

Data Management

03 Data Models & Normalization

Matthias Boehm

Graz University of Technology, Austria
Computer Science and Biomedical Engineering
Institute of Interactive Systems and Data Science
BMK endowed chair for Data Management

Announcements/Org

■ #1 Video Recording

- Link in [TeachCenter](#) & [TUbe](#) (lectures will be public)



■ #2 Course Registrations SS20

- Data Management (lectures/exercises): **149/147**
- Databases (combined lectures/exercises): **68**

Total:
217

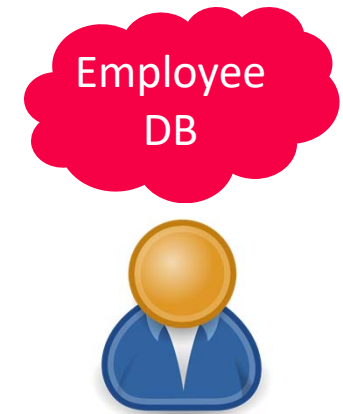
■ #3 Exercise 1

- Task Description published and discussed last week
- Deadline: **Nov 03** in TeachCenter

Q&A

Recap: DB Design Lifecycle Phases

- **#1 Requirements engineering**
 - Collect and analyze data and application requirements
 - Specification documents
- **#2 Conceptual Design** (lecture 02, exercise 1)
 - Model data semantics and structure, independent of logical data model
 - ER model / diagram
- **#3 Logical Design** (lecture 03, exercise 1)
 - Model data with implementation primitives of concrete data model
 - e.g., relational schema + integrity constraints, views, permissions, etc
- **#4 Physical Design** (lecture 07, exercise 2)
 - Model **user-level data organization** in a specific DBMS (and data model)
 - Account for deployment environment and performance requirements



Agenda

- Relational Data Model
- ER-Diagram to Relational Schema
- Normalization



[**Credit:** Alfons Kemper, André Eickler: Datenbanksysteme - Eine Einführung, 10. Auflage. De Gruyter Studium, de Gruyter Oldenbourg 2015, ISBN 978-3-11-044375-2, pp. 1-879]

Relational Data Model

Recap: History 1970/80s (relational)

Oracle, IBM DB2, Informix, Sybase
 → MS SQL



Ingres @ UC Berkeley
 (Stonebraker et al.,
Turing Award '14)

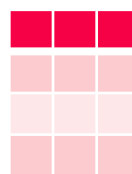
QUEL

System R @ IBM
 Research – Almaden
 (Jim Gray et al.,
Turing Award '98)

SEQUEL

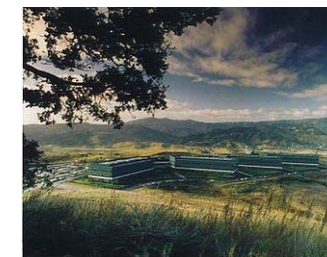
SQL Standard
 (SQL-86)

Tuple Calculus



Relational Algebra

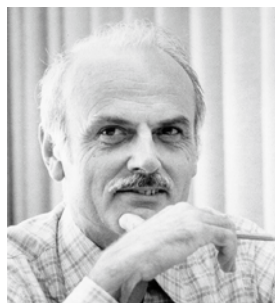
Relational Model



Recommended Reading

Goal: Data Independence
 (physical data independence)

- Ordering Dependence
- Indexing Dependence
- Access Path Depend.



Edgar F. “Ted” Codd @ IBM
 Research (**Turing Award '81**)

[E. F. Codd: A Relational Model of Data for Large Shared Data Banks. Comm. ACM 13(6), 1970]



Relations and Terminology

- Domain D (value domain): e.g., Set S, INT, Char[20]

- Relation R

- Relation schema RS: Set of k attributes $\{A_1, \dots, A_k\}$
- Attribute A_j : value domain $D_j = \text{dom}(A_j)$
- Relation: subset of the Cartesian product over all value domains D_j
 $R \subseteq D_1 \times D_2 \times \dots \times D_k, k \geq 1$

Attribute

	A1 INT	A2 INT	A3 BOOL
	3	7	T
	1	2	T
	3	4	F
Tuple	1	7	T

cardinality: 4
rank: 3

- Additional Terminology

- Tuple: row of k elements of a relation
- Cardinality of a relation: number of tuples in the relation
- Rank of a relation: number of attributes
- Semantics: Set := no duplicate tuples (in practice: Bag := duplicates allowed)
- Order of tuples and attributes is irrelevant

Relations and Terminology, cont.

