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
BMVIT endowed chair for Data Management

Last update: Mar 16, 2019
Announcements/Org

- **#1 Video Recording**
  - Link in TeachCenter & T Ub e (lectures will be public)
  - Live Streaming Mo 4.10pm until Apr 19

- **#2 Reminder Communication**
  - Newsgroup: news://news.tugraz.at/tu-graz.lv.dbase
    (https://news.tugraz.at/cgi-bin/usenet/nntp.csh?tu-graz.lv.dbase)
  - Office hours: Mo 1pm-2pm (https://tugraz.webex.com/meet/m.boehm)

- **#3 Exercise 1 Published**
  - Exercise 1: Data Modeling published Mar 14
  - Submission in TeachCenter (deadline Mar 31, 11.59pm)
  - Changed late-day policy: 7+3 days
  - Dataset (understanding of the domain):
    https://github.com/tugraz-isdss/datasets/tree/master/dblp_publications
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

Relational Data Model
Recap: History 1970/80s (relational)

Relational Data Model

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]

SQL Standard (SQL-86)
Oracle, IBM DB2, Informix, Sybase → MS SQL

Relational Model

Relational Algebra

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

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

Tuple Calculus

SEQUEL

QUEL

Oracle, IBM DB2, Informix, Sybase → MS SQL

Recommended Reading
## Relations and Terminology

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

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

- **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**

### Relational Data Model

<table>
<thead>
<tr>
<th>A1 INT</th>
<th>A2 INT</th>
<th>A3 BOOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>7</td>
<td>T</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>T</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>F</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
<td>T</td>
</tr>
</tbody>
</table>

Cardinality: 4  
Rank: 3
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

\[
\begin{align*}
\text{TRUE OR NULL} & \rightarrow \text{TRUE} \\
\text{FALSE OR NULL} & \rightarrow \text{NULL} \\
\text{TRUE AND NULL} & \rightarrow \text{NULL} \\
\text{FALSE AND NULL} & \rightarrow \text{FALSE}
\end{align*}
\]
### Example UniversityDB

#### Professors

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

#### Courses

<table>
<thead>
<tr>
<th>CID</th>
<th>Title</th>
<th>ECTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>INF.01014UF</td>
<td>Data Management (VO)</td>
<td>3</td>
</tr>
<tr>
<td>INF.02018UF</td>
<td>Data Management (KU)</td>
<td>1</td>
</tr>
<tr>
<td>706.010</td>
<td>Databases</td>
<td>3</td>
</tr>
<tr>
<td>706.520</td>
<td>Data Integration and Large-Scale Analysis</td>
<td>5</td>
</tr>
<tr>
<td>706.543</td>
<td>Architecture of Database Systems</td>
<td>5</td>
</tr>
<tr>
<td>706.550</td>
<td>Architecture of Machine Learning Systems</td>
<td>5</td>
</tr>
</tbody>
</table>
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 | PID | Title | First Name | Last Name |
  | N | Courses | CID | Title | ECTS | PID |
## 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

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

Alternative for composite foreign key:
```sql
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`
  
  ```sql
  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
ER-Diagram to Relational Schema

ER Diagram → Relational Schema → Normalized Relational Schema
Recap: UniversityDB

ER-Diagram to Relational Schema

Student
- SID
- Name
- DoB

Course
- CID
- Title
- ECTS

Assistant
- EID
- Name

Professor
- EID
- Title
- Name

Attend
- M: Student
- N: Course

Examine
- M: Student
- N: Course

Teach
- 1: Professor
- N: Course

Grade
- N: Student

Prerequisite
- M: Course
- N: Course

Work
- N: Assistant
- 1: Professor
Step 1: Mapping Entity Types

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

**Examples**

- **Students**
  
<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>SID</td>
<td>INTEGER</td>
</tr>
<tr>
<td>Name</td>
<td>VARCHAR(128)</td>
</tr>
<tr>
<td>Semester</td>
<td>INTEGER</td>
</tr>
</tbody>
</table>

