Functional Dependencies (9)

Motivace

- výsledkem relačního návrhu je množina relačních schémat

- **problémy**:
    - **redundance dat**
    - **anomálie při vkládání/aktualizaci/mazání**
        - vkládání a aktualizace musí zachovat redundantní data storage
        - mazání může způsobit ztrátu některých dat

    - **null hodnoty**
        - zbytečné prázdné místo

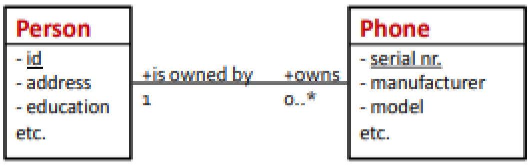
- **řešení**: normalizace relačního schématu

Příklad abnormálního schématu

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

- z funkcionální analýzy víme, že pozice určuje hodinovou mzdu
    - hodinová mzda je však ukládána několikrát → **redundance**
- smažeme-li employee 6, ztratíme informaci o mzdě lektora
- chceme-li změnit mzdu účetního, musíme to udělat na třech místech

- ***Co je příčinou?***
    - manuální návrh relačních schémat
    - špatně navržený konceptuální model (např. příliš atributů ve třídě)



        - UML diagram vede ke dvěma tabulkám
            - Person(id, address, education,…)
            - Mobil(serialnumber, maunfacturer, model,…,id)

- ***Jak to napravit?***
    - opravit UML model
    - upravit již existující schéma

Functional dependencies

- atributová integritní omezení definovaná uživatelem
- tak trochu alternativa ke konceptuálnímu modelování

functional dependency (FD) $X \rightarrow Y$ over schema R(A)

- mapping $f_i : X_i \rightarrow Y_i$, where $X_i, Y_i \subseteq A$ (where 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

- pro X → Y, hodnoty v X společně determinují hodnoty v Y
- pokud X → Y a Y → X, pak jsou X a Y funkčně ekvivalentní (X ↔ Y)
- pokud X → a, kde a ∈ A, pak X → a je elementární FD
    - tzn., na pravé straně je jen jeden atribut
- FD reprezentují generalizaci konceptu klíče (identifikátoru)

- **špatná interpretace:**

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

    - někdo by mohl z dat vyčíst, že:
        - Position → Hourly salary, a také Hourly Salary → Position
        - EmpId → vše
        - Hours completed → vše
        - Name → vše

    - … což je ale blbost

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***~~

- **správná implementace:**
    - po analýze dat jsou FD nastaveny „navždy“, omezují obsah tabulek
        - např. Position → Hourly salary, EmpId → vše

- vložené poslední řádky není povoleno, protože porušuje obě FD

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~~

Armstrongovy axiomy

Mějme R(A,F). Nechť X,Y,Z $\subseteq$ A a F je množina FD.

1. Pokud Y $\subseteq$ X, pak X → Y. (**triviální FD**)
2. Pokud X → Y a Y → Z, pak X → Z. (**transitivita**)
3. Pokud X → Y a X → Z, pak X → YZ. (**kompozice**)
4. Pokud X → YZ, pak X → Y a X→ Z. (**dekompozice**)

- jsou korektní (co je odvozeno z F, je platné pro jakoukoliv instanci z R)
- jsou kompletní (všechny FD platné v instancích R lze odvodit pomocí axiomů)
- 1,2,3 (trivialita, transitivita a kompozice) jsou nezávislé (odstranění jakéhokoliv z daných axiomů by narušilo kompletnost), dekompozice je odvoditelná z triviální FD a transitivity

Příklad – odvozování FD

R(A,F)

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

F = {ab → c, ac → d, cd → ed, e → f}

We could derive, e.g.,:

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

Příklad – odvození dekompozičního pravidla

R(A,F)

A = {a,b,c}

F = {a → bc}

Deriving:

a → bc	(assumption)
bc → b	(trivial FD)
bc → c	(trivial FD)
a → b	(transitivity)
a → c	(transitivity)

i.e., **a → bc** ⇒ **a → b** ∧ **a → c**

Uzávěr množiny FD

- **uzávěra** (**closure**) $\mathbf{F^+}$ množiny FD „F“ je množina všech FD odvoditelných z F pomocí Armstronových axiomů
- většinou exponencionální velikost vůči |F|

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

$$\begin{aligned}
F^+ = \\
&\{a \to a,\ b \to b,\ c \to c,\ d \to d,\\
&ab \to a,\ ab \to b,\ ab \to c,\\
&cd \to b,\ cd \to c,\ cd \to d,\\
&ad \to a,\ ad \to c,\ ad \to d,\\
&abd \to a,\ abd \to b,\ abd \to c,\ abd \to d,\\
&abd \to abcd,\ \ldots\}
\end{aligned}$$

Cover

- cover množiny F je jakákoliv množina FD „G“ taková, že $\mathbf{F^+} = \mathbf{G^+}$
- tj. množina FD, které mají stejnou uzávěru (generují stejnou množinu FD)