■ Database Schema

- Set of relation (table) schemas

■ Database

- Set of actual relations, including data
- Database instance: current status of database

■ NULL

- Special **NULL** value for unknown or missing values
- Part of every domain, unless **NOT NULL constraint** specified
- Special semantics for specific operations, e.g., three-value Boolean logic

TRUE OR **NULL** → TRUE

FALSE OR **NULL** → **NULL**

TRUE AND **NULL** → **NULL**

FALSE AND **NULL** → FALSE

Example UniversityDB

Professors

<u>PID</u>	Title	Firstname	Lastname
1	Univ.-Prof. Dipl.-Inf. Dr.	Stefanie	Lindstaedt
6	Ass.Prof. Dipl.-Ing. Dr.techn.	Elisabeth	Lex
4	Assoc.Prof. Dipl.-Ing. Dr.techn.	Denis	Helic
7	Univ.-Prof. Dr.-Ing.	Matthias	Boehm

Courses

	<u>CID</u>	Title	ECTS
Summer/ Winter	INF.01014UF	Data Management (VO)	3
	INF.02018UF	Data Management (KU)	1
Winter	706.010	Databases	3
	706.520	Data Integration and Large-Scale Analysis	5
Summer	706.543	Architecture of Database Systems	5
	706.550	Architecture of Machine Learning Systems	5

Primary and Foreign Keys

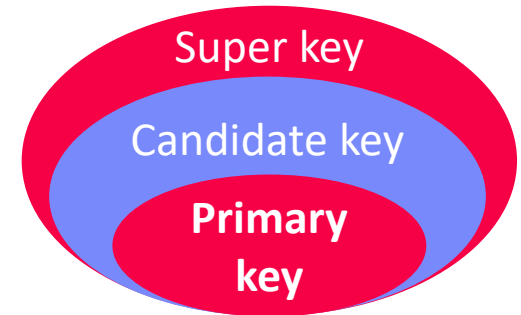
Primary Key X

- Minimal set of attributes X that uniquely identifies tuples in a relation R

E.g., PID=1 →

1	Univ.-Prof. Dipl.-Inf. Dr.	Stefanie	Lindstaedt
---	----------------------------	----------	------------

- #1 Unique:** $\forall t_i, t_j \in R: t_i[X] = t_j[X] \Rightarrow i = j$
- #2 Defined:** $\forall t_i \in R: t_i[X] \neq NULL$
- #3 Minimal:** no attribute can be removed wrt #1
- Super key: #1-2, **candidate key:** #1-3, **primary key:** pick one of the candidate keys



Foreign Key

- Reference of a primary key in another relation

1	Professors	<u>PID</u>	Title	First Name	Last Name
N	Courses	<u>CID</u>	Title	ECTS	PID

A blue arrow points from the 'PID' attribute in the 'Courses' table to the 'PID' attribute in the 'Professors' table, indicating a foreign key relationship.

Preview Next Lectures

- **Relational Algebra [Lecture 04]**
 - **Operands:** **relations** (variables for computing new values)
 - **Operators:** traditional set operations and specific relational operations (symbols representing the computation)

- **Structured Query Language (SQL) [Lecture 05]**
 - **Data Definition Language (DDL)** → Manipulate the database schema
 - **Data Manipulation Language (DML)** → Update and query database

Example CREATE TABLE

```
CREATE TABLE Professors (  
    PID INTEGER PRIMARY KEY,  
    Title VARCHAR(128),  
    Firstname VARCHAR(128),  
    Lastname VARCHAR(128)  
);
```

```
CREATE TABLE Courses (  
    CID INTEGER PRIMARY KEY,  
    Title VARCHAR(256),  
    ECTS INTEGER NOT NULL,  
    PID INTEGER  
    REFERENCES Professors  
);
```

Alternative for composite
primary key:

```
CREATE TABLE R (  
    ...,  
    PRIMARY KEY(A1, A2)  
);
```

Alternative for composite
foreign key:

```
CREATE TABLE S (  
    ...,  
    FOREIGN KEY(A1, A2)  
    REFERENCES R(A1, A2)  
);
```

Referential Integrity Constraints

Foreign Keys:

- Reference of a primary key in another relation
- Referential integrity:** FK need to reference existing tuples or NULL

Enforcing Referential Integrity

- #1 **Error** (default)
- #2 **Propagation** on request
 - E.g., for existential dependence
- #2 **Set NULL** on request
 - E.g., for independent entities

DELETE FROM Professors WHERE PID=7



```
CREATE TABLE Courses (...
  PID INTEGER REFERENCES Professors
  ON DELETE CASCADE);
```

```
CREATE TABLE Courses (...
  PID INTEGER REFERENCES Professors
  ON DELETE SET NULL);
```

Domain and Semantic Constraints

Domain/Semantic Constraints

- Value constraints of individual attributes (single and multi-column constraints)

- CHECK:** Value ranges or enumerated valid values

- Explicit naming via **CONSTRAINT**

```
CREATE TABLE Courses (
  CID INTEGER PRIMARY KEY,
  ECTS INTEGER
  CHECK (ECTS BETWEEN 1 AND 10)
);
```

(In PostgreSQL: no subqueries in CHECK constraints)

UNIQUE Constraints

- Enforce uniqueness of non-primary key attribute

NOT NULL Constraints

- Enforce known / existing values, potentially with DEFAULT

Triggers (in [lecture 05](#))

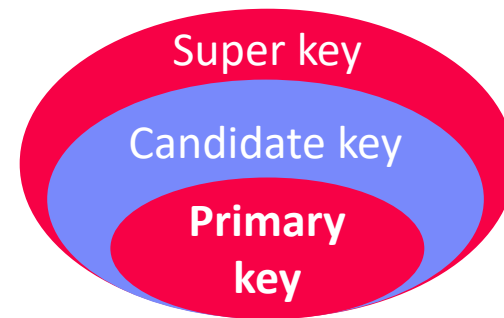
- Run stored procedures on insert/delete/update
- Full flexibility to specify arbitrary complex constraints

BREAK (and Test Yourself), cont.