- **Course**

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>CID</td>
<td>INTEGER</td>
</tr>
<tr>
<td>Title</td>
<td>VARCHAR(256)</td>
</tr>
<tr>
<td>ECTS</td>
<td>INTEGER</td>
</tr>
</tbody>
</table>

- **Professor**

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>EID</td>
<td>INTEGER</td>
</tr>
<tr>
<td>Title</td>
<td>VARCHAR(128)</td>
</tr>
<tr>
<td>Name</td>
<td>VARCHAR(256)</td>
</tr>
</tbody>
</table>

- **Assistant**

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>EID</td>
<td>INTEGER</td>
</tr>
<tr>
<td>Name</td>
<td>VARCHAR(256)</td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th>SID</th>
<th>CID</th>
<th>EID</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>12345</td>
<td>706.010</td>
<td>7</td>
<td>1.0</td>
</tr>
<tr>
<td>12399</td>
<td>706.550</td>
<td>7</td>
<td>1.7</td>
</tr>
<tr>
<td>12399</td>
<td>706.010</td>
<td>7</td>
<td>1.3</td>
</tr>
<tr>
<td>12282</td>
<td>INF.01014UF</td>
<td>7</td>
<td>1.0</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Cardinality</th>
<th>Implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:1</td>
<td>One relation E12, PK from E1 or E2</td>
</tr>
<tr>
<td>C:1</td>
<td>One relation E12 (or two), PK from E2</td>
</tr>
<tr>
<td>1:M</td>
<td>Two relations E1 + E2, E2 w/ FK to E1 (see Professor-Course above)</td>
</tr>
<tr>
<td>MC:MC</td>
<td>Three relations E1, R, E2; R w/ FKs to E1/E2</td>
</tr>
</tbody>
</table>
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**

[Credit: Jeffrey D. Ullman: CS145 Introduction to Databases - Entity-Relationship Model]
BREAK (and Test Yourself #1)

- **Task:** Map the given ER diagram into a relational schema, including data types, primary keys, and foreign keys (prev. exam 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

ER Diagram → Relational Schema → Normalized Relational Schema
Motivation Poor Relational Schemas

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

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

- **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?

⇒ **Goal 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: Further Normalization of the Data Base Relational Model. IBM Research Report, San Jose, California RJ909 (1971)]
Unnormalized Relation

Relation PartProject

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

Issues

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

<table>
<thead>
<tr>
<th>Relation Part</th>
<th>FK</th>
<th>Relation Project</th>
</tr>
</thead>
<tbody>
<tr>
<td>P#</td>
<td>J#</td>
<td>JDesc</td>
</tr>
<tr>
<td>203</td>
<td>12</td>
<td>Sorter</td>
</tr>
<tr>
<td>203</td>
<td>73</td>
<td>Collator</td>
</tr>
<tr>
<td>206</td>
<td>12</td>
<td>Sorter</td>
</tr>
<tr>
<td>206</td>
<td>29</td>
<td>Punch</td>
</tr>
<tr>
<td>206</td>
<td>36</td>
<td>Reader</td>
</tr>
</tbody>
</table>

- **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)
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 $X$
  - $X \rightarrow Y \iff \forall t_1, t_2 \in R: t_1[X] = t_2[X] \Rightarrow t_1[Y] = t_2[Y]$

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

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

<table>
<thead>
<tr>
<th>P#</th>
<th>J#</th>
<th>JDesc</th>
<th>Mgr</th>
<th>Qty</th>
</tr>
</thead>
<tbody>
<tr>
<td>203</td>
<td>12</td>
<td>Sorter</td>
<td>007</td>
<td>5</td>
</tr>
<tr>
<td>203</td>
<td>73</td>
<td>Collator</td>
<td>086</td>
<td>7</td>
</tr>
<tr>
<td>206</td>
<td>12</td>
<td>Sorter</td>
<td>007</td>
<td>33</td>
</tr>
<tr>
<td>206</td>
<td>29</td>
<td>Punch</td>
<td>086</td>
<td>25</td>
</tr>
<tr>
<td>206</td>
<td>36</td>
<td>Reader</td>
<td>111</td>
<td>16</td>
</tr>
</tbody>
</table>
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 derive by the following axioms)
    - Reflexivity: $X \supseteq Y \Rightarrow X \rightarrow Y$
    - Augmentation: $X \rightarrow Y \Rightarrow XZ \rightarrow YZ$
    - Transitivity: $(X \rightarrow Y) \land (Y \rightarrow Z) \Rightarrow X \rightarrow Z$

