Relational Design Theory

Relational Design Theory

- A poorly designed schema leads to the following anomalies
  - Update anomalies: when changing *one* fact, *many* tuples have to be touched
  - Insertion anomalies: when inserting a tuple, many values cannot be specified (in the worst case not all key attribute values are known $\Rightarrow$ the tuple cannot be inserted)
  - Deletion anomalies: when deleting a tuple, more information is lost than intended

Example

- A financial advisor stores their data in a relational DBMS
- There are (B)rokers, their (O)ffices, (I)nvestors, (S)tocks, (D)ividends, and the (Q)uantity of a stock that an investor holds
- Everything is stored in one large relation:

Finance(B, O, I, S, D, Q)

Example(2)

Finance

B	O	I	S	D	Q
Müller	103	Schmidt	IBM	1.20	50
Müller	103	Schmidt	Merck	1.00	30
Müller	103	Schmidt	Coca Cola	0.00	100
Jung	214	Wilhelm	BASF	0.80	80
Jung	214	Wilhelm	Merck	1.00	140
Jung	214	Wilhelm	IBM	1.20	30
...	...	...	...	...	...

Example(3)

- What happens if Müller moves from 103 to 145?
- What happens if Meier starts out as a new broker but does not have any clients yet?
- What happens if Wilhelm sells all stocks and drops out?

A Closer Look

- The quality of a relational schema can be checked formally
- At first glance, the relation `Finance` contains a lot of redundant data:
  - 103 is Müller's office
  - IBM pays a dividend of 1.20 on each share
  - Wilhelm's advisor is Jung
- Where does this redundancy come from?
- Certain attribute values determine other attribute values, there are *functional dependencies*

Functional Dependencies

- E.g., there is a functional dependency (FD for short) between `Broker` and `Office`: Broker $\rightarrow$ Office
- Formal definition of an FD
  - α and β are attribute sets of a relational schema $\mathcal{R}$
  - There is an FD $\alpha \rightarrow \beta$ iff for *all* instances of $\mathcal{R}$: for all pairs of tuples $r, t \in R$, $r.\alpha = t.\alpha \Rightarrow r.\beta = t.\beta$ holds

Functional Dependencies(2)

- It is unrealistic to check all possible instances of $\mathcal{R}$ in practice
- FDs have to be derived from background knowledge about the application
- Which other FDs exist in `Finance`?

Functional Dependencies(3)

- $\mathcal{F}_{\mathcal{R}} = \{B \rightarrow O, S \rightarrow D, I \rightarrow B, IS \rightarrow Q\}$
- What do we do with these FDs now?
- We can *compute* a key for $\mathcal{R}$!

Computing Keys

- Properties of a key κ:
  1. $\kappa \subseteq \mathcal{R}$
  2. $\kappa \rightarrow \mathcal{R}$
  3. There is no $\kappa' \subset \kappa$ such that $\kappa' \rightarrow \mathcal{R}$
- Property 2. is called completeness, property 3. minimality
- κ is also called a *candidate key* (there can be more than one κ satisfying the properties above)
- One κ is chosen as the *primary key*
- If only properties 1. and 2. hold, κ is called a *superkey*

Computing Keys(2)

- Is `IS` a key of `Finance`?
- Property 1: $\surd$
- To check 2. and 3. we need further concepts of relational theory

Deriving Further FDs

- Further FDs can be derived from a set $\mathcal{F}$ of FDs
- $\mathcal{F}^+$, the set of all FDs derivable from $\mathcal{F}$, is called the closure of $\mathcal{F}$
- There are inference rules, the *Armstrong axioms*, for deriving further FDs

Armstrong Axioms

- α, β, and γ are subsets of $\mathcal{R}$
- Then the following sound and complete rules exist:
  - Reflexivity: $\beta \subseteq \alpha \Rightarrow \alpha \rightarrow \beta$
  - Augmentation: If $\alpha \rightarrow \beta$, then also $\alpha\gamma \rightarrow \beta\gamma$ ($\alpha\gamma$ stands for $\alpha \cup \gamma$)
  - Transitivity: $\alpha \rightarrow \beta$ and $\beta \rightarrow \gamma \Rightarrow \alpha \rightarrow \gamma$
- Using the Armstrong axioms, all valid FDs can be derived.

