Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

CHAI
Humanities-Centered AI

Ralf Möller

Databases
Relational Schema Design Theory (Part 3)

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

CHAI
Humanities-Centered AI

Summary and outlook

1. **Introduction**
   - Models and abstraction
   - Database systems
   - Applications and data independence
2. **Entity-relationship model (ER model)**
3. **Relational data model (RDM)**
   - From ER to RDM
4. **Relational schema design theory**
   - Relational algebra as a query language
   - Functional dependencies
   - Normal forms
5. **Structured Query Language (SQL)**
   - Data definition
   - Query specification
   - Data manipulation
6. **Query processing**
   - Architecture
   - Indexing
   - Query plans, optimization
7. **Transactions**
   - Transaction processing, schedules, locks
   - Recovery
8. **Other types of databases**
   - Vector databases
   - Data stream query languages

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

CHAI
Humanities-Centered AI

What you should learn today

- Know even more about dependencies
- Divide (schemas) et impera (avoid redundancies)
- Normalize: “Key to happiness, the keys are”, Yoda says


Combating Redundancies
with Decomposition

Decomposition procedure

Start with $Z = \{\mathcal{R}\}$

As long as there are relational schemas $\mathcal{R}_i$ in Z that are not in BCNF:

- There is a non-trivial functional dependency $\alpha \longrightarrow \beta$ holding for a schema $\mathcal{R}_i$ such that
  - $\alpha \cap \beta = \emptyset$ and
  - $\alpha \longrightarrow \mathcal{R}_i$ does not hold
- Decompose $\mathcal{R}i$ into $\mathcal{R}_{i1} := \alpha \cup \beta$ und $\mathcal{R}_{i2} := \mathcal{R}_i \setminus \beta$
- Remove $\mathcal{R}_i$ from Z and add $\mathcal{R}_{i1}$ and $\mathcal{R}_{i2}$, i.e.,

  $$Z := (Z \setminus \{\mathcal{R}_i\}) \cup \{\mathcal{R}_{i1}\} \cup \{\mathcal{R}_{i2}\}$$

Example

Post code directory: {[Street, Place, FState, Post code]}

Functional dependencies:

- {Post code} ⟶ {Place, FState}
- {Street, Place, FState} ⟶ {Post code}

Consider decomposition

- Streets: {[Post code, Street]}
- Places: {[Post code, Place, FState]}

Decomposition

- Lossless, but…
- not dependency preserving (the second FD cannot be associated with any sub-relation)

Check of the second FD only with a join

Boyce-Codd normal form (BCNF): Summary

✓ Every relational schema can be decomposed into BCNF relations in a lossless way

- Sometimes, dependency preservation cannot be guaranteed

What to do when BCNF not possible?

Example:

$\mathcal{R}$ = {[A, B, C ,D]}
$F_{\mathcal{R}}$ = {A $\longrightarrow$ D, CD $\longrightarrow$ AB}

