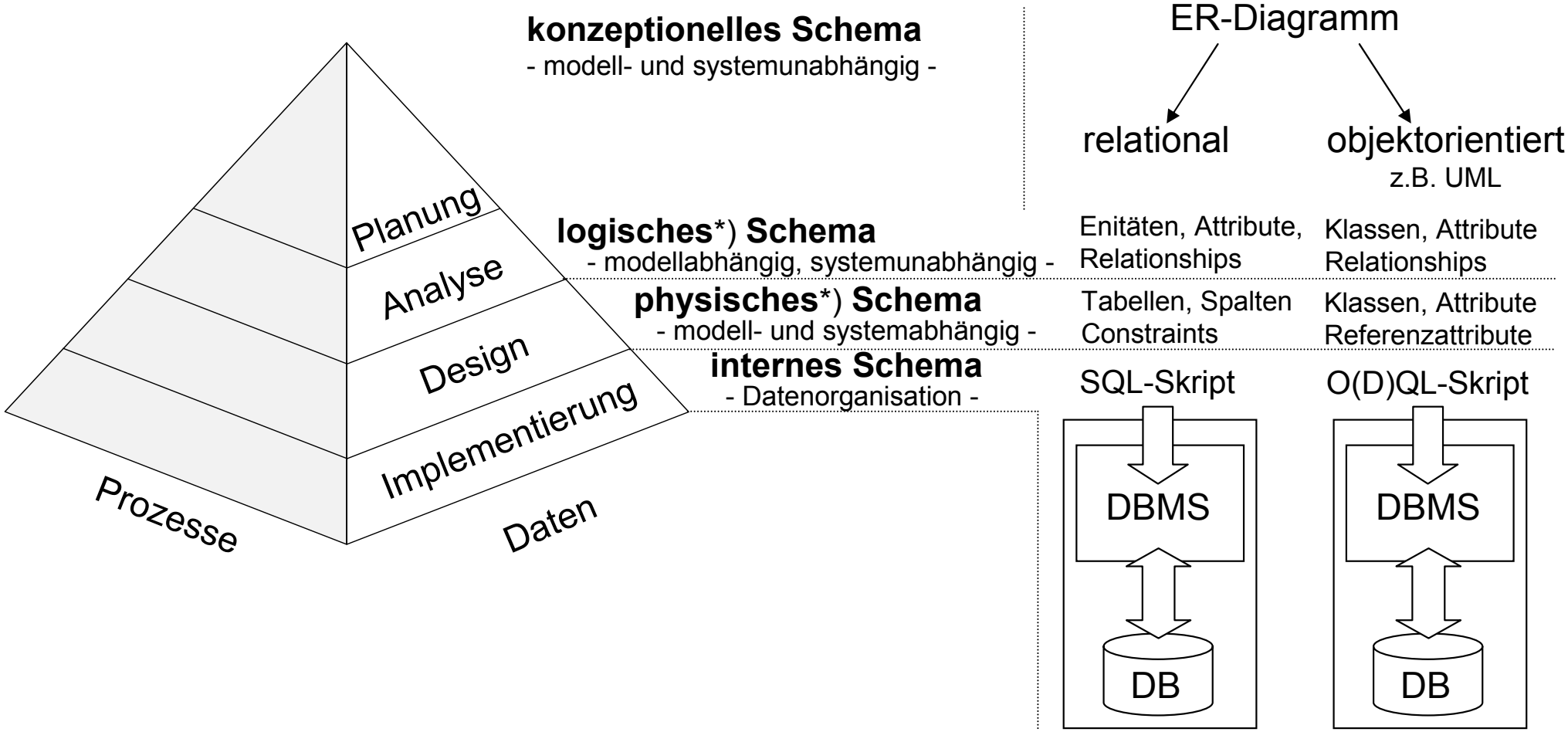
- Erbs, Karczewski, Schestag. *Datenbanken. Datenmodelle, Objekte, WWW, XML*. VDE Verlag 2003.