Armstrong Axioms(2)

- The following additional rules are not strictly necessary, but often useful:
  - Union rule: If $\alpha \rightarrow \beta$ and $\alpha \rightarrow \gamma$, then also $\alpha \rightarrow \beta\gamma$
  - Decomposition rule: If $\alpha \rightarrow \beta\gamma$, then also $\alpha \rightarrow \beta$ and $\alpha \rightarrow \gamma$
  - Pseudotransitivity rule: If $\alpha \rightarrow \beta$ and $\gamma\beta \rightarrow \delta$, then also $\alpha\gamma \rightarrow \delta$
- Since applying the Armstrong axioms to find keys is somewhat tedious, attribute closures are used in most cases.

Attribute Closure

- The *attribute closure* $AC(\alpha)$ of an attribute set α is the set of all attributes of $\mathcal{R}$ that are functionally dependent on α
- There is a simple algorithm for computing $AC(\alpha)$:

```
AC := α
while (AC still changes) do
    for each FD β → γ in F_R do
        if (β ⊆ AC)
        then AC := AC ∪ γ
```

Attribute Closure(2)

- $AC(IS) = BOISQD$
  $\Rightarrow IS$ is a superkey
- $AC(I) = IBO$
  $AC(S) = SD$
  $\Rightarrow IS$ is also a key

Checking Schema Quality

- There are *normal forms* (NF) that say something about the quality of a schema
- We consider 1NF, 2NF, 3NF, BCNF, 4NF
- There are higher ones, but they are mostly of theoretical interest

First Normal Form (1NF)

- A relational schema is in 1NF iff all attributes only take atomic values

Not in 1NF:

Parents

Mother	Father	Children
Marie	Hans	{Ines, David}
...	...	...

In 1NF:

Parents

Mother	Father	Child
Marie	Hans	Ines
Marie	Hans	David
...	...	...

Second Normal Form (2NF)

- A relation schema is in 2NF iff it is in 1NF and every non-key attribute (NKA) is fully functionally dependent on every key
- β is fully functionally dependent on α ($\alpha \overset{\bullet}{\rightarrow} \beta$) iff $\alpha \rightarrow \beta$ and there is no $\alpha' \subset \alpha$ such that $\alpha' \rightarrow \beta$
- `Finance`, e.g., is not in 2NF, since IS is a key, but $I \rightarrow B$
- A violation of 2NF indicates that different relationships (between entities) have been mixed in one relation

Third Normal Form (3NF)

- Even if 2NF is satisfied, the schema may still contain redundancies (due to transitive dependencies)
- A relation schema is in 3NF iff for every FD $\alpha \rightarrow \beta$ at least one of the following properties holds:
  - $\alpha \rightarrow \beta$ is trivial, i.e., $\beta \subseteq \alpha$
  - α is a superkey
  - Every attribute in β is contained in a key
- This avoids transitive dependencies (in which NKAs depend on the key via other NKAs)

Boyce-Codd Normal Form (BCNF)

- A further tightening is BCNF, here all attributes may only depend directly on the key
- A relation schema is in BCNF iff for every FD $\alpha \rightarrow \beta$ at least one of the following properties holds:
  - $\alpha \rightarrow \beta$ is trivial, i.e., $\beta \subseteq \alpha$
  - α is a superkey
- To describe further kinds of redundancy, we have to go beyond FDs

Multivalued Dependencies

Skills		
PersNr	Language	ProgLanguage
3002	English	C
3002	German	C
3002	English	Java
3002	German	Java
3005	English	C
3005	German	C

- There are no FDs in this schema, yet we have redundancy

Multivalued Dependencies(2)

- Someone who knows five programming languages and four languages needs 20 tuples to store this information!
- Here, mutually independent concepts are mixed together
- More compact storage is possible:

Skills	
PersNr	Language
3002	English
3002	German
3005	English
3005	German

Skills	
PersNr	ProgLanguage
3002	C
3002	Java
3005	C

Multivalued Dependencies(3)

- There are *multivalued dependencies* (MVDs) in this schema: PersNr $\twoheadrightarrow$ Language and PersNr $\twoheadrightarrow$ ProgLanguage
- Formal definition: $\alpha, \beta, \gamma \subseteq \mathcal{R}$ (with $\alpha \cup \beta \cup \gamma = \mathcal{R}$);
  $\alpha \twoheadrightarrow \beta$ holds iff for every instance R: for every pair of tuples t_1, t_2 in R with $t_1.\alpha = t_2.\alpha$ there exists a $t_3 \in R$ with $t_3.\alpha = t_1.\alpha$, $t_3.\beta = t_1.\beta$, and $t_3.\gamma = t_2.\gamma$
- Informally, this means that for all tuples with the same value for α, all β, γ combinations occur

Multivalued Dependencies(4)

- MVDs are a generalization of FDs, i.e., every FD is an MVD (but not necessarily vice versa)
- Similar to the Armstrong axioms, there are also derivation rules for MVDs (which will not be discussed in detail here)
- One important property of MVDs should be mentioned: $\alpha \twoheadrightarrow \beta \Rightarrow \alpha \twoheadrightarrow \gamma$ for $\gamma = \mathcal{R} - \alpha - \beta$

Fourth Normal Form

- The fourth normal form (4NF) extends the properties of BCNF to MVDs
- A relation schema is in 4NF iff for every MVD $\alpha \twoheadrightarrow \beta$ at least one of the following properties holds:
  - $\alpha \twoheadrightarrow \beta$ is trivial, i.e., $\beta \subseteq \alpha$ **OR** $\alpha \cup \beta = \mathcal{R}$
  - α is a superkey

- A higher NF includes all lower ones:
  $4NF \subset BCNF \subset 3NF \subset 2NF \subset 1NF$
- The higher the NF, the better the schema, i.e., the fewer redundancies
- What do we do if the quality of the current schema is not good enough?
- We transform the schema into a higher normal form, namely by means of decompositions

Decomposition of Relations

- A schema $\mathcal{R}$ is decomposed into the subschemas $\mathcal{R}_1, ..., \mathcal{R}_n$, with $\mathcal{R}_i \subset \mathcal{R}$ for $1 \leq i \leq n$
- A decomposition should have two properties:
  - Losslessness: the information contained in the original instance R must be reconstructible from the instances $R_1, ...R_n$ (for all possible instances R)
  - Dependency preservation: all FDs in $\mathcal{F}_\mathcal{R}$ should be preserved in the $\mathcal{F}_{\mathcal{R}_1}, ..., \mathcal{F}_{\mathcal{R}_n}$

Losslessness

- Decomposition of $\mathcal{R}$ into $\mathcal{R}_1$ and $\mathcal{R}_2$ (with $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2$ and $R_1 = \pi_{\mathcal{R}_1}(R)$, $R_2 = \pi_{\mathcal{R}_2}(R)$)
- The decomposition is lossless if for every possible instance R of $\mathcal{R}$:

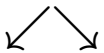
$$R = R_1 \bowtie R_2$$

- Sufficient condition for a lossless decomposition:
  - $(\mathcal{R}_1 \cap \mathcal{R}_2) \rightarrow \mathcal{R}_1 \in \mathcal{F}_{\mathcal{R}}^+$ or
  - $(\mathcal{R}_1 \cap \mathcal{R}_2) \rightarrow \mathcal{R}_2 \in \mathcal{F}_{\mathcal{R}}^+$

Example of Loss

Finance

B	O	I	S	D	Q
Müller	103	Schmidt	IBM	1.20	50
Müller	103	Schmidt	Merck	1.00	30
Jung	214	Wilhelm	Merck	1.00	140
Jung	214	Wilhelm	IBM	1.20	30



