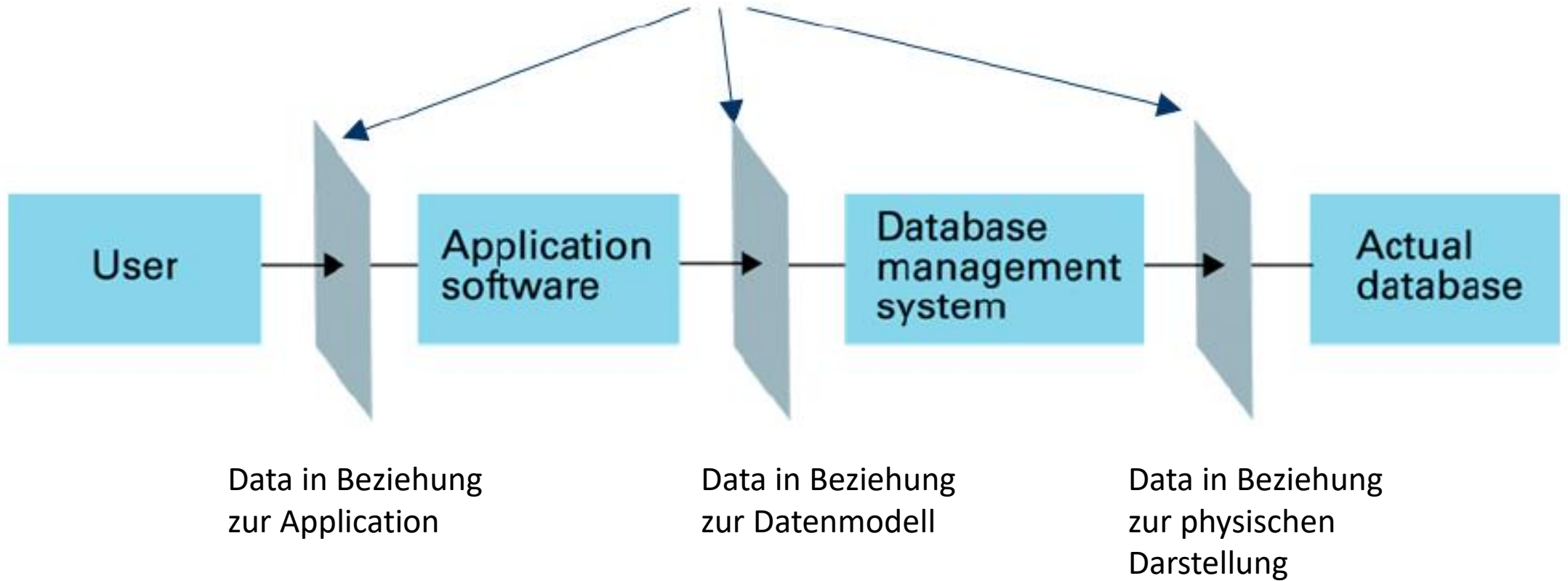


Abstraktionsschichten. Das Relationale Datenmodell

Verschiedene Abstraktionsebene

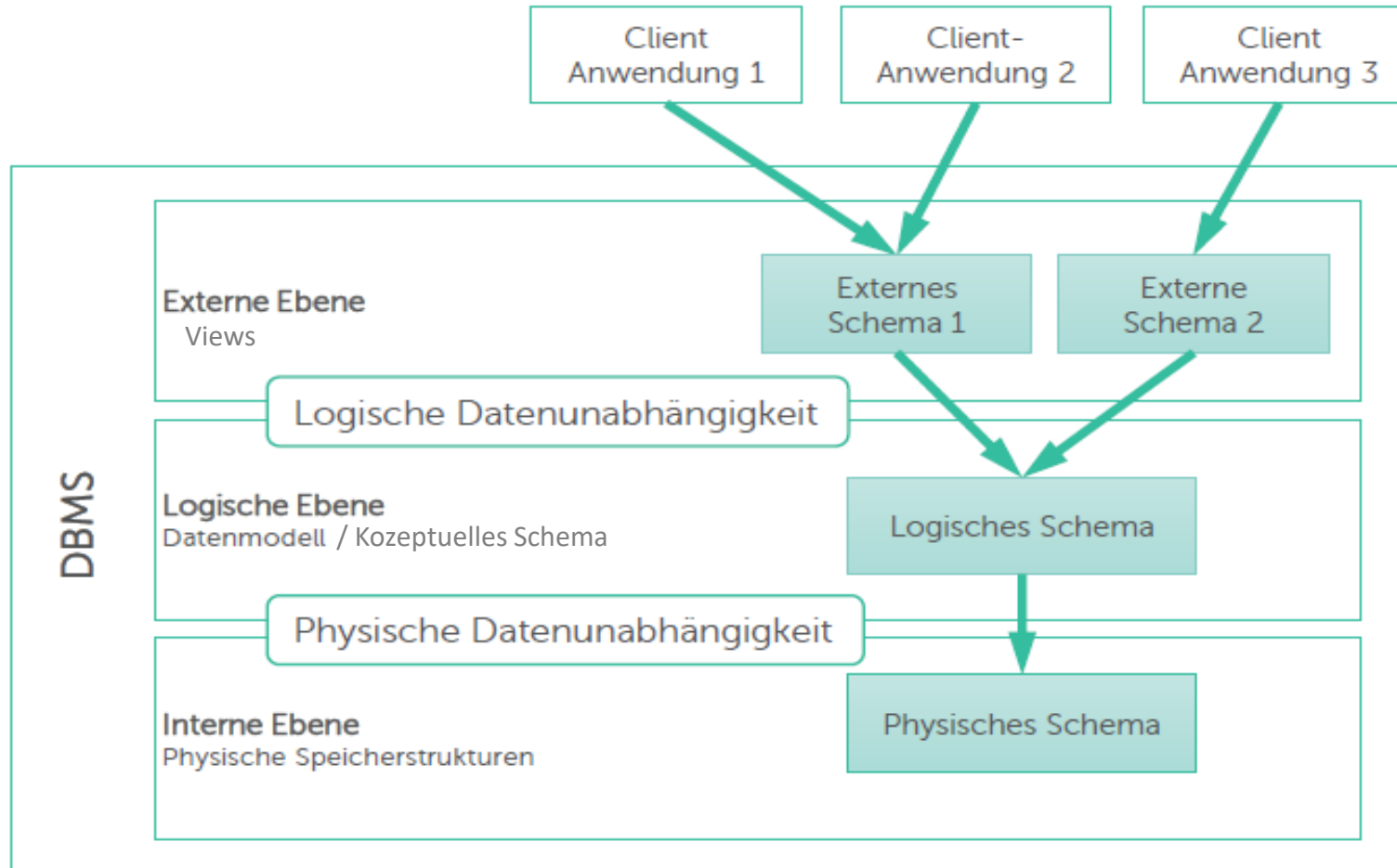


Datenunabhängigkeit

- Ziel: die Datendefinition in einer Schicht zu ändern, ohne dabei die Definition der Daten in der darüber liegenden Schicht zu beeinflussen
- Logische Datenstrukturen unabhängig von physischen Datenstrukturen
- Robustheit der Anwendungen gegenüber Änderungen

Abstraktionsebenen eines Datenbanksystems

- ANSI-SPARC-Architektur



3-Schichten nach ANSI-SPARC

- Externe Ebene:
 - Anwendungs-spezifische Sichten/Views (Ausschnitte aus dem Datenmodell)
 - Ebene der Anwendungen und Verwendung der Daten
 - Beschreibt wie der Benutzer die Daten sieht
- Logische Ebene:
 - Definition der logischen Datenstrukturen: die Beziehungen zwischen den Daten unabhängig von der physischen Repräsentation
- Interne/Physische Ebene:
 - Definition des physischen Schemas (wie die Daten gespeichert werden)
 - Beschreibt die Dateien und Indexstrukturen die benutzt werden
 - Es geht um Leistungsfähigkeit der Datenbankanwendungen

Beispiel: Universität Datenbank

- Logische Ebene/Konzeptuelles Schema:
 - **Studenten**(sid:string, name:string, email:string, age:integer, gruppe:integer)
 - **Vorlesung**(vid:string, vname:string, ects:integer)
 - **Klausur**(sid:string, vid:string, note:integer)
- Physische Ebene:
 - Relationen werden als ungeordnete Dateien gespeichert
 - Indexstruktur auf die erste Spalte (sid) der Tabelle **Studenten**
- Extere Ebene/View:
 - Vorlesungsinfo(cid:string, enrollment:integer)