- **Task: Assume Student(SID, Name, DoB), where SID is **unique** and **defined** (not null). List valid super and candidate keys (2/100 points)**

- **Super Keys**

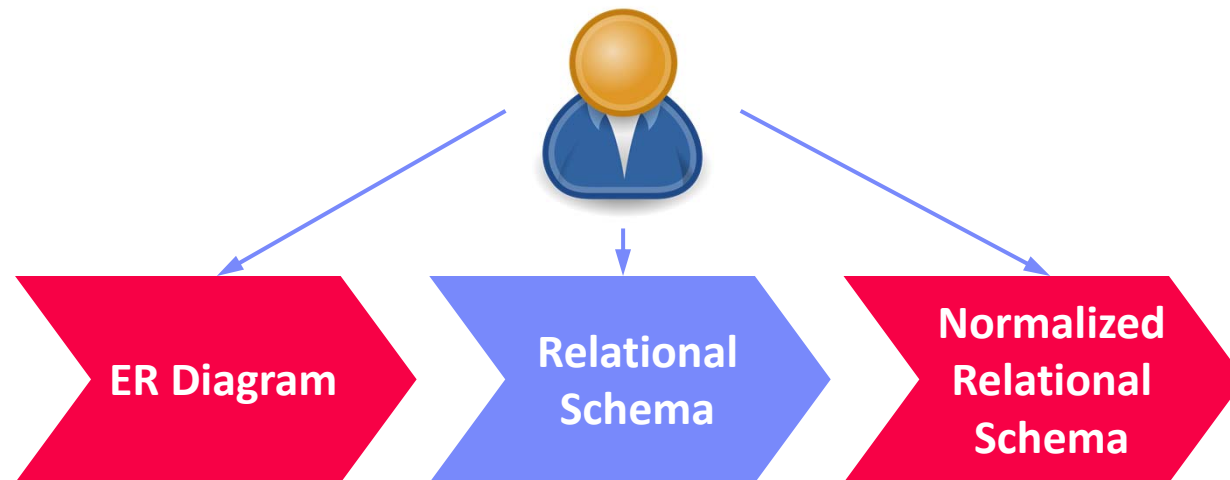
- (SID)
- (SID, Name), (SID, DoB)
- (SID, Name, DoB)