Finance1

B	O	S
Müller	103	IBM
Müller	103	Merck
Jung	214	Merck
Jung	214	IBM

Finance2

I	S	D	Q
Schmidt	IBM	1.20	50
Schmidt	Merck	1.00	30
Wilhelm	Merck	1.00	140
Wilhelm	IBM	1.20	30

Example of Loss(2)

Finance1 $\bowtie$ Finance2

Finance

B	O	I	S	D	Q
Müller	103	Schmidt	IBM	1.20	50
Müller	103	Schmidt	Merck	1.00	30
Jung	214	Wilhelm	Merck	1.00	140
Jung	214	Wilhelm	IBM	1.20	30
Müller	103	Wilhelm	IBM	1.20	50
Müller	103	Wilhelm	Merck	1.00	30
Jung	214	Schmidt	Merck	1.00	140
Jung	214	Schmidt	IBM	1.20	30

Dependency Preservation

- Decomposition of $\mathcal{R}$ into $\mathcal{R}_1, ..., \mathcal{R}_n$
- The decomposition is dependency preserving if
  $\mathcal{F}_{\mathcal{R}} \equiv \left(\mathcal{F}_{\mathcal{R}_1} \cup ... \cup \mathcal{F}_{\mathcal{R}_n}\right)_+$ or
  $\mathcal{F}^+_{\mathcal{R}} = \left(\mathcal{F}_{\mathcal{R}_1} \cup ... \cup \mathcal{F}_{\mathcal{R}_n}\right)^+$
- In the previous example, the FD $I \rightarrow B$ is lost, so the decomposition is also not dependency preserving

Decomposition Algorithms

- Decomposition is usually not done and checked laboriously by hand every time
- Most important algorithm: 3NF synthesis algorithm
  - Decomposes a schema into 3NF losslessly and dependency preserving
  - However, it requires a redundancy-free set of FDs as input (canonical cover)

Canonical Cover

- $\mathcal{F}_c$ is called a canonical cover of $\mathcal{F}$ if the following three criteria are met:
  - $\mathcal{F}_c \equiv \mathcal{F}$, i.e., $\mathcal{F}_c^+ = \mathcal{F}^+$
  - In $\mathcal{F}_c$ there are no FDs $\alpha \rightarrow \beta$ in which α or β contain superfluous attributes.
  - Every left-hand side of an FD in $\mathcal{F}_c$ is unique. This can be achieved by successive application of the union rule:
    $\alpha \rightarrow \beta, \alpha \rightarrow \gamma \Rightarrow \alpha \rightarrow \beta\gamma$

Canonical Cover(2)

- When are there no superfluous attributes?
  - $\forall A \in \alpha$:
    $(\mathcal{F}_c - (\alpha \rightarrow \beta) \cup ((\alpha - A) \rightarrow \beta)) \not\equiv \mathcal{F}_c$
  - $\forall B \in \beta$:
    $(\mathcal{F}_c - (\alpha \rightarrow \beta) \cup (\alpha \rightarrow (\beta - B))) \not\equiv \mathcal{F}_c$
- Superfluous attributes can be removed by left and right reduction

Left Reduction

- Perform left reduction for every FD $\alpha \rightarrow \beta \in \mathcal{F}$
- For all $A \in \alpha$, check whether A is superfluous, i.e., whether $\beta \subseteq AttrClosure\ (\mathcal{F}, \alpha - A)$
- If so, replace $\alpha \rightarrow \beta$ by $(\alpha - A) \rightarrow \beta$.

Right Reduction

- Perform right reduction for every (remaining) FD $\alpha \rightarrow \beta$
- For all $B \in \beta$, check whether $B \in AttrClosure\ (\mathcal{F} - (\alpha \rightarrow \beta) \cup (\alpha \rightarrow (\beta - B)), \alpha)$
- If so, B is superfluous on the right-hand side and can be eliminated, i.e., $\alpha \rightarrow \beta$ is replaced by $\alpha \rightarrow (\beta - B)$.