Physische Datenunabhängigkeit

- Änderungen an der Art der Datenspeicherung und den Zugriffstechniken haben keinen Einfluss auf Anwendungsprogramme
- Programme sind von interner Datenorganisation unabhängig
- Physische Datenunabhängigkeit wird durch relationale DBMS weitestgehend hergestellt
- Beispiele:
 - Verschiebung von Datenbank auf anderes Storage-System
 - Partitionierung
 - Indexstrukturen für performante Zugriffe

Logische Datenunabhängigkeit

- Ermöglicht durch externe Ebene
- Änderungen am logischen Schema haben nur geringe Auswirkungen auf die Anwendung
- Ist nur begrenzt möglich, den Anwendungen basieren sich auf dem logischen Schema
- Sichtenkonzept (Views) ermöglicht Verbergen von Details des logischen Datenmodells
- Beispiel: nach Umbenennung einer Tabelle oder eines Attributes kann die alte Datenstruktur virtuell über eine Sicht bereitgestellt werden

Queries (Abfragen) im DBMS

- Mögliche Fragen für die Universität Datenbank:
 - Welcher ist der Name des Studenten mit sid=2310?
 - Wie viele Studenten haben sich für die Vorlesung DB angemeldet?
 - Wie viele Studenten haben in der DB Klausur mehr als 9 gekriegt?
- Fragen, die sich auf Daten gespeichert im DBMS beziehen, sind Datenbank-Abfragen
- DBMS hat bestimmte Datenbanksprachen zum Abfragen und Manipulieren der Daten (Query Language)

Datenbanksprache

- Formale Sprache die folgende Komponenten hat:
 - Datenbeschreibungssprache / Data Definition Language (DDL)
 - Befehle zur Definition des Datenbankschemas (Anlegen, Ändern und Löschen von Datenstrukturen)
 - Constraint Definition Language (CDL) – beschreibt Integritätsregeln, die von den Instanzen der Datenbank erfüllt werden sollen
 - Storage Definition Language (SDL) – um den Layout der physischen Schema zu beeinflussen
 - Datenmanipulationssprache / Data Manipulation Language (DML)
 - Befehle zur Datenmanipulation (Abfragen, Einfügen, Ändern oder Löschen von Nutzdaten)
 - Procedural DML (wie) vs. Declarative DML (was)

Abfragesprachen für relationale Datenbanken

- SQL (Structured Query Language)
 - SELECT name FROM studenten WHERE age>20
- Relationale Algebra
 - $\pi_{\text{name}}(\sigma_{\text{age}>20}(\text{Studenten}))$
- Domänenkalkül (Domain Calculus)
 - $\{ \langle X \rangle \mid \exists V \exists Y \exists Z \exists T: \text{Studenten}(V, X, Y, Z, T) \wedge Z > 20 \}$
- Tupelkalkül (Tuple Calculus)
 - $\{ X \mid \exists Y : Y \in \text{Studenten} \wedge Y.\text{age} > 20 \wedge X.\text{name} = Y.\text{name} \}$

Das Relationale Datenmodell

- Verwendet einfache Datenstrukturen: **Tabellen**
 - Einfach zu verstehen
 - Mengenorientiert
 - Nützliche Datenstruktur (passend für viele Situationen)
 - Hat eine nicht zu komplizierte Abfragesprache
- Grundlage des Konzeptes: **Relation**
 - Eine mathematische Beschreibung einer Tabelle
 - Führt zu formellen Abfragesprachen

Terminologie

- **Domänen/Wertebereiche** – Integer, String, Datum, ...
- **Relation** – besteht aus Attribute und Tupeln
- Relation R hat ein Relationsschema RS und eine Ausprägung
 - **Relationenschema** RS: legt die Struktur der gespeicherten Daten fest
 - Menge von Attributen $\{A_1, \dots, A_k\}$
 - Attribute A_j : Wertebereiche $D_j = \text{dom}(A_j)$
 - **Ausprägung**: der aktuelle Zustand der Datenbasis
 - Teilmenge des kartesischen Produkt der Wertebereiche, $\text{val}(R) \subseteq D_1 \times D_2 \times \dots \times D_k, k \geq 1$
- **Datenbankschema** – Menge der Relationenschemata;
- **Datenbank** – Menge der aktuellen Relationen