- Zehnder, Carl August. *Informationssysteme und Datenbanken*. VdF Hochschulverlag, 8. Auflage 2005.

- Heuer, Andreas. *Objektorientierte Datenbanken. Konzepte, Modelle, Standards und Systeme*. Addison Wesley, 2. Auflage 1997.

- Saake, Gunter, Sattler, Kai-Uwe. *Datenbanken & Java. JDBC, SQLJ, ODMG und JDO*. dpunkt.verlag, 2. Auflage 2003.

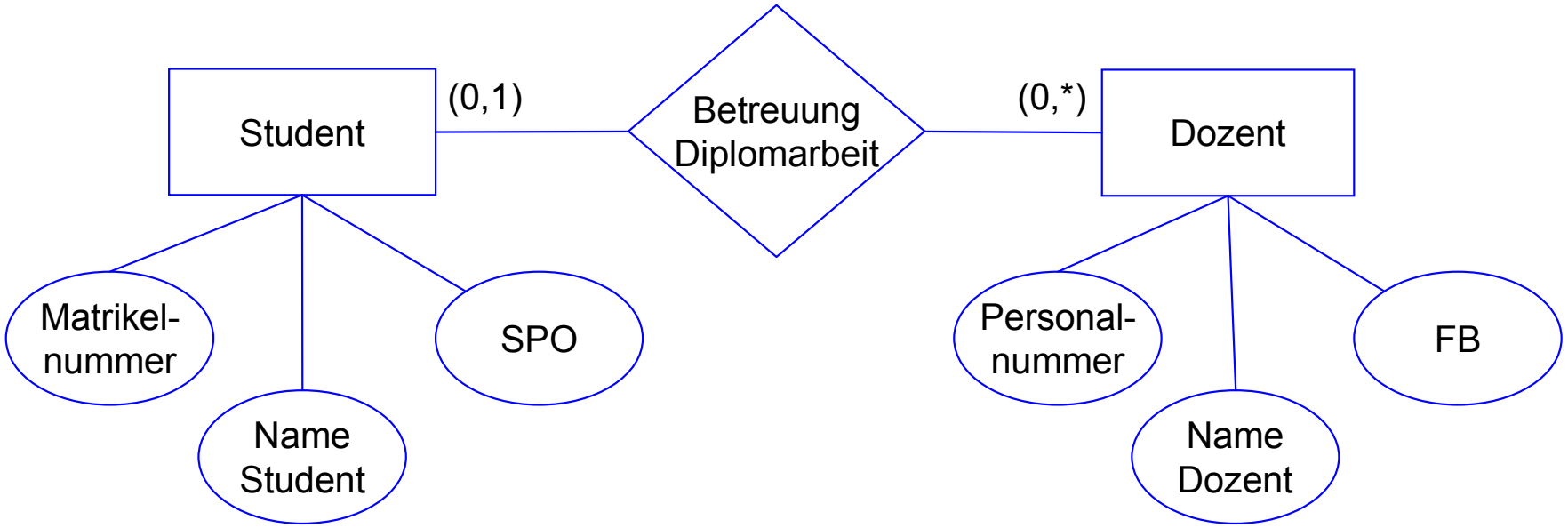
Wiederholung DB I: Die Rolle der Datenmodellierung im Software Lifecycle (1)



*) Der Begriff des 'physischen' Schemas weicht in diesem Teil II der Vorlesung etwas ab von Teil I und orientiert sich an der Nomenklatur des in der Vorlesung und im Praktikum eingesetzten Tools *Data Architect (Power Designer).* Das im folgenden und in dem eingesetzten Tool verwendete **physische Schema** entspricht einem logischen relationalen Schema, das bereits Implementierungsinformationen enthält (im Sinne eines UML-Klassendiagramms der Designphase).