Candidate keys: {{A,C}, {C,D}}

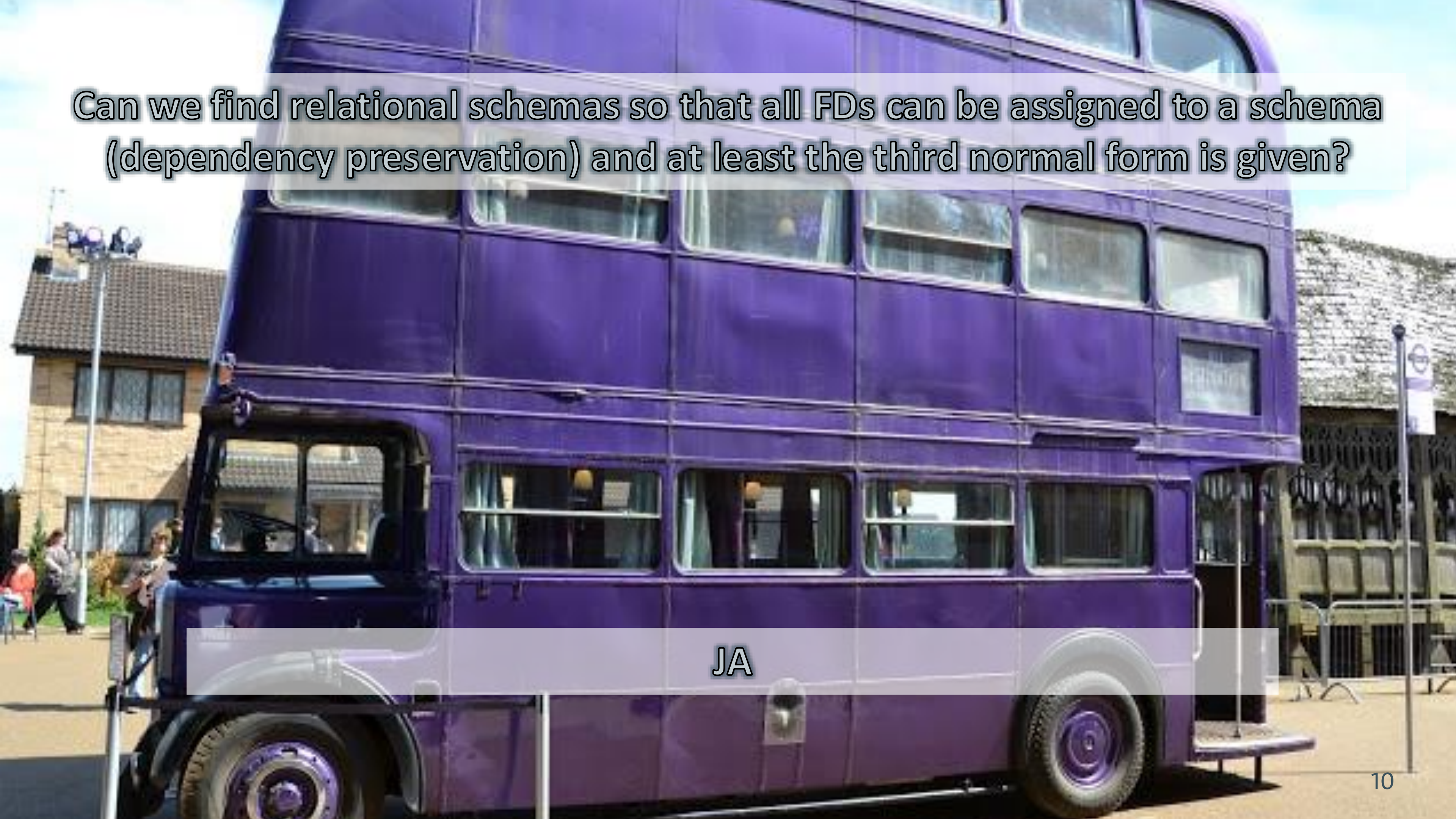
- Codd 71: A schema is „quite okay“ when there are no “transitive dependencies”
- FD A $\longrightarrow$ D associated with $\mathcal{R}$ might be tolerable (no “transitivity”) because D is part of a key
  - There is some redundancy, which is unavoidable
  - Check for A $\longrightarrow$ D required (besides uniqueness test for AC and CD), but unavoidable

Third normal form

A relational schema $\mathcal{R}$ is in 3rd normal form if additional to 2 NF, for every functional dependency associated with $\mathcal{R}$ of the form
$\alpha \rightarrow B$ with $\alpha \subseteq \mathcal{R}$ and $B \in \mathcal{R}$
at least one of the following conditions hold:

- $B \in \alpha$, i.e., the FD is trivial **or**
- α is a super key of $\mathcal{R}$ **or**
- the attribute B is contained in a candidate key of $\mathcal{R}$ (B is prime)

Note: The definition refers to every FD that holds for $\mathcal{R}$ → FD-Closure needed!



