Database Systems
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 18, 2019
Announcements/Org

- **#1 Video Recording**
  - *Starting today* through end of the semester
  - Available at [https://tube.tugraz.at/paella/ui/index.html](https://tube.tugraz.at/paella/ui/index.html)

- **#2 Reminder Newsgroup**
  - Ask general questions on newsgroup *not via email*, learn from other Qs
  - [news://news.tugraz.at/tu-graz.lv.dbase](news://news.tugraz.at/tu-graz.lv.dbase)

- **#3 Reminder Philosophy Study**
  - Experimentalphilosophische Studie zur *moralischen Intuition*
  - If interested, remain seated after the lecture

- **#4 New Office Hours**
  - Every Monday 1pm – 2pm (Inffeldgasse 13/V, PZ 205 014)

- **#5 Exercise Submission**
  - *Starting Mar 25* (deadline *Apr 02*)
Recap: Phases of the DB Design Lifecycle

- **#1 Requirements engineering**
  - Collect and analyze data and application requirements
  - Specification documents

- **#2 Conceptual Design** (last lecture)
  - Model data semantics and structure, independent of logical data model
  - ER model / diagram

- **#3 Logical Design** (this lecture)
  - Model data with implementation primitives of concrete data model
  - e.g., relational schema + integrity constraints, views, permissions, etc

- **#4 Physical Design**
  - 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)

**Recap: History 1970/80s (relational)**

- **Relational Data Model**
  - **SQL Standard** (SQL-86)
  - **SEQUEL**
  - **System R @ IBM Research – Almaden**
  - **Ingres @ UC Berkeley** (Stonebraker et al., Turing Award ‘14)
  - **Tuple Calculus**
  - **Relational Algebra**

**Relational Model**

- **Goal**: Data Independence (physical data independence)
  - Ordering Dependence
  - Indexing Dependence
  - Access Path Depend.

**Recommended Reading**

- 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

- **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 $D_j$
    $$ R \subseteq D_1 \times D_2 \times \ldots \times D_k, \ k \geq 1 $$

- **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**: \textbf{Set} := no duplicate tuples (in practice: \textbf{Bag} := duplicates allowed)
  - **Order of tuples and attributes is irrelevant**
Relations and Terminology, cont.

- **Database Schema**
  - Set of relation 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>First Name</th>
<th>Last Name</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. Dipl.-Wirt.-Inf. 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>Databases</td>
<td>4</td>
</tr>
<tr>
<td>706.004</td>
<td>Databases 1</td>
<td>3</td>
</tr>
<tr>
<td>706.550</td>
<td>Architecture of Machine Learning Systems</td>
<td>5</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>
</tbody>
</table>
Primary and Foreign Keys

- **Primary Key X**
  - Minimal set of attributes X that uniquely identifies tuples in a relation
  - E.g., PID=1

<table>
<thead>
<tr>
<th>PID</th>
<th>Title</th>
<th>First Name</th>
<th>Last Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Univ.-Prof. Dipl.-Inf. Dr.</td>
<td>Stefanie</td>
<td>Lindstaedt</td>
</tr>
</tbody>
</table>

- Unique: $\forall t_i, t_j \in R: t_i[X] = t_j[X] \Rightarrow i = j$
- Defined: $\forall t_i \in R: t_i[X] \neq NULL$
- Minimal $\Rightarrow$ candidate key (all three properties)

- **Foreign Key**
  - Reference of a primary key in another relation
  - Example

<table>
<thead>
<tr>
<th>1</th>
<th>Professors</th>
<th>PID</th>
<th>Title</th>
<th>First Name</th>
<th>Last Name</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>N</th>
<th>Courses</th>
<th>CID</th>
<th>Title</th>
<th>ECTS</th>
<th>PID</th>
</tr>
</thead>
</table>
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
Relational Data Model

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

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`

- **UNIQUE** Constraints
  - Enforce uniqueness of non-primary key attribute

- **NOT NULL** Constraints
  - Enforce known / existing values, potentially with DEFAULT

- **Triggers** (in future lectures)
  - Run stored procedures on insert/delete/update
  - Full flexibility to specify arbitrary complex constraints

```sql
CREATE TABLE Courses (  
CID INTEGER PRIMARY KEY,  
ECTS INTEGER  
CHECK (ECTS BETWEEN 1 AND 10)  
);  
(In PostgreSQL, no subqueries in CHECK constraints)
```
ER-Diagram to Relational Schema
Recap: UniversityDB

Student
- SID
- Name
- DoB

Course
- CID
- Title
- ECTS

Assistant
- EID
- Name

Professor
- EID
- Title
- Name

Attend
- M
- N

Grade

Exam
- M
- N

Teach
- 1

Prereq.
- M
- N

Work
- N
- 1
Step 1: Mapping Entity Types

- Each entity type directly maps to a relation

- Examples

  - **Students**
    - `SID: INTEGER`, `Name: VARCHAR(128)`, `Semester: INTEGER`

  - **Course**
    - `CID: INTEGER`, `Title: VARCHAR(256)`, `ECTS: INTEGER`

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

  - **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:**

<table>
<thead>
<tr>
<th>SID</th>
<th>CID</th>
<th>EID</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>12345</td>
<td>706.004</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>706.004</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**