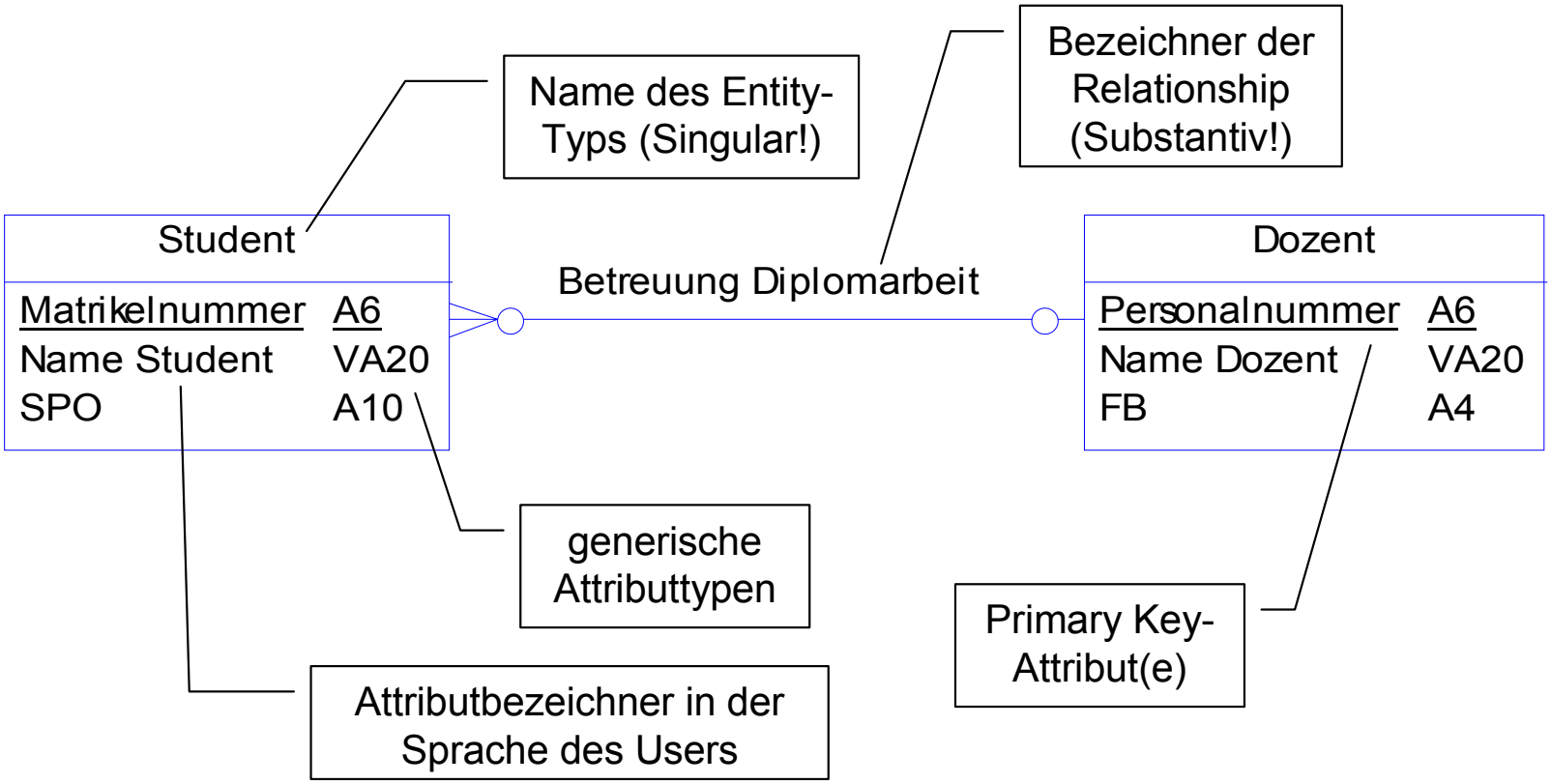
- Das **konzeptionelle Schema** beschreibt als **Entitiy-Relationship Diagramm** die Datenstrukturen (Entitäten mit Attributen) und ihre Beziehungen (Relationships) untereinander – unabhängig von einem spezifischen Datenbankmodell, d.h. unabhängig davon, ob das logische Schema z.B. ein relationales oder objektorientiertes Modell beschreibt.

