

Tutorium 3 – Datenbanksysteme

16.11.2022 – Finn Kapitza





LUDWIG-
MAXIMILIANS-
UNIVERSITÄT
MÜNCHEN

1. Wiederholung Schlüssel und Fremdschlüssel



Definition Schlüssel

Teilmenge **S** der Attribute eines Relationenschemas **R** heißt Schlüssel, falls gilt

1. **Eindeutigkeit**: keine zwei Tupel dürfen sich in allen Attributen von **S** gleichen
2. **Minimalität**: keine echte Teilmenge von **S** erfüllt die **Eindeutigkeit**

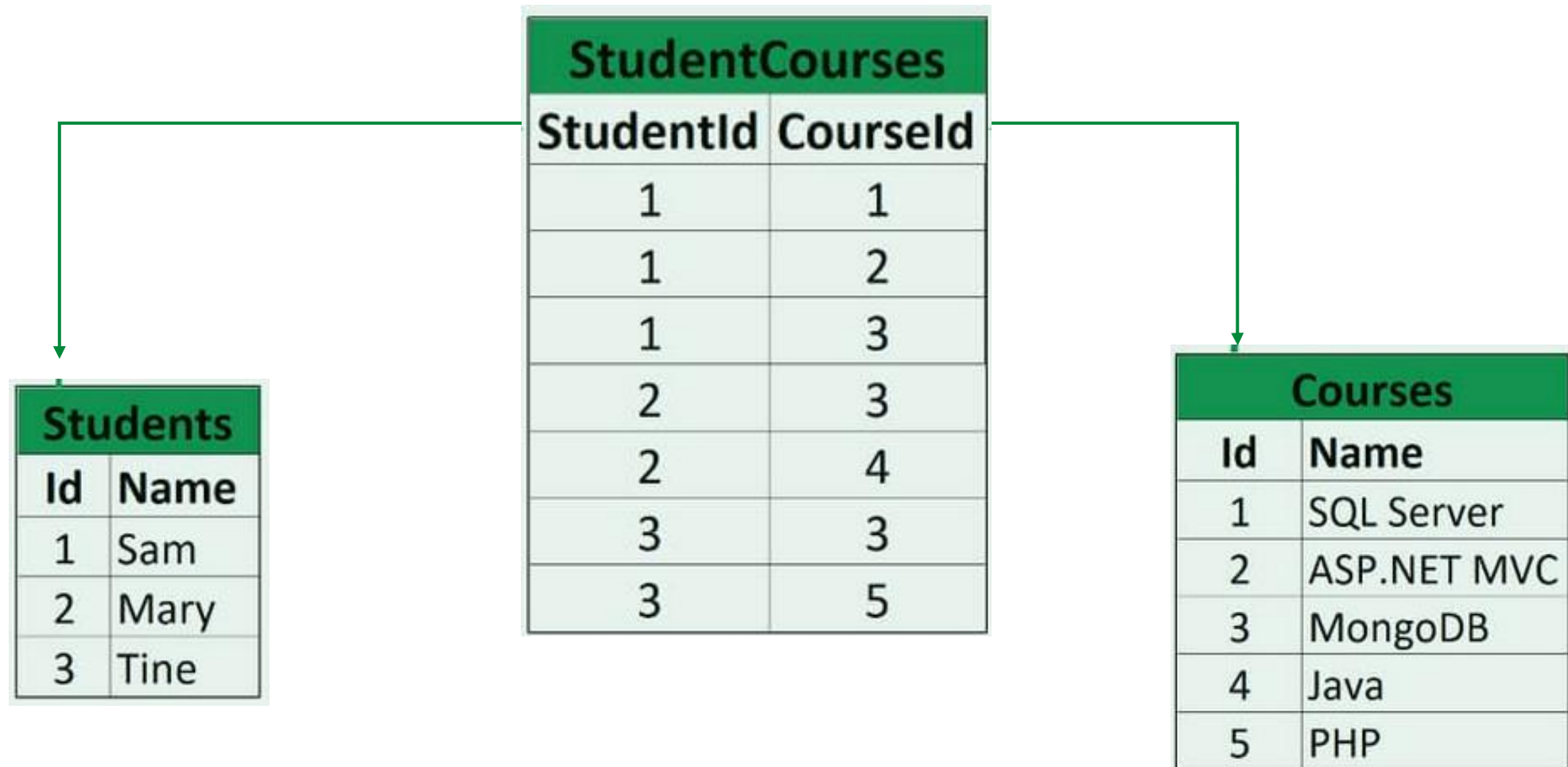
Trivial: Schlüssel mit nur einem Attribut sind immer minimal