Only Two More Steps

- Remove the FDs of the form $\alpha \to \varnothing$ that may have been created in the second step.
- Use the union rule to combine FDs of the form $\alpha \to \beta_1, \ldots, \alpha \to \beta_n$, so that $\alpha \to \beta_1 \cup \cdots \cup \beta_n$ remains.
- This completes the computation of the canonical cover
- Depending on the order in which the FDs are processed, different covers may result (but all of them are redundancy-free)

3NF Synthesis Algorithm

- First step: for every FD $\alpha \to \beta$ in $\mathcal{F}_c$ form a subschema $\mathcal{R}_\alpha = \alpha \cup \beta$, assign to $\mathcal{R}_\alpha$ the FDs $\mathcal{F}_\alpha := \{\alpha' \to \beta' \in \mathcal{F}_c \mid \alpha' \cup \beta' \subseteq \mathcal{R}_\alpha\}$
- Second step: add a schema $\mathcal{R}_\kappa$ with a candidate key
- Third step: eliminate redundant schemas, i.e., if $\mathcal{R}_i \subseteq \mathcal{R}_j$, discard $\mathcal{R}_i$

Example

- Applying the 3NF synthesis algorithm to `Finance` with $\mathcal{F}_{\mathcal{R}_c} = \{B \to O, S \to D, I \to B, IS \to Q\}$
- Result:
  - $\mathcal{R}_B(B, O)$
  - $\mathcal{R}_I(I, B)$
  - $\mathcal{R}_S(S, D)$
  - $\mathcal{R}_{IS}(I, S, Q)$
- What happened to the anomalies from the beginning of the chapter? They are gone!

Decomposition into BCNF and 4NF

- There are further decomposition algorithms for BCNF and 4NF
- Problem: there are schemas that cannot be decomposed into BCNF or 4NF in a dependency-preserving way
- Usually, one settles for 3NF; another reason: higher normal forms favor update operations over query operations

Decomposition into BCNF

- Start with $Z = \{\mathcal{R}\}$
- As long as there is still an $\mathcal{R}_i \in Z$ that is not in BCNF:
  - Find an FD $(\alpha \rightarrow \beta) \in F^+$ with
    - $\alpha \cup \beta \subseteq \mathcal{R}_i$
    - $\alpha \cap \beta = \varnothing$
    - $\alpha \rightarrow \mathcal{R}_i \notin F^+$
  - Decompose $\mathcal{R}_i$ into $\mathcal{R}_{i_1} := \alpha \cup \beta$ and $\mathcal{R}_{i_2} := \mathcal{R}_i - \beta$
  - Remove $\mathcal{R}_i$ from Z and insert $\mathcal{R}_{i_1}$ and $\mathcal{R}_{i_2}$, i.e., $Z := (Z - \{\mathcal{R}_i\}) \cup \{\mathcal{R}_{i_1}\} \cup \{\mathcal{R}_{i_2}\}$

Decomposition into 4NF

- Start with $Z = \{\mathcal{R}\}$
- As long as there is still an $\mathcal{R}_i \in Z$ that is not in 4NF:
  - Find an MVD $\alpha \twoheadrightarrow \beta \in \mathcal{F}^+$ with
    - $\alpha \cup \beta \subset \mathcal{R}_i$
    - $\alpha \cap \beta = \varnothing$
    - $\alpha \rightarrow \mathcal{R}_i \notin \mathcal{F}^+$
  - Decompose $\mathcal{R}_i$ into $\mathcal{R}_{i_1} := \alpha \cup \beta$ and $\mathcal{R}_{i_2} := \mathcal{R}_i - \beta$
  - Remove $\mathcal{R}_i$ from Z and insert $\mathcal{R}_{i_1}$ and $\mathcal{R}_{i_2}$, i.e., $Z := (Z - \{\mathcal{R}_i\}) \cup \{\mathcal{R}_{i_1}\} \cup \{\mathcal{R}_{i_2}\}$

Summary

- Normal forms can be used to determine the quality of a schema
- Losslessness is guaranteed for all decomposition algorithms
- Dependency preservation can only be guaranteed for decompositions up to the third normal form