  Insbesondere ist es deshalb also unabhängig von einem speziellen DBMS (Oracle, DB2, PostgreSQL, MySQL, FastObjects etc.).



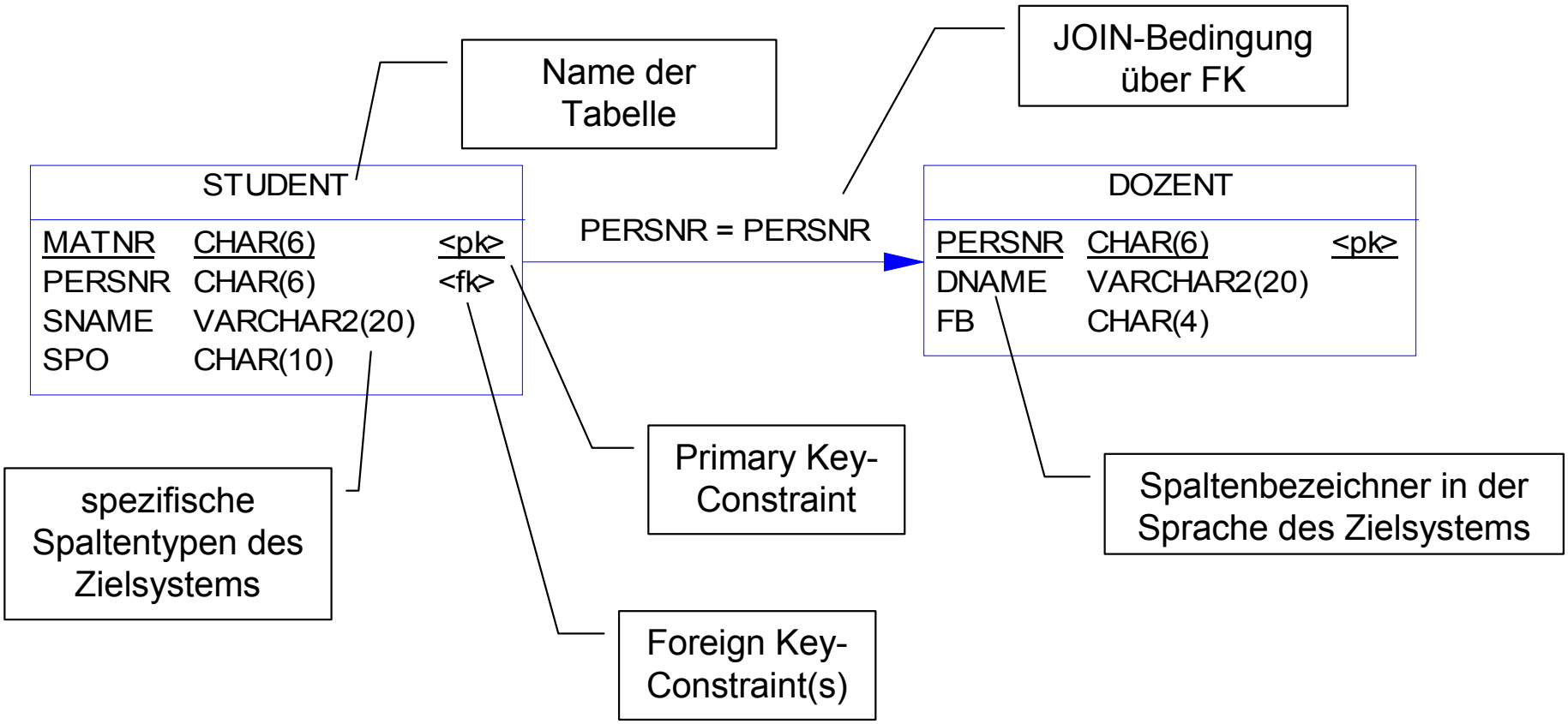
Wiederholung DB I: Das konzeptionelle Schema – im *Data Architect*