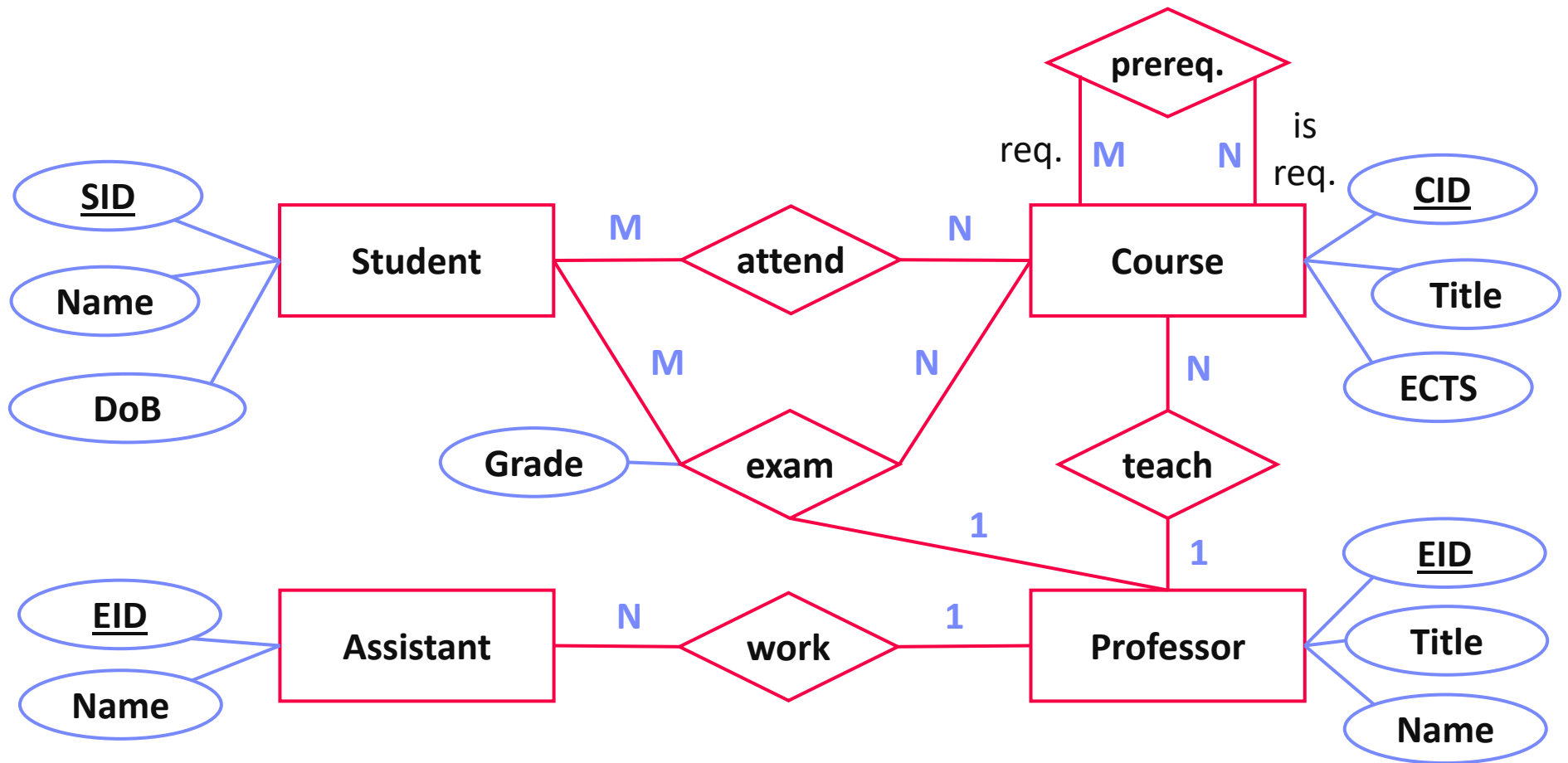
- **Candidate Keys**

- (SID)

ER-Diagram to Relational Schema



Recap: UniversityDB



Step 1: Mapping Entity Types

- Each entity type **directly maps to a relation**
 - Introduce surrogate (artificial) keys if needed
- **Examples**

Student

Students(
SID:INTEGER, Name:VARCHAR(128), Semester:INTEGER)

Course

Course(
CID:INTEGER, Title:VARCHAR(256), ECTS:INTEGER)

Professor

Professor(
EID:INTEGER, Title:VARCHAR(128), Name:VARCHAR(256))

Assistant

Assistant(
EID:INTEGER, Name:VARCHAR(256))

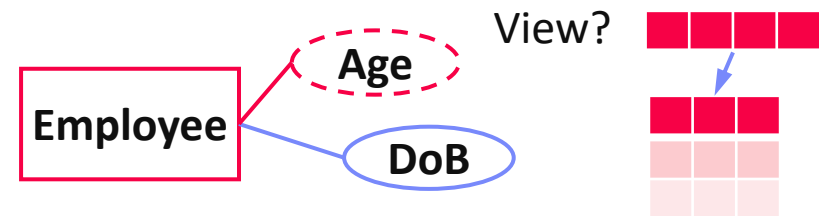
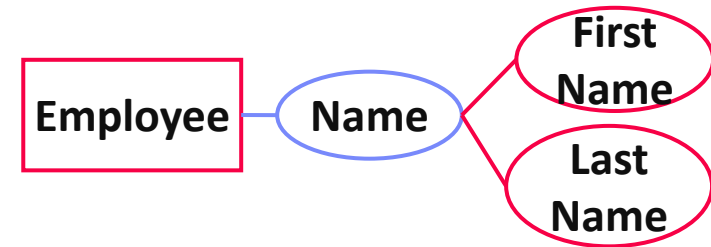
Step 2: Mapping Attributes

- **Atomic Attributes**
 - Direct mapping to attributes of relation
 - Choice of **data types and constraints**

- **Composite Attributes**
 - **Split into atomic attributes**,
 - Composite value, or
 - Object-relational data types

- **Derived Attributes**
 - Generated columns or **via views**

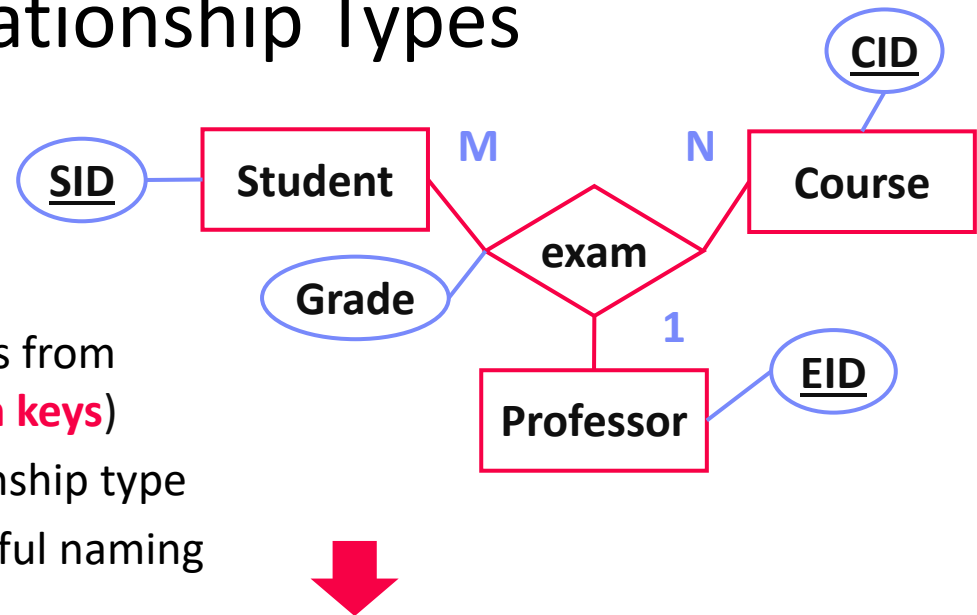
- **Multi-valued Attributes**
 - **Relation with FK** to originating relation