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

Example:

$(J\# \rightarrow JDesc, J\# \rightarrow Mgr) \Rightarrow (J\# \rightarrow \{JDesc,Mgr\})$
2\textsuperscript{nd} 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
  - \( \Rightarrow \) Split relations with 1:N and M:N relationships (lossless)

- **Example**

<table>
<thead>
<tr>
<th>Relation Part</th>
<th>Relation PJ</th>
<th>Relation Project</th>
</tr>
</thead>
<tbody>
<tr>
<td>P#</td>
<td>PDesc</td>
<td>Qty</td>
</tr>
<tr>
<td>203</td>
<td>CAM</td>
<td>30</td>
</tr>
<tr>
<td>203</td>
<td>73</td>
<td>7</td>
</tr>
<tr>
<td>206</td>
<td>COG</td>
<td>155</td>
</tr>
<tr>
<td>206</td>
<td>29</td>
<td>25</td>
</tr>
</tbody>
</table>

Split PJ and Project because \( J\# \rightarrow \{JDesc, Mgr\}, \text{instead of } \{P\#,J\}\rightarrow \{JDesc, Mgr\} \)
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

<table>
<thead>
<tr>
<th></th>
<th>E#</th>
<th>JCode</th>
<th>D#</th>
<th>DMgr</th>
<th>CType</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>A</td>
<td>X</td>
<td>11</td>
<td>G</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>C</td>
<td>X</td>
<td>11</td>
<td>G</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>A</td>
<td>Y</td>
<td>12</td>
<td>N</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>B</td>
<td>X</td>
<td>11</td>
<td>G</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>B</td>
<td>Y</td>
<td>12</td>
<td>N</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>C</td>
<td>Y</td>
<td>12</td>
<td>N</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>A</td>
<td>Z</td>
<td>13</td>
<td>N</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>C</td>
<td>Z</td>
<td>13</td>
<td>N</td>
</tr>
</tbody>
</table>
3rd Normal Form, cont.

Example

Relation Employee

<table>
<thead>
<tr>
<th>E#</th>
<th>JCode</th>
<th>D#</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>X</td>
</tr>
<tr>
<td>2</td>
<td>C</td>
<td>X</td>
</tr>
<tr>
<td>3</td>
<td>A</td>
<td>Y</td>
</tr>
<tr>
<td>4</td>
<td>B</td>
<td>X</td>
</tr>
<tr>
<td>5</td>
<td>B</td>
<td>Y</td>
</tr>
<tr>
<td>6</td>
<td>C</td>
<td>Y</td>
</tr>
<tr>
<td>7</td>
<td>A</td>
<td>Z</td>
</tr>
<tr>
<td>8</td>
<td>C</td>
<td>Z</td>
</tr>
</tbody>
</table>

Relation Department

<table>
<thead>
<tr>
<th>D#</th>
<th>DMgr</th>
<th>CType</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>11</td>
<td>G</td>
</tr>
<tr>
<td>Y</td>
<td>12</td>
<td>N</td>
</tr>
<tr>
<td>Z</td>
<td>13</td>
<td>N</td>
</tr>
</tbody>
</table>

“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) + 1.4 (+5)
  - Deadline: Mar 31, 11.59pm in TeachCenter
  - Make use of the news group and office hours (or email)

- Next lectures
  - 04 Relational Algebra and Tuple Calculus [Mar 23]
  - 05 Query Languages (SQL) [Mar 30]
  - Easter Break: Apr 06 – Apr 18