Can we find relational schemas so that all FDs can be assigned to a schema (dependency preservation) and at least the third normal form is given?

JA

Redundancies checked

Post code directory: {[Street, Place, FState, Post code]}

Functional dependencies:

- {Post code} ⟶ {Place, FState}
- {Street, Place, FState} ⟶ {Post code}
  - {Street, Place, FState} is a super key

Candidate keys

- { {Street, Place, FState}, {Street, Post code} }

3rd normal form given

- Place, FState, Post code are each contained in a candidate key

```
create table Post_code_directory
 (Street ...,
  Place ...,
  FState ...,
  Post_code ...,
  primary key (Street, Place, FState),
  unique (Street, Post_code)
  check(all x, all y:
          x.Post_code <> y.Post_code or
            (x.Place = y.Place and
             x.FState = y.FState))
);
```

Additional test required

But no join!

Synthesis of relational schemas

Professors: {[ProfID, Name, Type, Office, Place, Street, Pcode, Area code, FState, Inhabitants, StateGov]}
Canonical cover F_c:

- F1 {ProfID} ⟶ {Name, Type, Room, Place, Street, FState}
- F2 {Room} ⟶ {ProfID}
- F3 {Street, FState, Place} ⟶ {Post code}
- F4 {Place, FState} ⟶ {Inhabitants, Area code}
- F5 {FState} ⟶ {StateGov}
- F6 {PLZ} ⟶ {FState, Place}

Professors: {[ProfID, Name, Type, Room, Place, Street, FState]}
associated **FDs** = {F1, F2}, κ = {{ProfID}, {Room}}
Post code directory: {[Street, FState, Place, Post code]}
associated **FDs** = {F3, F6}, κ = {{Street, FState, Place}, {Street, Post code}}
Place directory: {[Place, FState, Inhabitants, Area code]}
associated **FDs** = {F4}, κ = {{Place, FState}}
Governments: {[FState, StateGov]}
associated **FDs** = {F5}, κ = {{FState}}

Synthesis procedure

For a given relational schema $\mathcal{R}$ with functional dependencies F a representation $\mathcal{R}_1$, ..., $\mathcal{R}_n$ is derived, such that the following three criteria are fulfilled:

- $\mathcal{R}_1$, ..., $\mathcal{R}_n$ is a lossless relational representation of $\mathcal{R}$
- The relational representation $\mathcal{R}_1$, ..., $\mathcal{R}_n$ is dependency preserving
- All relations $\mathcal{R}_1$, ..., $\mathcal{R}_n$ are in 3 NF

Computation of the canonical cover

1. For each FD $\alpha \rightarrow \beta \in F$ carry out a so-called left-reduction:
   - Check for all $A \in \alpha$ whether A is superfluous, i.e. check whether
     - β □ AttrClosure(F, $\alpha \setminus \{A\}$) holds
       If this is the case, replace $\alpha \rightarrow \beta$ by $(\alpha \setminus \{A\}) \rightarrow \beta$

2. For all remaining FDs apply a so-called right-reduction:
   - Check for all $B \in \beta$ whether
     - $B \in$ AttrClosure($(F \setminus \{\alpha \rightarrow \beta\}) * \{\alpha \rightarrow (\beta \setminus \{B\})\}$, α) holds
     - If this is the case, then B is superfluous on the righthand side and
     - can be eliminated, i.e., replace $\alpha \longrightarrow \beta$ by $\alpha \rightarrow (\beta \setminus \{B\})$

3. Remove FDs of the form $\alpha \rightarrow$ (might have been generated in step 2)

4. Combine FDs of the form $\alpha \longrightarrow \beta_1, \ldots, \alpha \longrightarrow \beta_n$
   such that $\alpha \longrightarrow (\beta_1 * \ldots * \beta_n)$ remains

Synthesis procedure

Determine canonical cover F_c for F.