Step 3: Mapping Relationship Types

Generic Solution

- Map every relationship type to a **relation**
- Compose primary key** of keys from involved entity types (**foreign keys**)
- Append attributes** of relationship type
- Recursive relationships**: careful naming

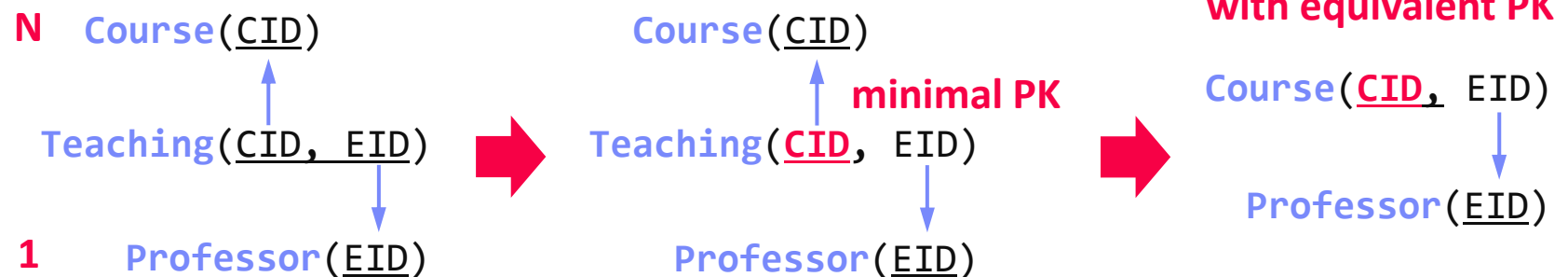


Exams:

<u>SID</u>	<u>CID</u>	<u>EID</u>	Grade
12345	706.010	7	1.0
12399	706.550	7	1.7
12399	706.010	7	1.3
12282	INF.01014UF	7	1.0

Step 4: Simplification

- Issue: **Unnecessary Relation per Relationship Type**
 → Simplify 1:1, 1:N, N:1 relationship types



- Examples
Fused Step 3-4
 - For **E1 – R – E2**
 - Modified Chen

Cardinality	Implementation
1:1	One relation E12 , PK from E1 or E2
C:1	One relation E12 (or two), PK from E2
1:M	Two relations E1 + E2 , E2 w/ FK to E1 (see Professor-Course above)
MC:MC	Three relations E1, R, E2 ; R w/ FKs to E1/E2

Step 5: Mapping Specializations

#1 Universal Relation

- One relation, **NULL** assigned for non-applicable attributes

→ **Employee**

#2 Object-oriented

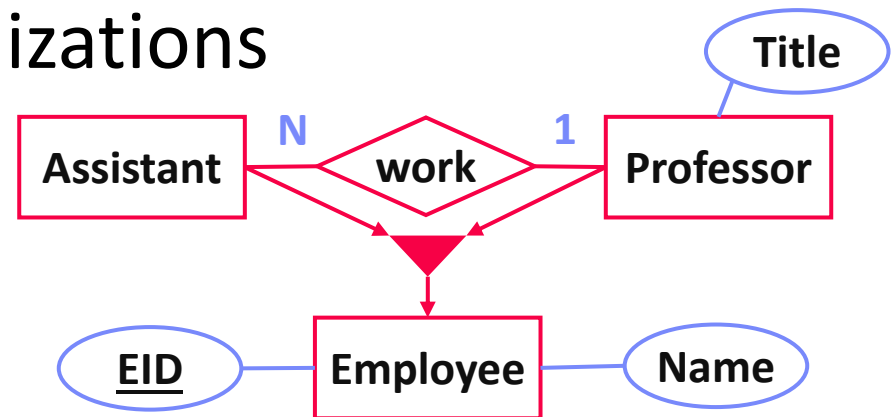
- One relation per specialized entity
- Horizontally partitioned

