course:
Database Systems (NDBI025)
SS2017/18

lecture 8:

Relational design – normal forms

doc. RNDr. Tomáš Skopal, Ph.D.
RNDr. Michal Kopecký, Ph.D.

Department of Software Engineering, Faculty of Mathematics and Physics, Charles University in Prague

Today's lecture outline

- motivation
  - data redundancy and update/insertion/deletion anomalies
- functional dependencies
  - Armstrong's axioms
  - attribute and dependency closures
- normal forms
  - 3NF
  - BCNF

Motaivation

- result of relational design
  – a set of relation schemas
- problems
  - data redundancy
    - unnecessary multiple storage of the same data – increased space cost
  - insert/update/deletion anomalies
    - insertions and updates must preserve redundant data storage
    - deletion might cause loss of some data
  - solution
    - relation schema normalization

Example of "abnormal" schema

EmpId	Name	Position	Hourly salary	Hours completed
1	John Goodman	accountant	200	50
2	Paul Newman	salesman	500	30
3	David Houseman	salesman	500	45
4	Brad Pittman	accountant	200	70
5	Peter Hitman	accountant	200	66
6	Adam Batman	lecturer	300	10

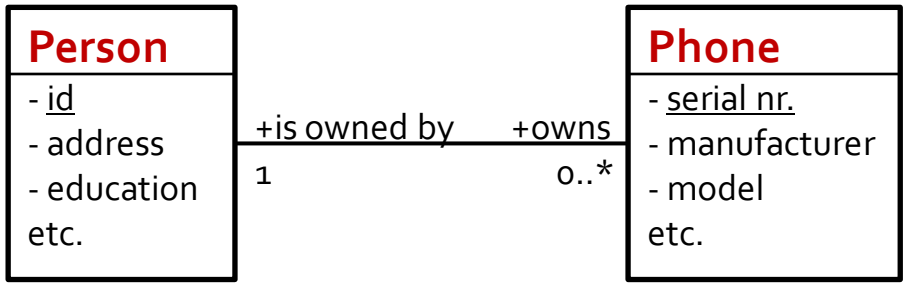
1) From functional analysis we know that position determines hourly salary.
However, hourly salary data is stored multiple times – redundancy.

2) If we delete employee 6, we lose the information on lecturer salary.

3) If we change the accountant hourly salary, we must do that in three places.

How could this even happen?

- simply
  - during "manual" design of relation schemas
  - badly designed conceptual model
    - e.g., too many attributes in a class



the UML diagram results in 2 tables:

Person(**id**, address, education, ...)
Mobil(**serial nr.**, manufacturer, model, ..., **id**)

How could this even happen?

Serial nr.	Manufacturer	Model	Made in	Certificate
13458	Nokia	Lumia	Finland	EU, USA
34654	Nokia	Lumia	Finland	EU, USA
65454	Nokia	Lumia	Finland	EU, USA
45464	Apple	iPhone 4S	USA	EU, USA
64654	Samsung	Galaxy S2	Taiwan	Asia, USA
65787	Samsung	Galaxy S2	Taiwan	Asia, USA

Redundancy in attributes Manufacturer, Model, Made in, Certificate

What happened?
Class Phone includes also other classes – Manufacturer, Model, ...

How to fix it?
Two options 1) fix the UML model (design of more classes)
2) alter the already created schemas (see next)

Functional dependencies

- attribute-based integrity constraints defined by the user (e.g., DB application designer)
- kind of alternative to conceptual modeling (ER and UML invented much later)
- functional dependency (FD) $X \rightarrow Y$ over schema R(A)
  - mapping $f_i : X_i \rightarrow Y_i$, where $X_i, Y_i \subseteq A$ (kde i = 1..number of FDs in R(A))
  - *n*-tuple from X_i **determines** *m*-tuple from Y_i
  - *m*-tuple from Y_i **is determined by (is dependent on)** *n*-tuple from X_i

Functional dependencies

- simply, for $X \rightarrow Y$, values in X **together determine** the values in Y
- if $X \rightarrow Y$ and $Y \rightarrow X$, then X and Y are **functionally equivalent**
  - could be denoted as $X \leftrightarrow Y$
- if $X \rightarrow a$, where $a \in A$, then $X \rightarrow a$ is **elementary FD**
- FDs represent a generalization of the key concept (identifier)
  - the key is a special case, see next slides