For every FD $\alpha \rightarrow \beta \in F_c$:

- Create a relation schema $\mathcal{R}\alpha := \alpha \cup \beta$
- Associate with $\mathcal{R}\alpha$ the FDs $F\alpha := \{\alpha' \rightarrow \beta' \in F_c \mid \alpha' * \beta' \subseteq \mathcal{R}\alpha\}$

If one of the schemas created in the previous step contains a candidate key of $\mathcal{R}$ w.r.t. F_c, we are done. Otherwise select a candidate key $\kappa \subseteq \mathcal{R}$ and define the following schema:

- $\mathcal{R}\kappa := \kappa$
- $F\kappa :=$

Eliminate schemas $\mathcal{R}\alpha$, which are contained in other relation schemas $\mathcal{R}\alpha'$, i.e., for which

- $\mathcal{R}\alpha \subseteq \mathcal{R}\alpha'$ holds

Synthesis procedure produces relations in 3 NF

See David Maier, The Theory of Relational Databases, Computer Science Press, 1983

http://web.cecs.pdx.edu/~maier/TheoryBook/TRD.html

Claim: All $\mathcal{R}_i$ created from FD $\alpha \rightarrow \beta \in F_c$ are in 3 NF.
Argument: Assume that there was a $\gamma \rightarrow \gamma \square B \in F_c^+$ of $\mathcal{R}_i$, such that B is not prime and γ no super key. It must hold that $B \in \beta$. Because $\gamma \rightarrow B$ and $\gamma \square \alpha * \beta \setminus \{B\}$, the attribute would be redundant, but then $\alpha \rightarrow \beta \notin F_c$

Derivation of a relational schema for an application

- Represent domain knowledge as functional dependencies (FDs)
- Determine candidate keys
- Try the decomposition procedure
  - Decomposition could be automatized by “inference engine”
- In case all FDs can be associated with some schema:
  - Done, return table declarations with primary keys and uniqueness statements
- Run synthesis procedure
  - Return table declarations primary keys and uniqueness statements and extra tests included

Despite 3 NF protection spells:

Further attacks on data integrity are possible

Further restrictions from domain knowledge

- Inclusion dependencies: $\mathcal{R} \square \mathcal{S}$
  - Example: has-child vs. has-descendant
  - Multidimensional foreign keys required (see above)
- Multiple value dependencies
  - (Further) decomposition required

Multiple-value dependencies (MVDs): Example

Notation of MVDs in the example relation:

- {ProfID} $\twoheadrightarrow$ {Language} and
- {ProfID} $\twoheadrightarrow$ {ProgLanguage}

MVDs lead to redundancies and anomalies

$\alpha \twoheadrightarrow \beta$ holds if and only if

- for two tuples t1 and t2 with identical α–values
- there exists also tuples t3 and t4 with
  - $t3.\alpha = t4.\alpha = t1.\alpha = t2.\alpha$
  - $t3.\beta = t2.\beta$, $t4.\beta = t1.\beta$
  - $t4.\gamma = t2.\gamma$, $t3.\gamma = t1.\gamma$

Skills		
ProfID	Language	ProgLanguage
3002	Greek	C
3002	Latin	Pascal
3002	Greek	Pascal
3002	Latin	C
3005	German	Ada

"For two tuples with the same α–values the β–values can be exchanged, and the resulting tuples must also be in the relation"

- You can make a relation MVD conformant by inserting additional tuples
- Not possible by FDs!!

MVDs: Example

Skills		
ProfID	Language	ProgLanguage
3002	Greek	C
3002	Latin	Pascal
3002	Greek	Pascal
3002	Latin	C
3005	German	Ada

$\pi_{\text{ProfID, Language}}$

Languages	
ProfID	Language
3002	Greek
3002	Latin
3005	German

$\pi_{\text{ProfID, Language}}$

ProgLanguages	
ProfID	ProgLanguage
3002	C
3002	Pascal
3005	Ada

MVDs: Example

Skills

ProfID	Language	ProgLanguage
3002	Greek	C
3002	Latin	Pascal
3002	Greek	Pascal
3002	Latin	C
3005	German	Ada