- Die Darstellung des **konzeptionellen Schema** im Tool ***Data Architect*** entspricht in seinen spezifischen Eigenschaften (s. Grafik) der Darstellung eines UML-Klassen-diagramms (Analysesicht), das ebenfalls (optional) generische Attributtypen enthält sowie Bezeichner „in der Sprache des Users“. Es enthält jedoch noch keinerlei imple-mentierungsspezifische Eigenschaften. Die Positionierung der Kardinalitäten ent-spricht der UML-Lesart.



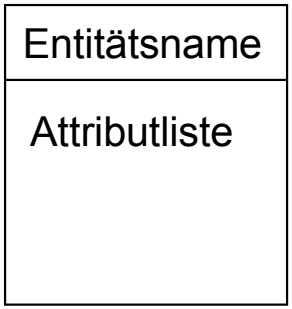
Wiederholung DB I: Das physische Schema – im *Data Architect*

- Das **physische Schema** des ***Data Architect*** kann bezeichnet werden als ein Relationales Schema (im Sinne des Teil I der Vorlesung), das bereits Implementierungseigenschaften im Sinne eines Klassendiagramms der Designphase aufweist: Dies zeigt sich z.B. durch die Verwendung von Datentypen des Zielsystems sowie durch die Implementierung einer Beziehung mittels FK-Spalte (Referenzattibut).



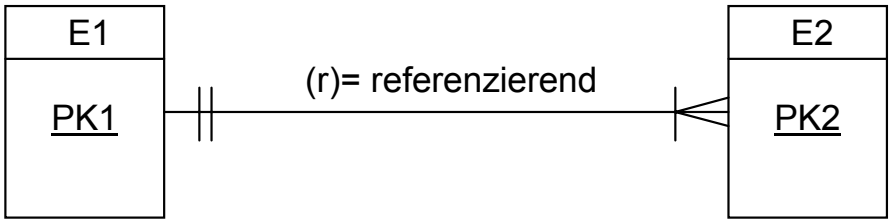
Wiederholung DB I: Strukturelemente eines relationalen Schemas (1)

konzeptionelles Schema
Analyse



Entitätstyp / Entitytype

Primary-Key-Attribute sind in der Regel durch einfache oder doppelte Unterstreichung gekennzeichnet.



Relationship mit **Kardinalitäten**
(auch Assoziation / Beziehung)

physisches relationales Schema
Design

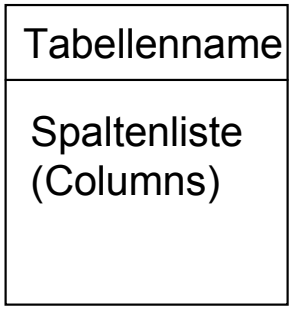
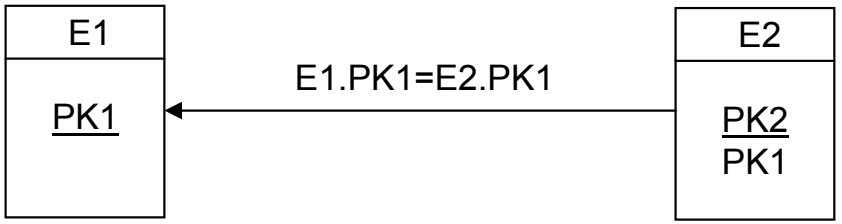


Tabelle / Table

Primary-Key-Spalten sind in der Regel durch einfache oder doppelte Unterstreichung gekennzeichnet.