Example – wrong interpretation

EmpId	Name	Position	Hourly salary	Hours completed
1	John Goodman	accountant	200	50
2	Paul Newman	salesman	500	30
3	David Houseman	salesman	500	45
4	Brad Pittman	accountant	200	70
5	Peter Hitman	accountant	200	66
6	Adam Batman	lecturer	300	10

One might **observe** from the **data**, that:

Position → Hourly salary and also **Hourly salary → Position**
EmpId → *everything*
Hours completed → *everything*
Name → *everything*

(**but that is nonsense w.r.t. the natural meaning of the attributes**)

Example – wrong interpretation

EmpId	Name	Position	Hourly salary	Hours completed
1	John Goodman	accountant	200	50
2	Paul Newman	salesman	500	30
3	David Houseman	salesman	500	45
4	Brad Pittman	accountant	200	70
5	Peter Hitman	accountant	200	66
6	Adam Batman	lecturer	300	10
7	Fred Whitman	advisor	300	70
8	Peter Hitman	salesman	500	55

newly inserted records

Position → Hourly salary
EmpId → *everything*

~~**Hourly salary → Position**~~
~~**Hours completed → *everything***~~
~~**Name → *everything***~~

Example – correct interpretation

- at first, after the data analysis the FDs are set "forever", **limiting** the content of the tables
  - e.g. **Position → Hourly salary**
    EmpId → *everything*
  - insertion of the last row is **not allowed** as it violates both the FDs

EmpId	Name	Position	Hourly salary	Hours completed
1	John Goodman	accountant	200	50
2	Paul Newman	salesman	500	30
3	David Houseman	salesman	500	45
4	Brad Pittman	accountant	200	70
5	Peter Hitman	accountant	200	66
5	Adam Batman	salesman	300	23

Armstrong's axioms

Let us have R(A,F). Let $X, Y, Z \subseteq A$ and F is the set of FDs

1) if $Y \subseteq X$, then $X \rightarrow Y$ (trivial FD, really axiom)

2) if $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$ (transitivity, rule)

3) if $X \rightarrow Y$ and $X \rightarrow Z$, then $X \rightarrow YZ$ (composition, rule)

4) if $X \rightarrow YZ$, then $X \rightarrow Y$ and $X \rightarrow Z$ (decomposition, rule)

Armstrong's axioms

Armstrong's axioms:

- **are correct (sound)**
  - what is derived from F is valid for any instance from R
- **are complete**
  - all FDs valid in all instances in R (w.r.t. F) can be derived using the axioms
- **1,2,3 (trivial, transitivity, composition) are independent**
  - removal of any axiom 1,2,3 violates the completeness (decomposition could be derived from trivial FD and transitivity)

Proof follows from the FD definition
(from the properties of mapping, respectively).

Example – deriving FDs

R(A,F)

A = {a,b,c,d,e,f}

F = {ab $\rightarrow$ c, ac $\rightarrow$ d, cd $\rightarrow$ ed, e $\rightarrow$ f}

We could derive, e.g.,:

ab $\rightarrow$ a	(trivial)
ab $\rightarrow$ ac	(composition with ab $\rightarrow$ c)
ab $\rightarrow$ d	(transitivity with ac $\rightarrow$ d)
ab $\rightarrow$ cd	(composition with ab $\rightarrow$ c)
ab $\rightarrow$ ed	(transitivity with cd $\rightarrow$ ed)
ab $\rightarrow$ e	(decomposition)
ab $\rightarrow$ f	(transitivity)

Example – deriving the decomposition rule

R(A,F)

A = {a,b,c}

F = {$a \rightarrow bc$}

Deriving:

$a \rightarrow bc$	(assumption)
$bc \rightarrow b$	(trivial FD)
$bc \rightarrow c$	(trivial FD)
$a \rightarrow b$	(transitivity)
$a \rightarrow c$	(transitivity)

i.e., $\mathbf{a \rightarrow bc \Rightarrow a \rightarrow b \wedge a \rightarrow c}$