$\bowtie$

Languages

ProfID	Language
3002	Greek
3002	Latin
3005	German

ProgLanguages

ProfID	ProgLanguage
3002	C
3002	Pascal
3005	Ada

4 NF

An MVD $\alpha \twoheadrightarrow \beta$ is trivial if and only if

- $\beta \subseteq \alpha$ or
- $\beta = R \setminus \{\alpha\}$

A relation R is in 4 NF if for all associated MVDs $\alpha \twoheadrightarrow \beta$ one of the following conditions holds:

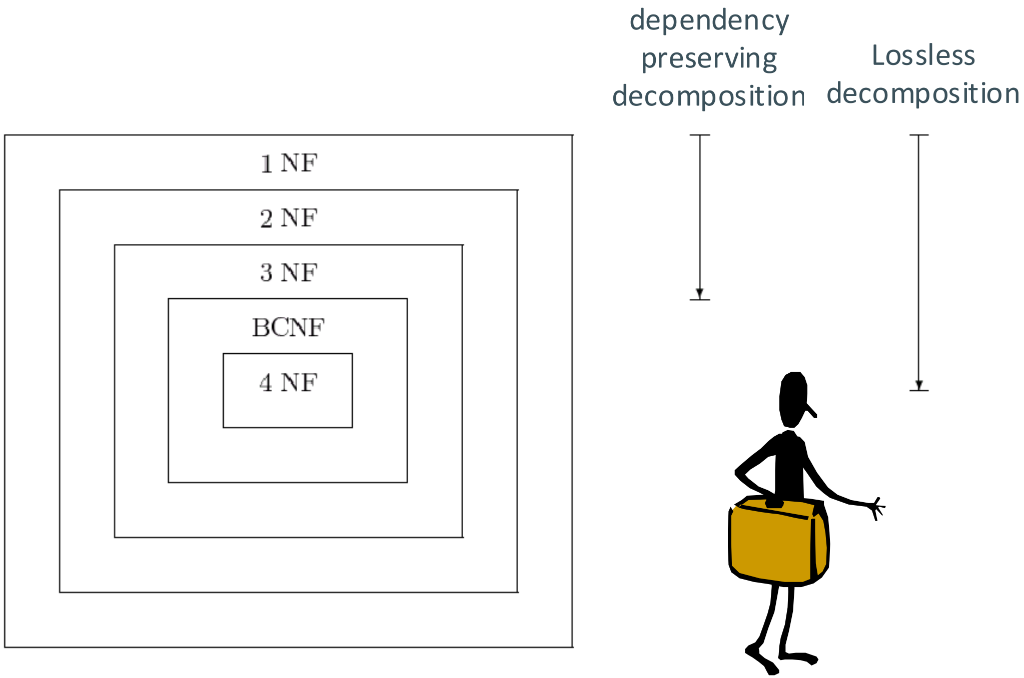
- The MVD is trivial **or**
- α is super key of R

Decomposition of a relation in 3 NF such that only trivial MVDs result is lossless