Wenn zusammengesetzt: Prüfe jede Teilmenge von **S** auf **Eindeutigkeit**

Oft wird eine fortlaufende ID als Schlüssel eingesetzt → Einfach, nur ein Attribut, Eindeutig für jedes Tupel

- Attribut das auf einen Schlüssel einer anderen Relation verweist, heißt **Fremdschlüssel**
- Fremdschlüssel braucht einen gültigen Partner in der anderen Tabelle
- Vorsicht beim Löschen von Tupeln auf die (von einer anderen Tabelle) verwiesen wird

Beispiel – Schlüssel & Fremdschlüssel




Aufgabe 3.1 – Schlüssel und Fremdschlüssel

Eine relationale Datenbank enthält alle Informationen über bereits gesehenen Serien einer Streamingplattform und die darin vorkommenden Rollen:

Serie	<u>SID</u>	Sender	Serienname	
	47	HBO	Game of Thrones	
	59	CBS	The Big Bang Theory	

Rolle	<u>RID</u>	Vorname	Nachname	SID
	13	Jon	Snow	47
	19	Tyrion	Lannister	47
	24	Sheldon	Cooper	59
	33	Rick	Grimes	<i>null</i>



Die Attribute `Serie.SID` und `Rolle.RID` stellen die Primärschlüssel der beiden Relationen dar. Das Schema enthält außerdem folgende Fremdschlüsselbeziehung zwischen `Rolle` und `Serie`:

`Rolle.SID → Serie.SID`

Aufgabe 3.1.a – Wie reagiert ein Datenbanksystem wenn *Primärschlüssel* definiert wurden?

- Das Datenbankmanagementsystem (DBMS) verhindert das Einfügen von Duplikaten

```
CREATE TABLE Studierende (  
    Id Int Primary key,  
    Vorname Varchar(50)  
);
```

```
INSERT INTO Studierende VALUES  
(1, "Sam"),  
(2, "Mary"),  
(3, "Tine");
```

```
INSERT INTO Studierende VALUES  
(1, "Tom")
```



Schema Error: Error: ER_DUP_ENTRY: Duplicate entry '1' for key 'PRIMARY'

Aufgabe 3.1.b – Warum fordert man, dass Schlüssel *minimal* sein sollen?

- Effizientere Überprüfbarkeit von Schlüssel-/Fremdschlüsselbedingungen

Aufgabe 3.1.c – Was versteht man unter referenzieller Integrität?

- Zu jedem Attributwert in der abhängigen Relation muss es auch den entsprechenden Schlüsselwert in der referenzierten Relation geben!
- Keine *dangling references*

Aufgabe 3.1.d – Welche der Einfügeoperationen kann das Datenbanksystem erfolgreich verarbeiten?

- Einfügen von (12, „Harvey“, „Specter“, 41) in Rolle
-> referenzielle Integrität verletzt!! 41 ist kein Schlüsselwert in *Serie*
- Einfügen von (47, „FOX“, „Suits“) in *Serie*
-> Schlüsseleigenschaft von *SID* in *Serie* wird verletzt
- Einfügen von (42, „Leonard“, „Hofstadter“, 59) in Rolle
-> Kann erfolgreich eingefügt werden

<u>SID</u>	Sender	Seriename
47	HBO	Game of Thrones
59	CBS	The Big Bang Theory



<u>RID</u>	Vorname	Nachname	SID
13	Jon	Snow	47
19	Tyrion	Lannister	47
24	Sheldon	Cooper	59
33	Rick	Grimes	<i>null</i>

2. Wiederholung Data Definition Language (DDL)



Data Definition Language

-> Anlegen von Tabellen

```
CREATE TABLE tabellenname (  
    attribut1 datentyp1 [constraint11] [, ...],  
    attribut2 datentyp2 [constraint21] [, ...],  
    ...,  
    attributk datentypk ,  
    [tabellenconstraint1, ..., tabellenconstraintm]  
);
```

- **attribut_i** – Name des i-ten Attributs
- **datentyp_i** – Datentyp des i-ten Attributs
 - CHAR(*n*) – String der festen Länge *n*
 - VARCHAR(*n*) – String variabler Länge (maximal *n*)
 - INT – ganze Zahl (positive oder negative natürliche Zahl)
 - DECIMAL(*n*, *m*) – Festkommazahl mit *n* Stellen insgesamt, *m* davon hinter dem Komma
 - FLOAT – Gleitkommazahl, Kommazahl aber egal wie viele Stellen vor oder hinter dem Komma
- **constraint_{ik}** – k-ter Constraint des i-ten Attributs -> mehrere möglich (auch keiner)
 - NOT NULL – Attribut muss gefüllt werden
 - UNIQUE – Attribut darf nicht doppelt vorkommen
 - PRIMARY KEY – Attribut ist alleiniger Primärer Schlüssel
 - CHECK(*b*) – Attribut muss Bedingung *b* erfüllen (z.B. CHECK attribut_i > 0)
 - DEFAULT *x* – Wenn nicht gefüllt, dann Default wert *x*
 - REFERENCES *t*(*a*) – Fremdschlüssel der auf Attribut *a* in Tabelle *t* verweist