Referenzierung mit JOIN-Bedingung

Wiederholung DB I: Strukturelemente eines relationalen Schemas (2)

Transformationen beim Übergang vom konzeptionellen zum physischen relationalen Schema:

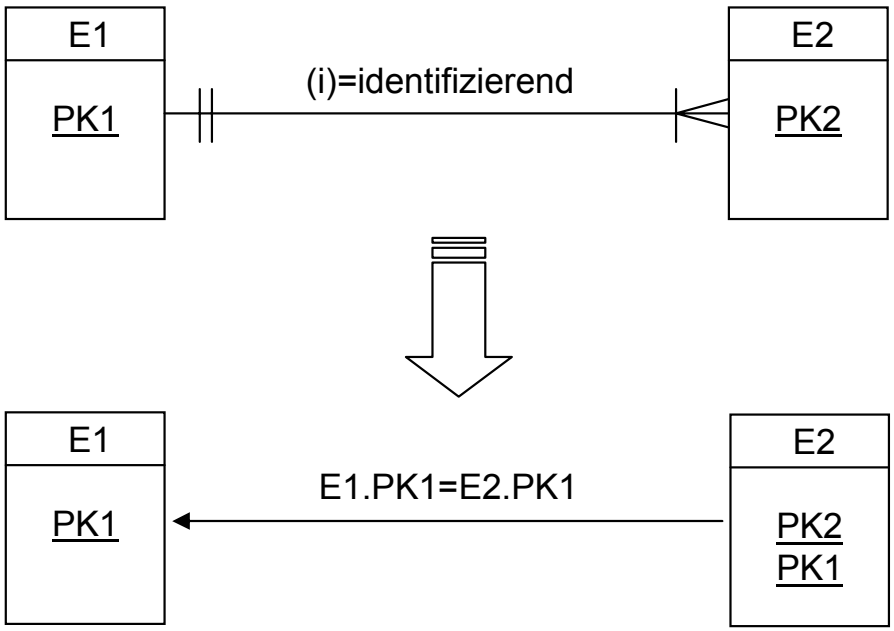
1. Bei einer referenzierenden Relationship (r) (not dependent) wird das Primar Key-Attribut PK1 zur **Foreign Key-Spalte** in der Tabelle E2 und hat nur referenzierende Eigenschaft: JOIN-Spalte beim Lesen.

2. Bei einer identifizierenden Relationship (i) (dependent) ist jede Ausprägung der Entität E2 abhängig (dependent) von genau einer Ausprägung der Entität E1.
   Deshalb wird das Primar Key-Attribut PK1 nicht nur zur **Foreign Key-Spalte** in der Tabelle
   E2 (referenzierende Eigenschaft), sondern darüber hinaus auch noch **Primary Key-Komponente** (identifizierende Eigenschaft – vgl. auch die folgende Folie).

3. Bei den meisten relationalen DBMS wird automatisch zusätzlich zu einer PK-Spalte auch ein **Primary Key-Constraint** generiert.

4. Es empfiehlt sich außerdem, zu jeder FK-Spalte einen **Foreign Key-Constraint** zu generieren, um die Integrität der Daten beim Schreiben zu gewährleisten.

Wiederholung DB I: Strukturelemente eines relationalen Schemas (3)