- **canonical cover** = cover složený z elementárních FD

$$\begin{aligned}
&R1(A,F),\ R2(A,G),\\
&\quad A = \{a,b,c,d\},\\
&\quad F = \{a \to c,\ b \to ac,\ d \to abc\},\\
&\quad G = \{a \to c,\ b \to a,\ d \to b\}
\end{aligned}$$

For checking that $G^+ = F^+$ we do not have to establish the whole covers,
it is sufficient to derive F from G, and vice versa, i.e.,
$F' = \{a \to c,\ b \to a,\ d \to b\}$ – decomposition
$G' = \{a \to c,\ b \to ac,\ d \to abc\}$ – transitivity and composition
$\Rightarrow G^+ = F^+$

  - schémata R1 a R2 jsou ekvivalentní, protože G je cover F a sdílí stejný set atributů A
  - ba co více, G je **minimální cover**, zatímco F není

Redundantní FD

- FD „f“ je redundantní v F, pokud $\mathbf{(F - \{f\})^+ = F^+}$
- tj. f je odvoditelné ze zbytku F
- **neredundantní cover** F = cover F po odstranění všech redundantních FD
  - pořadí odstraňování FD je důležité – redundantní FD se může stát neredundantní FD po odstranění jiného redundantního FD
  - může tedy existovat vícero neredundantních coverů F

R(A,F)
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

Atributový uzávěr, klíč

- **atributový uzávěr X^+** (vzhledem k F) je podmnožina atributů z A determinována X-kem
    - důsledek: je-li X^+ = A, pak X je super-klíč

- pokud F obsahuje FD „X → Y“ a existuje atribut „a“ v X takový, že **$Y \subseteq (X - a)^+$**, pak a je **redundantní atribut** v X → Y

- redukovaný FD neobsahuje žádné redundantní atributy¨

- pro R(A) je **klíč** množina $K \subseteq A$ taková, že je super-klíčem (tzn. K → A) a K → A je zredukovaná
    - může existovat vícero klíčů (alespoň jeden)
    - pokud v F není žádný FD, pak triviálně A → A, tj. klíč je celá množina A
    - **klíčový atribut** = atribut, který je v jakémkoliv klíči

Příklad – atributový uzávěr

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} a → c (+ trivial)
{a,d}+ = {a,b,c,d} ad → c, cd → b (+ trivial)
{c,d}+ = {b,c,d} cd → b (+ trivial)

Příklad – redundantní atribut

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. note that **ijk → l** remains in F because we **ADD** new FD **ij → ?**
   ⇒ so 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.

FD vs. atributy

Funkcionální závislosti	Atributy
mohou být redundantní	mohou být redundantní
mohou být uzávěr (všechny odvoditelné FD)	mohou mít uzávěr (vš. od. atributy)
mohou být elementární (jediný atribut vpravo)	může tvořit (super-)klíče
mohou být redukovány (žádné redundance vlevo)	

Minimální cover

- neredundantní kanonický cover, který se skládá jen z redukovaných FD
  - tj. žádné redundantní FD, žádné redundantní atributy, dekomponované FD
  - je vytvořen odstraněním redundantních atributů v FD, a pak odstraněním redundantních FD (na pořadí záleží)

Example: abcd → e, e → d, a → b, ac → d

Correct order of reduction:

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

Wrong order of reduction:

1. no redundant FD
2. redundant b,d in abcd → e

(now not a minimal cover, because
ac → d is redundant)

Klíče

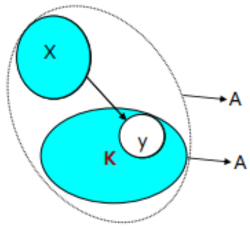
Nalezení (prvního) klíče

- redundantní atributy iterativně odstraníme z levé strany triviálního FD A → A

```
algorithm GetFirstKey(set of deps. F, set of attributes A)
   : returns a key K;
   return ReduceAttributes(F, A → A);
```

- může existovat vícero klíčů, algoritmus najde jen první z nich

Nalezení všech klíčů



Princip

- mějme schéma S(A,F)
- zjednodušme F na minimální cover

1.) Najít jakýkoliv klíč K.
2.) Vzít FD X → y v „F“ takové, že y ∈ K (případně skončit, pokud neexistuje).
3.) Protože X → y a K → A, tranzitivně také X{K-y} → A, tzn. X{K-y} je superklíč.
4.) Redukovat FD X{K-y} → A, získáme klíč K‘ na levé straně. Klíč je určitě odlišný od K (odebrali jsme y).
5.) Pokud K‘ ještě není mezi nalezenými klíči, přidáme ho, deklarujeme K = K‘ a pokračujeme krokem 2. Jinak skončíme.