→ **Employee, Assistant, Professor**

#3 ER-oriented

- One relation per specialized entity
- Vertically partitioned

→ **Employee, Assistant, Professor**



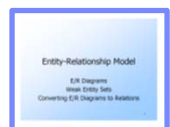
<u>EID</u>	Name
7	Matthias Boehm

<u>EID</u>	Title	Name
7	Univ.-Prof. Dr.-Ing.	Matthias Boehm

<u>EID</u>	Name
7	Matthias Boehm

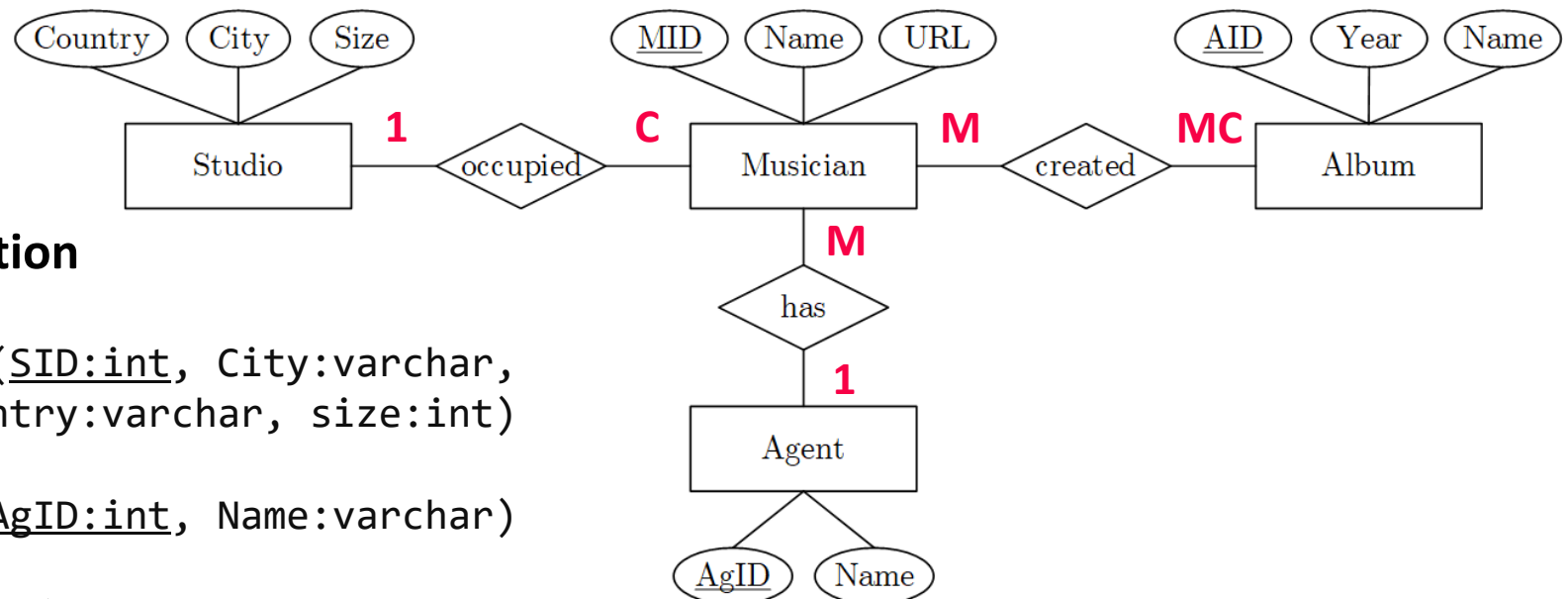
<u>EID</u>	Title
7	Univ.-Prof. Dr.-Ing.

[Jeffrey D. Ullman: CS145
Introduction to Databases - Entity-
Relationship Model, Stanford 2007]



BREAK (and Test Yourself)

- Task: Map the given ER diagram into a relational schema, including data types, primary keys, and foreign keys (9/100 points)



Solution

Studio(SID:int, City:varchar,
Country:varchar, size:int)

Agent(AgID:int, Name:varchar)

Musician(MID:int, Name:varchar,
URL:varchar, **SID^{FK}:int**, **AgID^{FK}:int**)

Album(AID:int, Year:int, Name:varchar)

Created(**MID^{FK}:int**, **AID^{FK}:int**)

5x relations

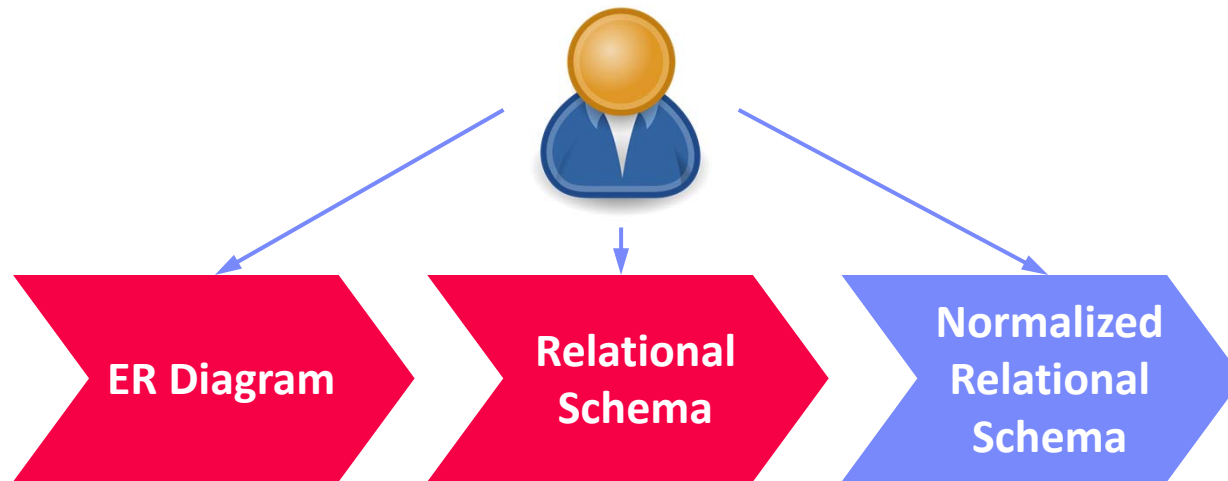
1x data types

1x primary keys

1x FKs in Musician

1x Composite PK in Created

Normalization



Motivation **Poor** Relational Schemas

ProfCourse (mixed entity types → **redundancy**)

EID	Name	CID	Title	ECTS
7	Boehm	INF.01014UF	Data Management (VO)	3
7	Boehm	INF.02018UF	Data Management (KU)	1
7	Boehm	706.010	Databases	3
7	Boehm	706.520	Data Integration and Large-Scale Analysis	5
7	Boehm	706.543	Architecture of Database Systems	5
7	Boehm	706.550	Architecture of Machine Learning Systems	5

- **Insert Anomaly:** How to insert a new lecture or prof?
- **Update Anomaly:** How to update “Boehm” → “Böhm”?
- **Delete Anomaly:** What if we delete all data management lectures?