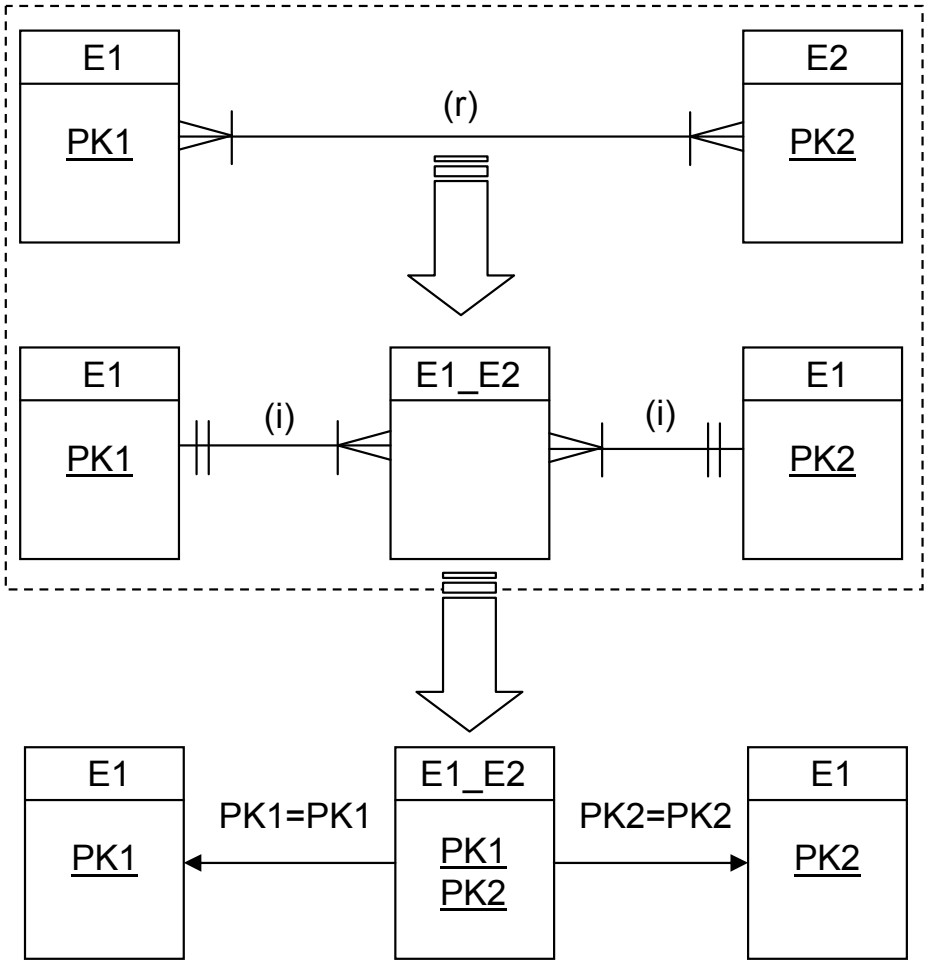


Die attributive Entitität

Eine attributive Entität (im Bild E2) entsteht aufgrund der 1. Codd‘schen Normalform (1NF), die **mehrwertige Attribute** innerhalb einer Entität nicht zulässt.

Mehrwertige Attribute werden in einer eigenen Entität ausgelagert (attributive Entität), sind aber von der abgeleiteten Kernentität E1 abhängig (dependent) und benötigen deshalb die Primary Key-Komponente PK1 nicht nur zum Referenzieren sondern auch zum Identifizieren der zugehörigen Ausprägung aus E1.