Summary

Losslessness is guanranteed for all decomposition procedures into all NFs

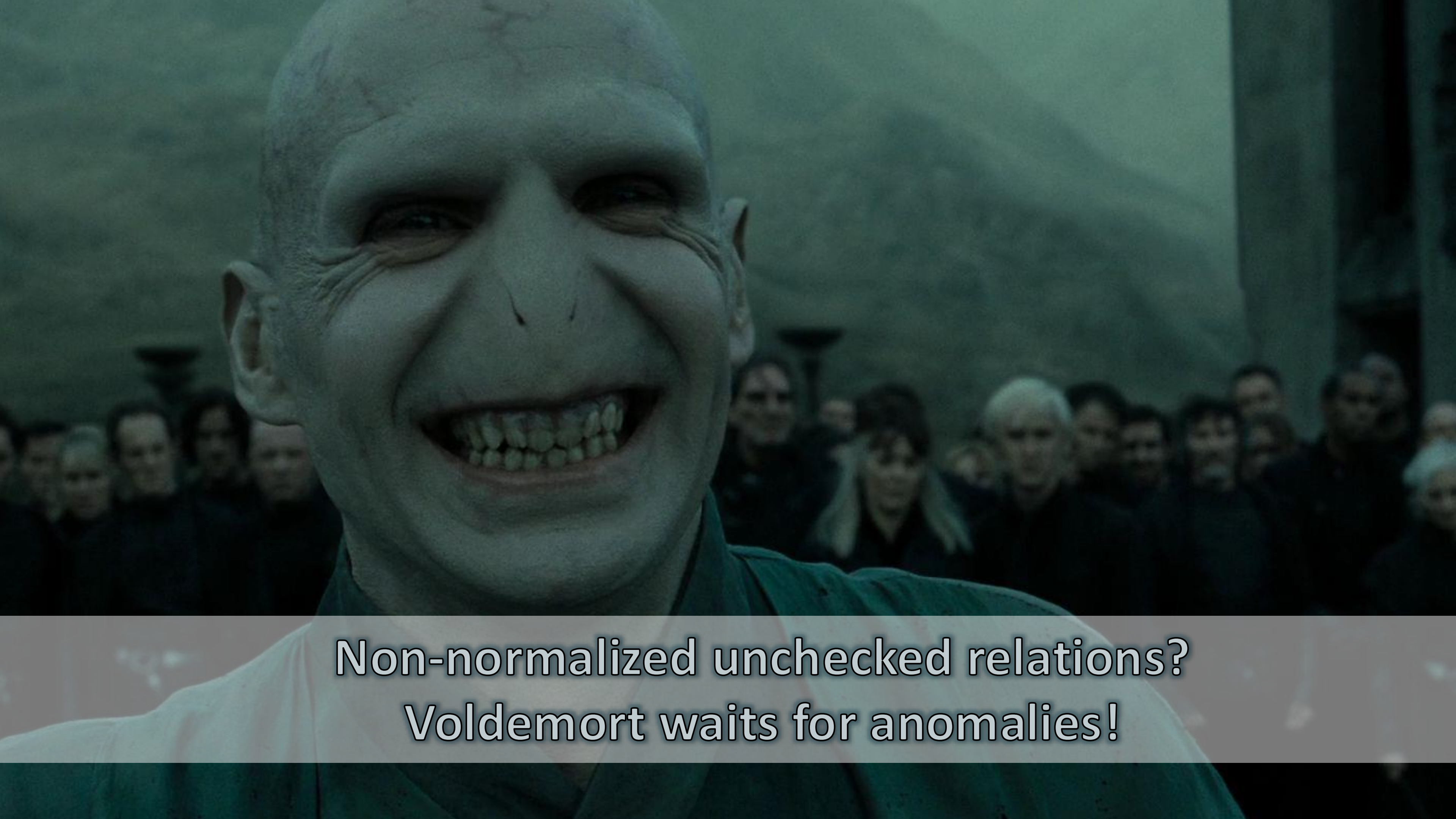
Dependency preservation can be guaranteed only up to 3 NF




Normalization is protective enchantment!

Summary and outlook

1. Introduction
   - Models and abstraction
   - Database systems
   - Applications and data independence
2. Entity-relationship model (ER model)
3. Relational data model (RDM)
   - From ER to RDM
4. Relational schema design theory
   - Relational algebra as a query language
   - Functional dependencies
   - Normal forms
5. **Structured Query Language (SQL)**
   - Data definition
   - Query specification
   - Data manipulation
6. **Query processing**
   - Architecture
   - Indexing
   - Query plans, optimization
7. **Transactions**
   - Transaction processing, schedules, locks
   - Recovery
8. **Other types of databases**
   - Vector databases
   - Data stream query languages


Non-normalized unchecked relations?
Voldemort waits for anomalies!