Data Definition Language -> Anlegen von Tabellen

- `tabellenconstrainti` – gilt meist für mehrere Attribute
 - `PRIMARY KEY(a1, ... ak)` – Zusammengesetzter Primärer Schlüssel
 - `FOREIGN KEY(a1, ... ak) REFERENCES t(b1, ... bk)` – Wenn auf einen zusammengesetzten Schlüssel verwiesen wird

```
CREATE TABLE tabellenname (  
    attribut1 datentyp1 [constraint11] [,...],  
    attribut2 datentyp2 [constraint21] [,...],  
    ...,  
    attributk datentypk [constraintk1] [,...],  
    [tabellenconstraint1, ..., tabellenconstraintm]  
);
```

Data Definition Language -> Verändern von Tabellen

```
ALTER TABLE tabellenname  
    ADD (attribut datentyp); |  
    MODIFY (attribut datentyp); |  
    DROP (attribut);
```

- *ADD (attribut datentyp)* – Hinzufügen eines Attributs
- *MODIFY (attribut neuer_datentyp)* – Ändern eines Attributs
- *DROP (attribut)* – Löschen eines Attributs
- *ADD CONSTRAINT (constraint_name constraint)* – Hinzufügen eines *constraint* mit Name = *constraint_name*

Data Definition Language -> Löschen von Tabellen

```
DROP TABLE tabellenname;
```

- Tabelle mit Name *tabellenname* wird gelöscht

-> Auf referenzielle Integrität aufpassen

Wenn Tabelle **ABC** auf Tabelle **A** verweist, darf Tabelle **A** nicht zuerst gelöscht werden, da die Verweise dann in der Luft hängen (dangling references)

Aufgabe 3.2 – SQL-DDL

Gegeben sei die Datenbank eines Softwareunternehmens, welches spezielle Anwendungen für unterschiedliche Kunden entwickelt. Sie enthält die Relationen Kunde K, Team T und Leistung L. Zusätzlich existiert eine Relation KTL, welche die Aufträge beinhaltet und so die Beziehungen der vorgenannten Relationen modelliert:

K (KNr, KName, Ansprechpartner)
T (TNr, TLeiter, TGröße, Stundensatz)
L (LNr, Bezeichnung, Komplexität)
KTL (KNr, TNr, LNr, Volumen, Datum)

Die Schlüssel der jeweiligen Relationen sind integer Werte. Das Attribut TGröße ist eine positive ganze Zahl. Das Attribut Komplexität ist eine ganze Zahl kleiner oder gleich 10. Das Attribut Stundensatz ist eine Festkommazahl mit insgesamt 5 Stellen, davon 2 Nachkommastellen. Das Attribut Volumen ist eine Fließkommazahl. Das Attribut Datum ist ein String der Länge 10. Alle übrigen Attribute sind variable Strings mit maximaler Länge von 50 Zeichen. Die Attribute KName, TLeiter und Bezeichnung müssen immer einen Wert enthalten.

Aufgabe 3.1.a – CREATE TABLE Befehle mit notwendigen Constraints für das oben genannte Schema

Tabelle K:

```
CREATE TABLE K (  
    KNr                INTEGER        PRIMARY KEY,  
    KName              VARCHAR(50)    NOT NULL,  
    Ansprechpartner VARCHAR(50)  
);
```

Aufgabe 3.1.a – CREATE TABLE Befehle mit notwendigen Constraints für das oben genannte Schema

Tabelle T:

```
CREATE TABLE T (
    TNr                INTEGER        PRIMARY KEY,
    TLeiter           VARCHAR(50)    NOT NULL,
    TGröße            INTEGER        CHECK(TGröße > 0),
    Stundensatz       DECIMAL(5,2)
);
```

Aufgabe 3.1.a – CREATE TABLE Befehle mit notwendigen Constraints für das oben genannte Schema

Tabelle L:

```
CREATE TABLE L (
    LNr                INTEGER        PRIMARY KEY,
    Bezeichnung        VARCHAR(50)   NOT NULL,
    Komplexität        INTEGER        CHECK (Komplexität <= 10)
);
```

Aufgabe 3.1.a – CREATE TABLE Befehle mit notwendigen Constraints für das oben genannte Schema