➔ **Normalization:** Find good schema to avoid redundancy, ensure consistency, and prevent information loss

Overview Normalization

■ Normalization Process

- “[...] **reversible process** of replacing a given collection of relations [...] a progressively **simpler and more regular structure**”
- Principled approach of **improving the quality** (redundancy, inconsistencies)
- Input: DB-Schema and functional dependencies

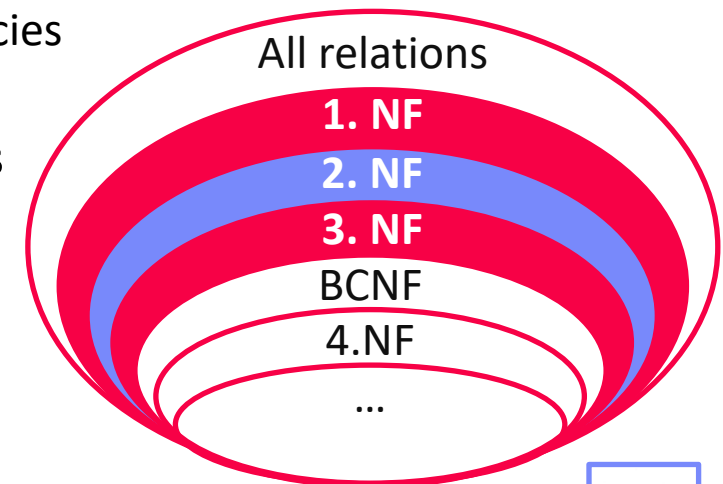
■ 1st Normal Form: no multi-valued attributes

■ 2nd Normal Form: all non-key attributes fully functional dependent on primary key

■ 3rd Normal Form: no dependencies among non-key attributes

■ Boyce-Codd Normal Form (BCNF)

■ 4th, 5th, 6th Normal Form



[E. F. Codd: Normalized Data Structure: A Brief Tutorial. SIGFIDET Workshop 1971]



[E. F. Codd: Further Normalization of the Data Base Relational Model. IBM Research Report, San Jose, California RJ909 (1971)]

Unnormalized Relation



Relation PartProject

<u>P#</u>	PDesc	Qty	Project (J#, JDesc, Mgr, Qty)			
203	CAM	30	12	Sorter	007	5
			73	Collator	086	7
206	COG	155	12	Sorter	007	33
			29	Punch	086	25
			36	Reader	111	16

Issues

- Column 'Project' is **not atomic, but set of tuples**
- Redundancy** across projects appearing in multiple parts

Solution:

- Create multiple tables with PK-FK relationships

1st Normal Form

Definition and Approach

- Relation is in 1NF if all its **attributes are atomic**

→ Split relations with 1:N and M:N relationships (lossless)

Example

Relation Part			FK		Relation Project		
<u>P#</u>	PDesc	Qty	<u>P#</u>	<u>J#</u>	JDesc	Mgr	Qty
203	CAM	30	203	12	Sorter	007	5
206	COG	155	203	73	Collator	086	7
			206	12	Sorter	007	33
			206	29	Punch	086	25
			206	36	Reader	111	16

Issues

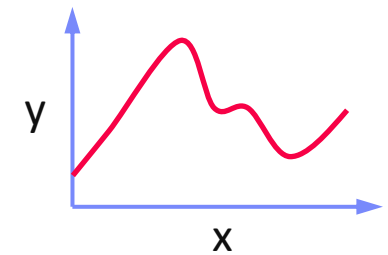
- Insert anomaly** (e.g., no project without parts)
- Update anomaly** (e.g., redundant updated Mgr)
- Delete anomaly** (e.g., project deleted on last part)

depend on J# depends on (J#,P#)

Background: Functional Dependency

- **Function $y = f(x)$**

- For deterministic functions f , the value x determines y (aka, y depends on x)



- **Functional Dependency (FD) $X \rightarrow Y$**

- X and Y are sets of attributes, Y functionally depends on X
- $X \rightarrow Y \Leftrightarrow \forall t_1, t_2 \in R: t_1[X] = t_2[X] \Rightarrow t_1[Y] = t_2[Y]$

- **Examples**

- $J\# \rightarrow \{JDesc, Mgr\}$
- $\{P\#, J\#\} \rightarrow Qty$

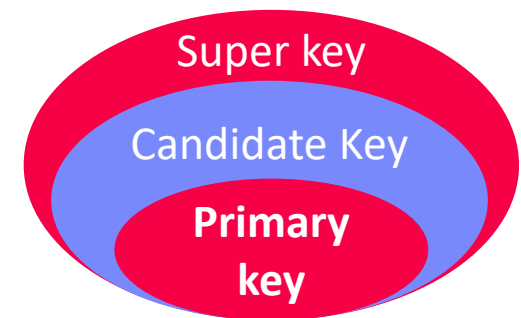
- **FDs derived from schema semantics not existing data**

P#	J#	JDesc	Mgr	Qty
203	12	Sorter	007	5
203	73	Collator	086	7
206	12	Sorter	007	33
206	29	Punch	086	25
206	36	Reader	111	16

Background: Functional Dependency, cont.