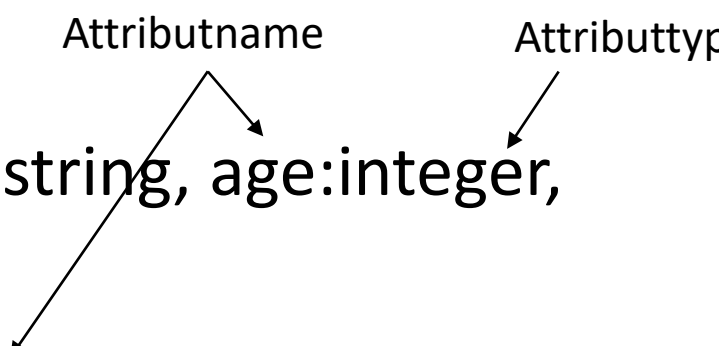
Terminologie

- Ein Attribut beschreibt den Typ eines möglichen Attributwertes und bezeichnet ihn mit einem Attributnamen
- **Tupel**/Datensatz – Element einer Relation (eine konkrete Kombination von Attributwerten)
- Alle Tupel in der Relation sind verschieden
- **Kardinalität** – Anzahl der Tupel in einer Relation
- **Grad k einer Relation** (Degree) – Anzahl von Attributen in der Relationsschema;

$$R \subseteq D_1 \times D_2 \times \dots \times D_k, k \geq 1$$

Relation - Beispiel

- **Studenten**(sid:string, name:string, email:string, age:integer, gruppe:integer)



sid	Name	Email	Age	gruppe
2831	Anne	anne@scs.ubbcluj.ro	20	231
2532	Silvia	silvia@scs.ubbcluj.ro	19	233
2754	Hannes	hannes@scs.ubbcluj.ro	21	231

Relationsschema

Ausprägung-/Instanz der Relation

Tupel

Kardinalität = 3

Grad/Degree = 5

Grundregeln

- Jedes Tupel (Zeile) ist eindeutig und beschreibt ein Objekt
- Die Ordnung der Zeilen und Spalten ist ohne Bedeutung
- Jeder Datenwert innerhalb einer Relation ist ein atomares Datenelement (integer, string, date)

Integritätsregeln (Integrity Constraints)

- Regeln, die für jede Instanz der Datenbank erfüllt werden sollen
- Integritätsregeln werden beim Erstellen des Schemas festgelegt
- Fehlerhafte Datensätze werden nicht angenommen
- Beispiel von Integritätsregeln:
 - Studenten**(sid:string, name:string, email:string, age:integer, gruppe:integer)
 - Domäne-Constraints: gruppe:integer
 - Wertebereich-Constraints (Range constraints): $18 \leq \text{age} \leq 70$

Primärschlüssel

- Notation: R – Relation, $X \subseteq R$ (X ist eine Menge von Attributen aus der Relation R)

$\pi_X(t)$ = Tupel t eingeschränkt auf die Attribute X

- Definition. Eine Menge von Attributen $X \subseteq R$ wird als **Schlüsselkandidat** bezeichnet, wenn folgende Bedingungen erfüllt sind:
 - *Eindeutigkeit*: für alle Relationen R des Schemas RS gilt
$$\forall t_i, t_j \in R, \pi_X(t_i) = \pi_X(t_j) \Rightarrow i = j$$
 - *Definiertheit* $\forall t_i \in R, \pi_X(t_i) \neq NULL$
 - *Minimalität* $\nexists Z \subset X, Z \neq X$, so dass die vorigen Bedingungen erfüllt sind
- Intuitiv: Schlüssel (Schlüsselkandidat) sind minimale Mengen von Attributen, deren Werte ein Tupel eindeutig identifizieren