Closure of set of FDs

- **closure F^+** of FDs set F (FD closure) is the set of all FDs derivable from F using the Armstrong's axioms
  - generally exponential size w.r.t. |F|
- a FD f is **redundant** in F if
  $(F - \{f\})^+ = F^+$, i.e., f can be derived from the rest of F

Example – closure of set of FDs

R(A,F), A = {a,b,c,d}, F = {ab $\rightarrow$ c, cd $\rightarrow$ b, ad $\rightarrow$ c}

F^+ =

{a $\rightarrow$ a, b $\rightarrow$ b, c $\rightarrow$ c,
ab $\rightarrow$ a, ab $\rightarrow$ b, ab $\rightarrow$ c,
cd $\rightarrow$ b, cd $\rightarrow$ c, cd $\rightarrow$ d,
ad $\rightarrow$ a, ad $\rightarrow$ c, ad $\rightarrow$ d,
abd $\rightarrow$ a, abd $\rightarrow$ b, abd $\rightarrow$ c, abd $\rightarrow$ d,
abd $\rightarrow$ abcd,…}

Cover

- **cover** of a set **F** is any set of FDs **G** such that $\mathbf{F^+=G^+}$
- **canonic cover** = cover consisting of **elementary FDs**
  (decompositions are performed to obtain singleton sets on the right side)
- **non-redundant cover** of F
  = cover of F after removing all redundant FDs
  - note the order of removing FDs matters – a redundant FD could become non-redundant FD after removing another redundant FD
    - implies that there may exist multiple non-redundant covers of F

Example – cover

R1(A,F), R2(A,G),

A = {a,b,c,d},

F = {a → c, b → ac, d → abc},

G = {a → c, b → a, d → b}

For the check of $G^+ = F^+$ we do not have to establish the whole closures, it is sufficient to derive F from G, and vice versa, i.e.,

F′ = {a → c, b → a, d → b} – decomposition

G′ = {a → c, b → ac, d → abc } – transitivity and composition

$\Rightarrow G^+ = F^+$

Schemas R1 and R2 are equivalent because G is cover of F, while they share the attribute set A.

Moreover, G is **minimal cover**, while F is not (for minimal cover see next slides).

Example – redundant FDs

R1(A,F), R2(A,G),
A = {a,b,c,d},
F = {a → c, b → a, b → c, d → a, d → b, d → c}

FDs b → c, d → a, d → c are redundant

after their removal F^+ is not changed, i.e., they could be derived from the remaining FDs

b → c derived using transitivity a → c, b → a
d → a derived using transitivity d → b, b → a
d → c derived using transitivity d → b, b → a, a → c

Attribute closure, key

- **attribute closure X^+** (w.r.t. F)
  is a subset of attributes from A **determined by** X (using F)
  - consequence: if $X^+ = A$, then X is a **super-key**
- if F contains a FD $X \rightarrow Y$ and there exist an attribute *a* in X such that $Y \subseteq (X - a)^+$, then *a* is **attribute redundant in $X \rightarrow Y$**
- **reduced FD** is such FD that does not contain any redundant attributes (otherwise it is a partial FD)
- **key** is a set $K \subseteq A$ such that it is a super-key (i.e., $K \rightarrow A$) and $K \rightarrow A$ is moreover reduced
  - there could exist multiple keys (at least one)
  - if there is no FD in F, it trivially holds $A \rightarrow A$, i.e., the key is the entire set A
  - **key attribute** = attribute that is in **any** key

Example – attribute closure

R(A,F), A = {a,b,c,d}, F = {a → c, cd → b, ad → c}

{a}+ = {a,c}	it holds	a → c	(+ trivial a → a)
{b}+ = {b}			(trivial b → b)
{c}+ = {c}			(trivial c → c)
{d}+ = {d}			(trivial d → d)
{a,b}+ = {a,b,c}		ab → c	(+ trivial)
{a,d}+ = {a,b,c,d}		ad → bc	(+ trivial)
{c,d}+ = {b,c,d}		cd → b	(+ trivial)

Example – redundant attribute

R(A,F), A = {i,j,k,l,m},
F = {m → k, lm → j, **ijk → l**, j → m, l → i, l → k}

Hypothesis:
k is redundant in **ijk → l**, i.e., it holds **ij → l**

Proof:

1. based on the hypothesis let's construct FD **ij → ?**
2. **ijk → l** remains in F because we **ADD** new FD **ij → ?**
   so that we can use **ijk → l** for construction of the attribute closure $\{i,j\}^+$
3. we obtain $\{i,j\}^+ = \{i, j, m, k, l\}$,
   i.e., there exists **ij → l** which we add into F (it is the member of F^+)
4. now forget how **ij → l** got into F
5. because **ijk → l** could be trivially derived from **ij → l**,
   it is redundant FD and we can remove it from F
6. so, we removed the redundant attribute **k** in **ijk → l**

In other words, we transformed the problem of removing redundant attribute on the problem of removing redundant FD.

Minimal cover

- non-redundant canonic cover that consists of only reduced FDs
  - is constructed by removing redundant attributes in FDs **followed by** removing of redundant FDs (i.e., the order matters)

Example: abcd $\rightarrow$ e, e $\rightarrow$ d, a $\rightarrow$ b, ac $\rightarrow$ d

Correct order of reduction:

1. b,d are redundant in abcd $\rightarrow$ e, i.e., removing them
2. ac $\rightarrow$ d is redundant

Wrong order of reduction:

1. no redundant FD
2. redundant b,d in abcd $\rightarrow$ e

(now not a minimal cover, because ac $\rightarrow$ d is redundant)

First normal form (1NF)

Every attribute of in a relation schema is of **simple non-structured type**.

(1NF is the basic condition on „flat database" – a table is really two-dimensional array, not a hidden graph or tree)

Example – 1NF

Person(Id: **Integer**, Name: **String**, Birth: **Date**)

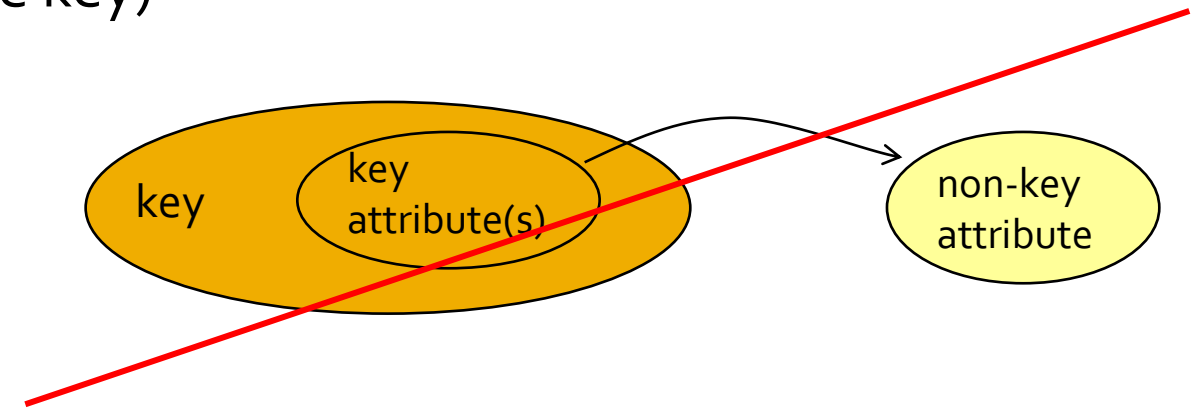
is in 1NF

Employee(Id: **Integer**, Subordinate : **Person[ ]**, Boss : **Person**)

not in 1NF
(nested table of type Person in attribute Subordinate, and the Boss attribute is structured)

2nd normal form (2NF)

- 1NF + there do not exist partial dependencies of non-key attributes on (any) key, i.e., it holds $\forall x \in NK \nexists KK : KK \rightarrow x$
  (where NK is the set of non-key attributes and KK is subset of some key)



Example – 2NF

Company	DB server	HQ	Purchase date
John's firm	Oracle	Paris	1995
John's firm	MS SQL	Paris	2001
Paul's firm	IBM DB2	London	2004
Paul's firm	MS SQL	London	2002
Paul's firm	Oracle	London	2005