<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, 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>M:M</td>
<td>Three relations E1, R, E2; R w/ FKS to E1/E2</td>
</tr>
</tbody>
</table>

**Examples**

- **Fused Step 1-4**
  - For E1 – R – E2
  - Modified Chen
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]
Normalization

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

ProfCourse (mixed entity types \(\rightarrow\) 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>Databases</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>Boehm</td>
<td>706.004</td>
<td>Databases 1</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>Boehm</td>
<td>706.550</td>
<td>Architecture of Machine Learning Systems</td>
<td>5</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>
</tbody>
</table>

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

\(\rightarrow\) **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

- **1\textsuperscript{st} Normal Form**: no multi-valued attributes
- **2\textsuperscript{nd} Normal Form**: all non-key attributes fully dependent on keys
- **3\textsuperscript{rd} Normal Form**: no dependencies among non-key attributes
  - Boyce-Codd Normal Form (BCNF)
- **4\textsuperscript{th}, 5\textsuperscript{th}, 6\textsuperscript{th} 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</th>
<th>Qty</th>
</tr>
</thead>
<tbody>
<tr>
<td>203</td>
<td>CAM</td>
<td>30</td>
<td>Sorter</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>007</td>
<td></td>
</tr>
<tr>
<td>73</td>
<td>Collator</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>086</td>
<td></td>
</tr>
<tr>
<td>206</td>
<td>COG</td>
<td>155</td>
<td>Sorter</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>007</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>Punch</td>
<td>25</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>086</td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>Reader</td>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>111</td>
<td></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

#### Relation Part

<table>
<thead>
<tr>
<th>P#</th>
<th>PDesc</th>
<th>Qty</th>
</tr>
</thead>
<tbody>
<tr>
<td>203</td>
<td>CAM</td>
<td>30</td>
</tr>
<tr>
<td>206</td>
<td>COG</td>
<td>155</td>
</tr>
</tbody>
</table>

#### Relation Project

<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>

#### 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 derived by the following axioms)
  - **Reflexivity:** $X \supseteq Y \Rightarrow X \rightarrow Y$
  - **Augmentation:** $X \rightarrow Y \Rightarrow XZ \rightarrow YZ$
  - **Transitivity:** $(X \rightarrow Y) \land (Y \rightarrow Z) \Rightarrow X \rightarrow Z$

- **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:

$(J\# \rightarrow \{JDesc,Mgr\}) = (J\# \rightarrow JDesc, J\# \rightarrow Mgr)$
2\textsuperscript{nd} Normal Form

\begin{itemize}
  \item **Definition and Approach**
    \begin{itemize}
      \item Relation is in 2NF if it’s \textbf{in 1NF} and every non-key attribute \textbf{fully functional dependent} from every candidate key
    \end{itemize}
  \item \textbf{Example}
  \end{itemize}

\begin{table}[h!]
\centering
\begin{tabular}{|c|c|c|}
\hline
P\# & PDesc & Qty \\
\hline
203 & CAM & 30 \\
206 & COG & 155 \\
\hline
\end{tabular}
\end{table}

\begin{table}[h!]
\centering
\begin{tabular}{|c|c|c|}
\hline
P\# & J\# & Qty \\
\hline
203 & 12 & 5 \\
203 & 73 & 7 \\
206 & 12 & 33 \\
206 & 29 & 25 \\
206 & 36 & 16 \\
\hline
\end{tabular}
\end{table}

\begin{table}[h!]
\centering
\begin{tabular}{|c|c|c|}
\hline
J\# & JDesc & Mgr \\
\hline
12 & Sorter & 007 \\
73 & Collator & 086 \\
29 & Punch & 086 \\
36 & Reader & 111 \\
\hline
\end{tabular}
\end{table}

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

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

Example

<table>
<thead>
<tr>
<th>Relation Employee</th>
<th>FK</th>
</tr>
</thead>
<tbody>
<tr>
<td>E#</td>
<td>JCode</td>
</tr>
<tr>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>2</td>
<td>C</td>
</tr>
<tr>
<td>3</td>
<td>A</td>
</tr>
<tr>
<td>4</td>
<td>B</td>
</tr>
<tr>
<td>5</td>
<td>B</td>
</tr>
<tr>
<td>6</td>
<td>C</td>
</tr>
<tr>
<td>7</td>
<td>A</td>
</tr>
<tr>
<td>8</td>
<td>C</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Relation Department</th>
</tr>
</thead>
<tbody>
<tr>
<td>D#</td>
</tr>
<tr>
<td>X</td>
</tr>
<tr>
<td>Y</td>
</tr>
<tr>
<td>Z</td>
</tr>
</tbody>
</table>

“Denormalization”: Conscious creation of materialized views in non-3NF/2NF to improve performance (primarily for read-only DBs)
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
  - Deadline: **Apr 02**
  - Submission details announced next week

- **Next lectures**
  - Mar 25: **04 Relational Algebra and Tuple Calculus**
  - Apr 01: **05 Query Languages (SQL)**