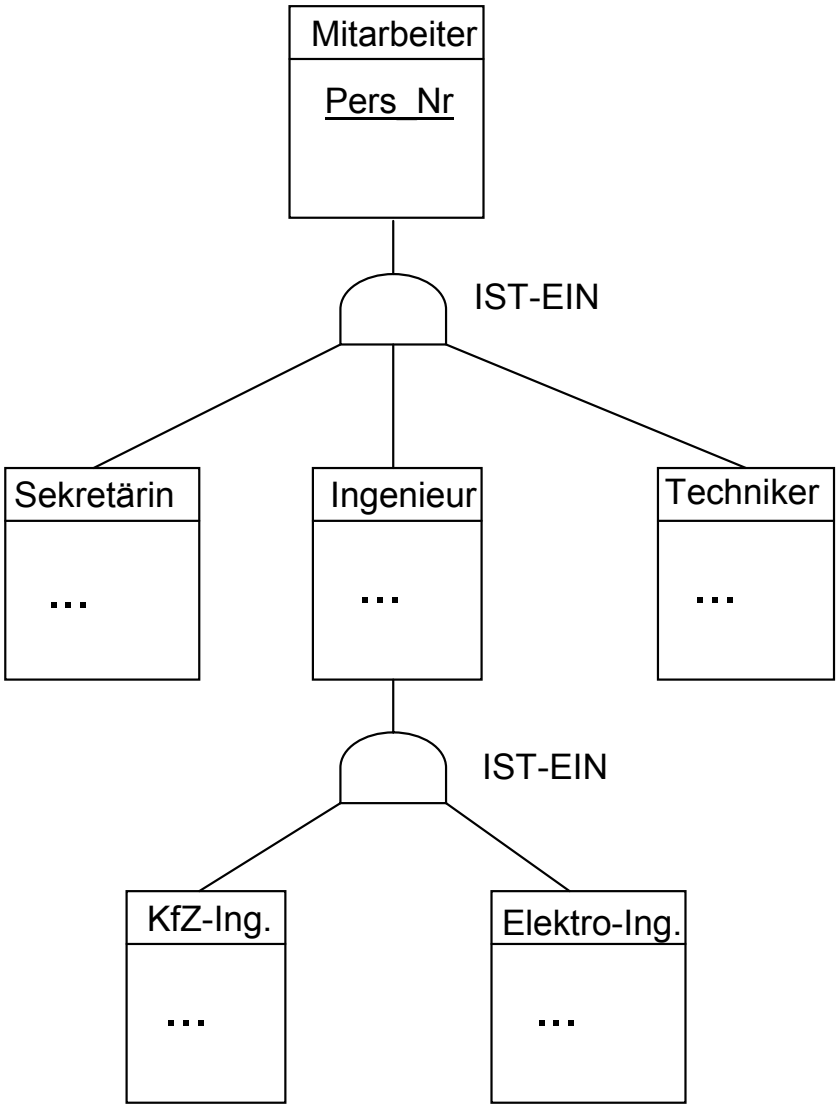
Wiederholung DB I: Strukturelemente eines relationalen Schemas (4)



Die assoziative Entitität

Assoziative Entitäten entstehen bei der **Auflösung von n:m-Beziehungen**, die man als doppelte Verletzung der 1NF interpretieren kann. Als Repräsentant der n:m-Beziehung (Assoziation) entsteht die assoziative Entität (im Bild E1_E2

Wiederholung DB I: Supertype-/Subtype-Hierarchie (Vererbung – Inheritance) (1)



Kann eine klar abgegrenzte Teilmenge einer Entität durch zusätzliche Attribute näher beschrieben werden, so kann man aus diesen Attributen eine abgeleitete Entität als Ergänzung zur Ausgangsentität bilden.

Die Ausgangsentität nennt man **Supertype**, die abgeleitete Entität **Subtype**.

Ist eine Entität E2 Subtype einer Entität E2, so ist jede Instanz von E2 auch eine Instanz von E1.

E1 ist die **Generalisierung** von E2,
E2 ist die **Spezialisierung** von E1.

Wiederholung DB I: Supertype-/Subtype-Hierarchie (Vererbung – Inheritance) (2)

Ein Exemplar eines Subtypes kann nicht existieren, ohne auch ein Mitglied des Supertypes zu sein. Beide Exemplare stellen das gleiche reale Objekt dar.

Es muß jedoch nicht jedes Mitglied des Supertypes auch Mitglied eines Subtypes sein; die Subtypen schließen sich nicht gegenseitig aus.

Subtypen können selbst wieder Subtypen haben.

Beziehungen zwischen Super– und Subtypen sind 1:1-Beziehungen, die man als **IST-EIN-Beziehung** bezeichnet (engl.: **IS-A**).

Subtypen haben alle Attribute ihrer übergeordneten Typen und einige, aber nicht notwendig alle ihrer Beziehungen.

Ein Subtyp kann zusätzliche Attribute und Beziehungen haben, die spezifisch für ihn sind.

Für die **Implementierung** einer Supertype- / Subtype-Hierarchie gibt es verschiedene Methoden $\Rightarrow$ vgl. Skript U. Störl (WS 05/06, Kapitel 3) und Praktikum 2 des SoSe 06.