Company, DB Server → *everything*
Company → HQ

← **not in 2NF**, because HQ is determined by a part of key (Company)

consequence:
redundancy of HQ values

both schemas **are in 2NF** →

Company	DB server	Purchase date
John's firm	Oracle	1995
John's firm	MS SQL	2001
Paul's firm	IBM DB2	2004
Paul's firm	MS SQL	2002
Paul's firm	Oracle	2005

Company, DB Server → *everything*

Company	HQ
John's firm	Paris
Paul's firm	London

Company → HQ

Transitive dependency on key

- FD A $\rightarrow$ B such that A $\not\rightarrow$ *some key*
  (A is not a super-key), i.e., we get transitivity *key* $\rightarrow$ A $\rightarrow$ B

- Transitivities point to probable redundancies, because from the definifion of FD as a mapping:
  - unique values of ***key*** are mapped to the same or **less** unique values of **A**, and those are mapped to the same or **less** unique values of **B**

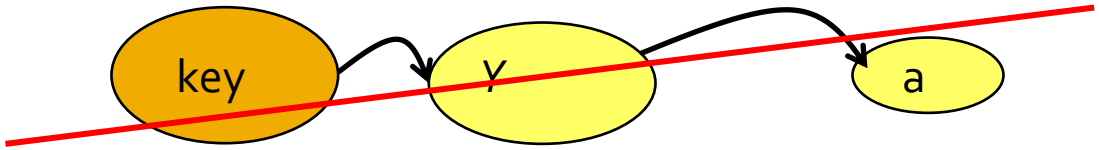
Example in 2NF:
ZIPcode $\rightarrow$ City $\rightarrow$ Country

ZIPcode	City	Country
CZ 118 00	Prague	Czech rep.
CZ 190 00	Prague	Czech rep.
CZ 772 00	Olomouc	Czech rep.
CZ 783 71	Olomouc	Czech rep.
SK 911 01	Trenčín	Slovak rep.

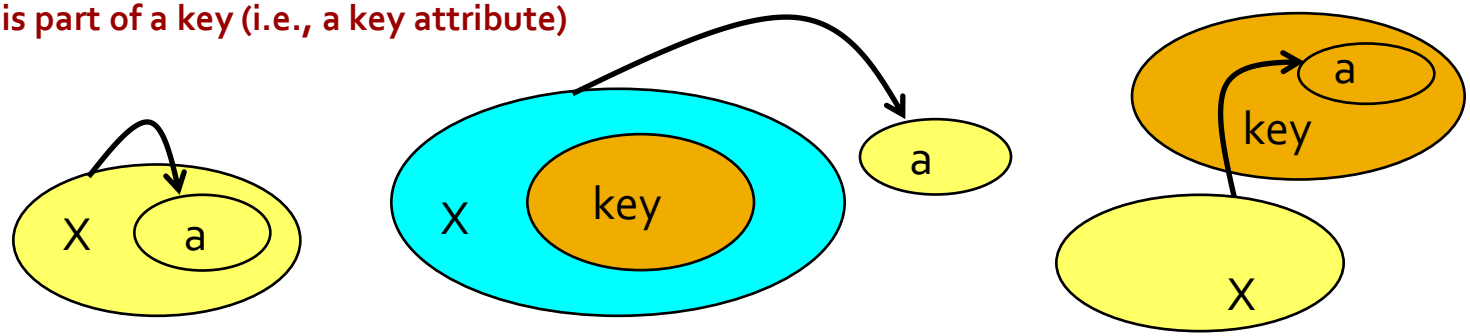
no redundancy medium redundancy high redundancy

3rd normal form (3NF)

- 1NF + non-key attributes are not transitively dependent on key



- as the 3NF using the above definition cannot be tested without construction of F^+, we use a definition that assumes only R(A,F)
  - at least one condition holds for each FD $X \rightarrow a$ (where $X \subseteq A$, $a \in A$)
    - **FD is trivial**
    - **X is super-key**
    - ***a* is part of a key (i.e., a key attribute)**



Example – 3NF

Company	HQ	ZIPcode
John's firm	Prague	CZ 11800
Paul's firm	Ostrava	CZ 70833
Martin's firm	Brno	CZ 22012
David's firm	Prague	CZ 11000
Peter's firm	Brno	CZ 22012

