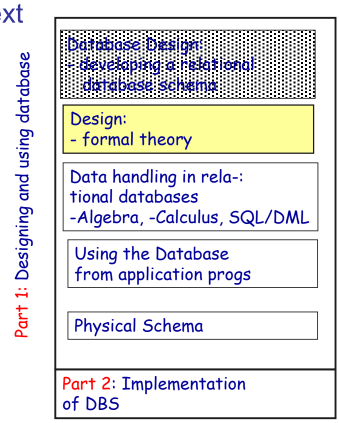
5 Normalization: Quality of relational designs

Lit: Kemper/Eickler: chap 6; Garcia-Molina/Ullman/Widom: chap 3.4 ff.; Elmasr/Navathe: chap 14

Lausen: Datenbanken - Grundlagen und XML-Technologien

Context



5.1.1 Design quality

- What is a "good" conceptual model ?
  - Usually many alternatives
  - No clear criteria for comparison guidelines
  - wanted: formal methods for comparing designs
- Informal guidelines
  - Avoid redundancies:
    (mId , format) in one relation?

    Movie (mId, title, format, director, ...)
    repeats the title, director information for each format;
    if tape with the format exists
- Avoid to model more than one obejct from reality in one entity / relation

  Data about director in Movie relation?

  Movie (mId, title, director, birthdate, livesInCity,...)

Bad (!) design variants for Movie table!

5.1.2 Update Anomalies

- Basic idea: constraints must be modeled explicitly

  e.g. a tape is loaned by zero or one customer,
  each person has a unique birthday

  Movie (mId, title, director, birthdate, livesInCity,...)
- Redundancies may cause "anomalies"
  - Deletion of a row may delete all data about a different object
  - Update of an attribute may cause update on many rows
  - Insertion may be difficult / impossible, since data are missing

Examples….

"update anomaly" = deletion, update, insertion anomaly

Update anomalies: Examples

```
CREATE TABLE Experiment (
   id SERIAL PRIMARY KEY,
   responsible_Person VARCHAR(40),
   institute          VARCHAR (30),
   phone              INT,
   purpose            VARCHAR(100),
   start              TIMESTAMP,
   endTime            TIMESTAMP,
   result             INT)

DELETE  FROM Experiment WHERE  result < 10
```

Consequence: data on experimentator might be lost

Update anomalies

- Deletion anomaly: example

```
Delete  (43, 'Amistad', 'Spielberg', 12.10.47, 'LA',..)
from  table Movie (mId, title, director,birthdate,livesInCity ,…
```

effect: data about director are lost if this is the only movie with this director ('Spielberg') → deletion anomaly

- Update anomaly: example

```
update Movie set livesInCity = 'SF'
  where director = "Sp.."
```

all those movie tuples having director = 'Spielberg' have to be changed (update anomaly)

- what is an insertion anomaly?

5.1.3 Functions and Functional dependencies

- Important formal concept: Functions
  - Used to formalize integrity constraints on relationships

    Rents: Tape -> Customer is a (partial) function
  - General approach:
    find functions among attributes in Relation R

Examples:

```
Experiment (id, responsible_Person, institute, phone,...result)
```

{responsible_Person} -> {institute} is a function

```
Movie ( mId, title, director, birthdate, livesInCity,…)
```

{director} -> {birthdate, livesInCity} is a function

which means: if (43, 'Amistad', 'Spielberg', 12.10.47, 'LA',
is a row of Movie,
(43, 'Amistad', 'Spielberg', 1.7.49, 'LA',..) is not a valid row

Functional Dependencies (FD)

- Keys and Functional dependencies
  - Property of a key : at most one row for each value k
  - Let the key of Relation R be composed of attributes $K=\{a_1,...,a_k\}$
    Then the attributes $\Sigma(R) \setminus \{a_1,...,a_k\}$ are functionally dependent from K

    This means:
    - There is a function which maps keys to values of attributes
    - Function is represented by table
    - Table may be changed, but functional property is time invariant

- Primary key: one of the candidate keys
- Prime attributes: attributes belonging to a candidate key

Functional Dependencies

- Generalization: Functional dependency between non-key attributes or sets of attributes

  Example:
  - Movie (mId, title, format, director, birthdate, livesInCity,…)
    director -> birthday
    i.e. a particular director has one and only one birthday
  - No FD between director and title :
    A particular person may have directed many films (titles)

- Functional dependencies:
  are constraints (invariants) of the application domain

Functional Dependency: Definition

Functional Dependencies (FDs)

Let $A = \Sigma(R)^* = \{a,b,c,...a_i,..\}$ be the attribute set of a relation E, e, e' tuples of R,
let $X, Y \subseteq A$

Y is functionally dependent on X (written: $X \rightarrow Y$)
iff
$(\forall x_i \in X)\ e.x_i = e'.x_i \Rightarrow (\forall y_i \in Y)\ e.y_i = e'.y_i$

- Important: invariants are independent of the particular database state

- They must hold at all times,
  i.e. they restrict the valid states of the database.

* $\Sigma(R)$: Attribute set of relation R

Functional Dependencies: Example

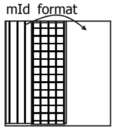
- "Video rental" from above:

  "format" as an attribute of "Movie"-table
  (stupid! just an example)
- "mId" is not a key any more !

  There may be one film available with two different formats.

Movie
mId: ... format: cha.. year: date title: String category ... format: String ....

{mId} functionally determines all attributes of movie but one (format).



5.1.3 Properties of Functional Dependencies

- Trivial functional dependency

  $X \subseteq Y \Rightarrow Y \rightarrow X$

  Trivial: if values of attributes $y_i \in Y$ are given, then the values of attributes in every subset of Y
- Augmentation

  $Z \subseteq A=\Sigma(R),\quad X \rightarrow Y \Rightarrow XZ \rightarrow YZ$

- Transitivity

  $X,Y,Z \subseteq A=\Sigma(R),\ X \rightarrow Y,\ Y \rightarrow Z \Rightarrow X \rightarrow Z$

Notation XY -> Z means $X \cup Y$ -> Z

Armstrong inference rules

Given a set of FDs, find all implied FD's

A sound, complete, minimal set (Armstrong axioms):	
$Y \subseteq X \Rightarrow X \rightarrow Y$	(I: inclusion)
$\{X \rightarrow Y\ ,\ Y \rightarrow Z\} \Rightarrow X \rightarrow Z$	(T: transitivity)
$\{X \rightarrow Y\} \Rightarrow XZ \rightarrow YZ$	(A: augmentation)

sound:
only logically implied FDs are produced by the inference rules
complete:
every logically implied FD will be produced by finite many inferences

Means:
- given a set F of FDs. Every FD implied by F will be produced by a finite number of inferences I, T or A
- No FD will be inferred, which is not implied by F

We will use this result to calculate normal forms

Functional Dependencies and keys

Non-key attributes functionally dependent on part of the key

{mId, format} is the key, but e.g. {mId} → {title} holds

- Bad: key properties are checked by the DB system, other functional dependencies are NOT

e.g. more than one title or director for ONE mId cannot be prevented by the DBS

Different kinds of FDs of a relation :

1. Partial dependencies on one of the candidate keys
   {mId} -> {title} , since key is {mId, format}
2. Dependencies among non-key attributes
   {director} -> {birthday}
3. Dependencies among attributes of different candidate keys