Primärschlüssel

- Intuitiv:
 - *Eindeutigkeit+Definiertheit = jedes Tupel eindeutig identifizieren*
 - *Minimal = bei Weglassen eines einzelnen Attributs geht die Eindeutigkeit verloren*
- **Primärschlüssel** = minimale Menge von identifizierenden Attributen
- Wenn eine Relation mehrere Schlüsselkandidaten besitzt, wird einer davon als *Primärschlüssel* ausgewählt
- Die anderen: *alternative Schlüssel*

Fremdschlüssel

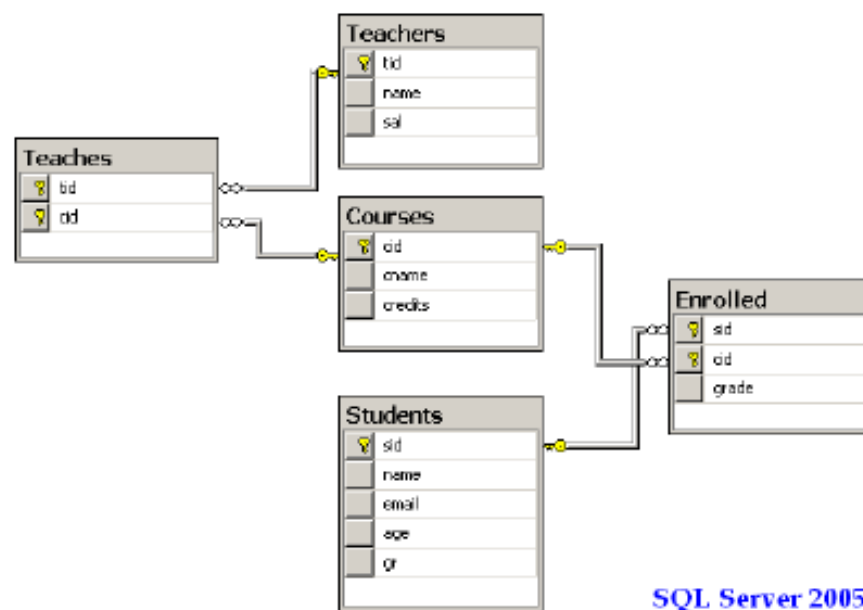
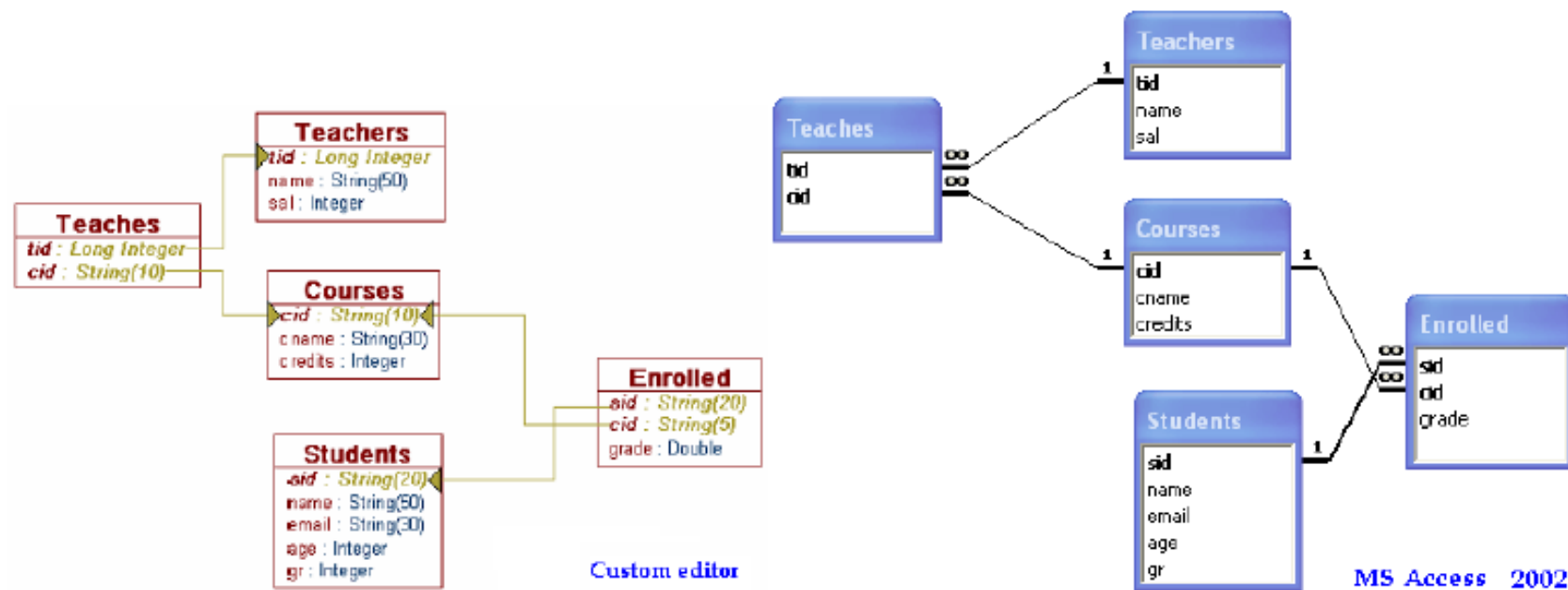
- Notation: Relation R_1 , Relation R_2 und $X \subseteq R_2$ Primärschlüssel
- Definition. $Y \subseteq R_1$ als **Fremdschlüssel** für R_1 bezüglich der Relation R_2 bezeichnet, wenn folgende Bedingungen erfüllt sind:
 - *Definiertheit* $\forall t_i \in R_1: (\pi_Y(t_i) = NULL \vee \exists t_j \in R_2: \pi_Y(t_i) = \pi_X(t_j))$
 - *Minimalität* $\nexists Z \subset Y, Z \neq Y$, so dass die vorige Bedingung erfüllt ist
- Intuitiv: ein Fremdschlüssel verweist auf einen Primärschlüssel einer anderen Relation