Company → *everything*
ZIPcode → HQ

is in 2NF, **not in 3NF** (transitive dependency of HQ on key through ZIPcode)

consequence:
redundancy of HQ values

Company → *everything*
ZIPcode → *everything*

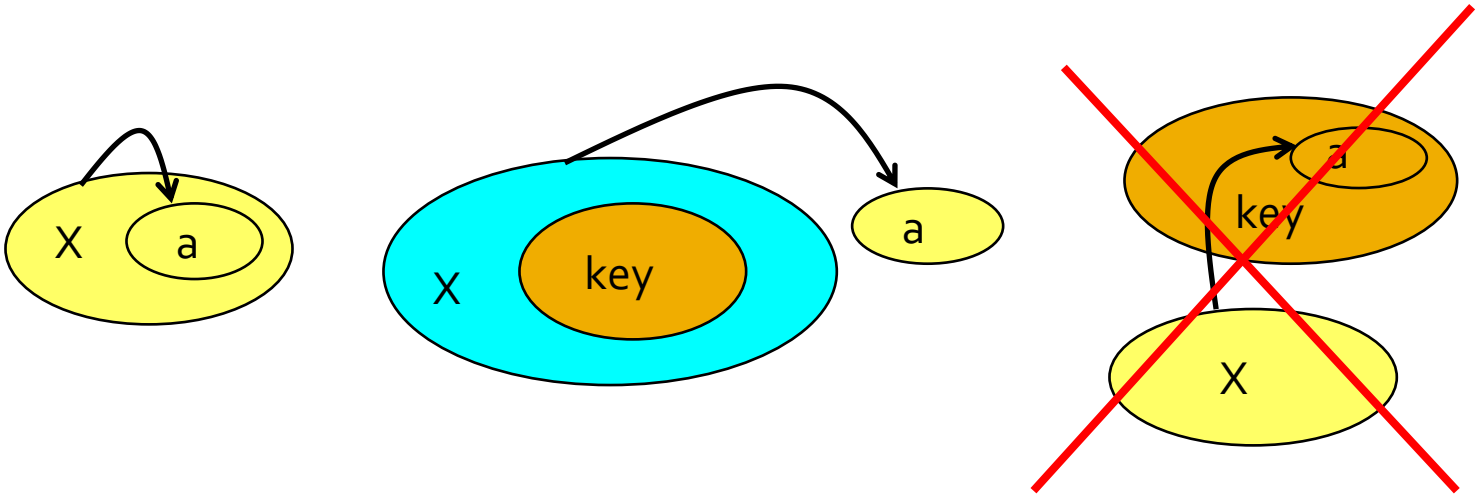
both schemas **are in 3NF**

Company	ZIPcode
John's firm	CZ 11800
Paul's firm	CZ 70833
Martin's firm	CZ 22012
David's firm	CZ 11000
Peter's firm	CZ 22012

ZIPcode	HQ
CZ 11800	Prague
CZ 70833	Ostrava
CZ 22012	Brno
CZ 11000	Prague

Boyce-Codd normal form (BCNF)

- 1NF + every attribute is (non-transitively) dependent on key
- more exactly, in a give schema R(A, F) there holds at least one condition for each FD $X \rightarrow a$ (where $X \subseteq A$, $a \in A$)
  - **FD is trivial**
  - **X is super-key**
- the same as 3NF without the last option (a is key attribute)



Example – BCNF

Destination	Pilot	Plane	Day
Paris	cpt. Oiseau	Boeing #1	Monday
Paris	cpt. Oiseau	Boeing #2	Tuesday
Berlin	cpt. Vogel	Airbus #1	Monday

Pilot, Day → *everything*
Plane, Day → *everything*
Destination → Pilot

is in 3NF, **not in BCNF**
(Pilot is determined by Destination, which is not a super-key)

consequence:
redundancy of Pilot values

Destination → Pilot
Plane, Day → *everything*

Destination	Pilot
Paris	cpt. Oiseau
Berlin	cpt. Vogel

Destination	Plane	Day
Paris	Boeing #1	Monday
Paris	Boeing #2	Tuesday
Berlin	Airbus #1	Monday

both schemas **are in BCNF**