Full Functional Dependency

- Full functional dependency $X \rightarrow Y$ iff there is no proper subset $Z \subset X$ such that $Z \rightarrow Y$
- Candidate key:** $X \rightarrow$ relational schema (**minimal**)



Implied FDs via Armstrong Axioms

- Given a set **F** of FDs, the **closure F+** is the set of all implied FDs (which can be derived by the following axioms)
- Reflexivity: $X \supseteq Y \Rightarrow X \rightarrow Y$
- Augmentation: $X \rightarrow Y \Rightarrow XZ \rightarrow YZ$
- Transitivity: $(X \rightarrow Y) \wedge (Y \rightarrow Z) \Rightarrow X \rightarrow Z$

Composition

- Composition: $(X \rightarrow Y) \wedge (X \rightarrow Z) \Rightarrow X \rightarrow YZ$
- Decomposition: $X \rightarrow YZ \Rightarrow (X \rightarrow Y) \wedge (X \rightarrow Z)$
- Pseudo-Transitivity: $(X \rightarrow Y) \wedge (YW \rightarrow Z) \Rightarrow XW \rightarrow Z$

Example:

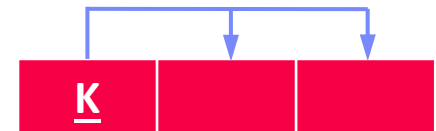
$(J\# \rightarrow JDesc, J\# \rightarrow Mgr)$
 $\rightarrow (J\# \rightarrow \{JDesc, Mgr\})$

2nd Normal Form

Definition and Approach

- Relation is in 2NF if it's **in 1NF** and every non-key attribute **fully functional dependent** from every candidate key

→ Split relations with 1:N and M:N relationships (lossless)



Example

Relation Part

<u>P#</u>	PDesc	Qty
203	CAM	30
206	COG	155

Relation PJ

<u>P#</u>	<u>J#</u>	Qty
203	12	5
203	73	7
206	12	33
206	29	25
206	36	16

Relation Project

<u>J#</u>	JDesc	Mgr
12	Sorter	007
73	Collator	086
29	Punch	086
36	Reader	111

Split PJ and Project because
 J# → {JDesc, Mgr}, **instead of** {P#,J#} → {JDesc, Mgr}

3rd Normal Form

■ Definition and Approach

- Relation is in 3NF if it's **in 2NF** and every non-key attribute is **non-transitively dependent** from every candidate key (→ no non-key dependencies)
- Split relations with 1:N and M:N relationships (lossless)
- Preserves all dependencies but might still contain anomalies (→ BCNF)

■ Example

NOT in 3NF

- **E# → D#**
- **D# → DMgr**
- **D# → CType**

Relation Employee

<u>E#</u>	JCode	D#	DMgr	CType
1	A	X	11	G
2	C	X	11	G
3	A	Y	12	N
4	B	X	11	G
5	B	Y	12	N
6	C	Y	12	N
7	A	Z	13	N
8	C	Z	13	N

3rd Normal Form, cont.



■ Example

Relation Employee			FK	Relation Department		
<u>E#</u>	JCode	D#	→	<u>D#</u>	DMgr	CType
1	A	X		X	11	G
2	C	X		Y	12	N
3	A	Y		Z	13	N
4	B	X				
5	B	Y				
6	C	Y				
7	A	Z				
8	C	Z				

➔ “Denormalization”:

Conscious creation of materialized views in non-3NF/2NF to improve performance (primarily for read-only DBs)

BREAK (and Test Yourself #2)

- **Task:** Assume the functional dependency **City→Country**. Bring your schema into 3NF and explain why it is in 3NF (prev. exam 10/100 points)

`Studio(SID:int, City:varchar, Country:varchar, size:int)`



- **Solution**

`Studio(SID:int, CNameFK:varchar, size:int)`

`Cities(CName:varchar, Country:varchar)`

- **1st Normal Form:** no multi-valued attributes
- **2nd Normal Form:** all non-key attributes fully functional dependent on PK
- **3rd Normal Form:** no dependencies among non-key attributes

Conclusions and Q&A

■ Summary

- Fundamentals of the relational data model + SQL DDL
- Mapping ER diagrams into relational schemas
- Relational normalization (1NF, 2NF, 3NF)

■ Exercise 1 Reminder

- All background to solve tasks 1.1-1.3 (25/25)
- Deadline: **Nov 03, 11.59pm** in TeachCenter, draft submissions are fine
- Make use of the news group and office hours (or **email**)

■ Next lectures

- **Holidays: Oct 26** Austrian National Day, **Nov 02** All Souls' Day,
- **04 Relational Algebra and Tuple Calculus** [Nov 09]
- **05 Query Languages (SQL)** [Nov 16], double? (**Nov/Dec room issues**)