Fremdschlüssel - Beispiel

- **Studenten**(sid:string, name:string, email:string, age:integer, gruppe:integer)
- **Vorlesung**(vid:string, vname:string, ects:integer)
- **Klausur**(sid:string, vid:string, note:integer)
- Klausur:
 - sid – Fremdschlüssel, verweist auf Studenten
 - vid – Fremdschlüssel, verweist auf Vorlesung

Referenz-Integritätsregel

- Eine relationale Datenbank enthält keinen Fremdschlüssel (ungleich NULL), der auf einen nichtexistenten Primärschlüssel verweist.
- *Bemerkung.* Ein neuer Datensatz mit einem Fremdschlüssel kann nur dann in einer Tabelle eingefügt werden, wenn in der referenzierten Tabelle ein Datensatz mit entsprechendem Wert im Primärschlüssel existiert.
- *Problem:* ein Tupel mit Primärschlüssel auf den Fremdschlüssel verweisen kann nicht einfach gelöscht werden.
- Mögliche *Lösungen*:
 - Löschen/Ändern nicht durchführen
 - Löschen /Ändern rekursiv aller darauf verweisender Tupel
 - Nullsetzen aller darauf verweisender Fremdschlüssel



Relationale Datenbankabfragesprache/ Relational Query Language

- Vorteil des relationalen Modells:
 - unterstützt einfache Datenbankabfragesprachen
- Abfragen können intuitiv formuliert werden und der DBMS ist zuständig für die optimale Abfrage Bearbeitung
- Aber: manchmal müssen wir die Abfrage optimal formulieren um die gewünschte Zeitkomplexität zu erreichen

Structured Query Language (SQL)

- Ist die Sprache die von den meisten relationalen Datenbanken unterstützt wird
- Wurde 1970 von IBM entwickelt
- SQL Standard:
 - SQL 86
 - SQL 89
 - SQL 92 (SQL 2)
 - SQL 2003
 - SQL 2008
 - SQL 2011

SQL Befehle

- Kategorien von SQL Befehle:
 - Datenbeschreibungssprache / Data Definition Language (DDL)
 - Anlegen, Ändern und Löschen von Relationen oder Views
 - Beschreiben von Integritätsregeln
 - Datenmanipulationssprache / Data Manipulation Language (DML)
 - Einfügen, Ändern oder Löschen von Daten in der Relationen (Tupeln)
 - Abfragen (Queries)
 - Datenüberwachungssprache / Data Control Language (DCL)
 - Befehle für Rechteverwaltung (auf Tabellen und Views) und Transaktionskontrolle