Tabelle KTL:

```
CREATE TABLE KTL (  
    KNr          INTEGER          REFERENCES K(KNr),  
    TNr          INTEGER          REFERNCES T(TNr),  
    LNr          INTEGER          REFERENCES L(LNr),  
    Volumen     FLOAT,  
    Datum       CHAR(10),  
    Foreign Key KNr references K(KNr),  
    PRIMARY KEY (KNr, TNr, LNr)  
);
```

Aufgabe 3.2.b – Fügen Sie in die Kunden-Relation *K* eine neue Spalte *Branche* als variablen Text mit der Länge 30 mit dem Default-Wert „*Automobil*“ ein

```
ALTER TABLE K
```

```
ADD (Branche    VARCHAR(30)    DEFAULT („Automobil“));
```

Aufgabe 3.2.c – Löschen Sie die Spalte *TGröße* aus der Team-Relation *T*

```
ALTER TABLE T  
DROP (TGröße);
```

Aufgabe 3.2.d – Ändern Sie den Datentyp des Attributs *Volumen* aus der Relation *KTL* in eine ganze Zahl größer als 0

```
ALTER TABLE KTL
```

```
MODIFY (Volumen INTEGER);
```

```
ALTER TABLE KTL
```

```
ADD CONSTRAINT volumen_constraint CHECK (Volumen > 0);
```

Aufgabe 3.2.e – Löschen Sie nun alle Tabellen ohne die referenzielle Integrität zu verletzen

DROP TABLE *KTL*;

DROP TABLE *K*;

DROP TABLE *T*;

DROP TABLE *L*;

KTL muss als erstens gelöscht werden, dann ist die Reihenfolge egal

3. Wiederholung Relationale Algebra



- Datenbank umfasst eine Menge von Relationen
- Um Informationen aus diesen Relationen zu bekommen gibt es verschiedene Modell -> eines ist das Relationale Modell
- Es gibt verschiedene Operationen auf diese Relationen
- Operationen sind abgeschlossen (d.h. liefern als Ergebnis wieder eine Relation)
- In der Praxis wird dies in SQL (Structured Query Language) umgesetzt

- **Notation:**
 - R und S – Relationen
 - t – Tupel aus einer Relation
 - A und B – Attribute
 - F – Formel
 - A_i – Attribute beim Namen i referenziert
- **Vereinigung:** $R \cup S = \{t \mid t \in R \text{ oder } t \in S\}$
- **Differenz:** $R - S = \{t \mid t \in R \text{ und } t \notin S\}$
- **Kartesisches Produkt:**
$$R \times S = \{(a_1, \dots, a_r, a_{r+1}, \dots, a_{r+s}) \mid (a_1, \dots, a_r) \in R \text{ und } (a_{r+1}, \dots, a_{r+s}) \in S\}$$

- **Selektion:** $\sigma_F(R) = \{t \mid t \in R \wedge t \text{ erfüllt } F\}$

F besteht aus Konstanten, Attributen, Vergleichsoperatoren und Booleschen Operatoren

-> „Selektiere“ Zeilen nach einer Bedingung

- **Projektion:** $\Pi a_1, \dots, a_m(R) = \{t[a_1, \dots, a_m] \mid t \in R\}$

$t[a_1, \dots, a_m]$ bezeichnet ein Tupel aus R was nur die Attributwerte $a_1 \dots a_m$ enthält

-> „Projiziere“ nur bestimmte Spalten

Weiter wichtige Operationen

- **Durchschnitt:** $R \cap S = \{t \mid t \in R \text{ und } t \in S\}$
- **Theta-Join:** $R \bowtie_{A \theta B} S = \sigma_{A \theta B}(R \times S)$ ($\theta \in \{=, <, \leq, \geq, >, \neq\}$)
- **Equi-Join:** Gleich mit dem Theta-Join bis auf: $\theta \in \{=\}$
 - Gleichnamige Attribute werden behalten (gibt es dann doppelt)
- **Natural-Join:** $R \bowtie S$, Equi-Join bzgl. Aller gleichnamigen Attribute in R und S
 - Gleichnamige Attribute werden entfernt (existieren dann nur einmal im Ergebnis)
- **Quotient:** $R \div S = \{t \mid t \in \Pi_{R-S}(R) \wedge \{t\} \times S \subseteq R\}$
 - Ergebnis enthält alle linken Hälften von Tupeln aus R , die mit allen rechten Hälften von S kombiniert in R auftreten



LUDWIG-
MAXIMILIANS-
UNIVERSITÄT
MÜNCHEN



Finn Kapitza – Finn.Kapitza@campus.lmu.de