Algoritmus

- Lucchesi-Osbornův algoritmus
    - již máme nalezený klíč, hledáme ekvivalentní množiny atributů (tj. další klíče)
- NP-complete problém

```
algorithm GetAllKeys(set of FDs F, set of attr. A)
   : returns set of all keys Keys;
   let all dependencies in F be non-trivial
   K := GetFirstKey(F, A);
   Keys := {K};
   for each K in Keys do
      for each X → Y in F do
         if (Y ∩ K ≠ ∅ and ¬∃K' ∈ Keys : K' ⊆ (K ∪ X) - Y) then
            N := ReduceAttributes(F, ((K ∪ X) - Y) → A);
            Keys := Keys  ∪ {N};
         endif
      endfor
   endfor
return Keys;
```

Příklad

Contracts(A,F)

A = {c = ContractId, s = SupplierId, j = ProjectId, d = DeptId, p = PartId, q = Quantity, v = Value}

F = {c $\rightarrow$ all, sd $\rightarrow$ p, p $\rightarrow$ d, jp $\rightarrow$ c, j $\rightarrow$ s}

1.) První klíč – Keys = {c}
2.) Iterace 1: Vezmeme **jp $\rightarrow$ c**, které má část posledního klíče na pravé straně (v tomto případě celý klíč – **c**) a **jp** není superklíč již nalezeného klíče
3.) **jp $\rightarrow$ all** je redukované (žádné redundantní atributy), tj. **Keys = {c, jp}**
4.) Iterace 2: Vezmeme **sd $\rightarrow$ p**, které má část posledního klíče na pravé straně (**jp**), {**jsd**} není super klíč „**c**“ ani „**jp**“, tj. je to kandidát na klíč
5.) v **jsd $\rightarrow$ all** dostaneme redundantní atribut „s“, tzn. **Keys = {c, jp, jd}**
6.) Iterace 3: Vezmeme **p $\rightarrow$ d**, ale **jp** je již nalezeno, takže jej nepřidáme.
7.) Skončíme, jelikož třetí iterace nevyústila v přidání klíče.

Normálové formy

První normálová forma (1NF)

Každý atribut v relačním schématu je jednoduchého nestrukturovaného typu.

- 1NF je základní podmínkou pro „flat database“
- tabulka je vlastně dvoudimenzionální pole

Příklad

Person(Id: Integer, Name: String, Birth: Date) – **je v 1NF**

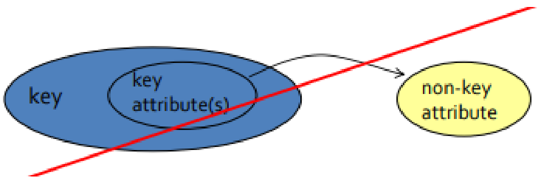
Employee(Id: Integer, Subordinate : Person[ ], Boss : Person) – **není v 1NF** (vnořená tabulka, strukturovaný typ)

Druhá normálová forma (2NF)

Neexistují parciální závislosti neklíčových atributů na (jakémkoliv) klíči, tzn.:

$$\forall x \in NK\ \exists KK : KK \rightarrow x$$

- NK = množina neklíčových atributů
- KK = podmnožina nějakého klíče



Příklad

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

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

consequence:
redundancy of HQ values

Company, DB Server → *everything*
Company → HQ

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

Tranzitivní závislost na klíči

- **FD A → B taková, že A ! → nějaký klíč**
  - (A není super klíč), tj. máme tranzitivitu klíč → A → B

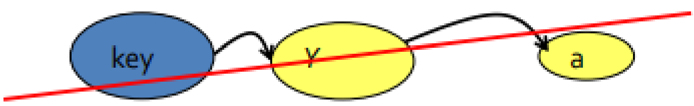
- tzn., unikátní hodnoty klíče jsou mapovány na stejně či méně unikátní hodnoty A, a ty jsou mapovány na stejně či méně unikátní hodnoty B
  Example in 2NF:
  ZIPcode → City → Country

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

Třetí normálová forma (3NF)

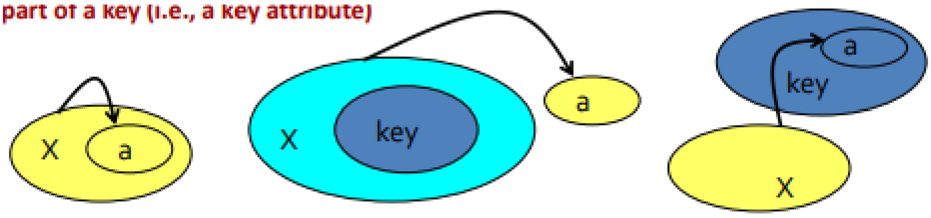
Neklíčové atributy nejsou tranzitivně závislé na klíči.



- Jelikož 3NF definovaná výše nemůže být otestována bez vytvoření F^+, využíváme definici která předpokládá jen R(A,F):

- **Platí alespoň jedna podmínka pro každou FD X → a (kde X ⊆ A, a ∈ A):**
  - **FD je triviální**
  - **X je superklíč**
  - **a je část klíče** (tj. klíčový atribut)

part of a key (i.e., a key attribute)



Příklad

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

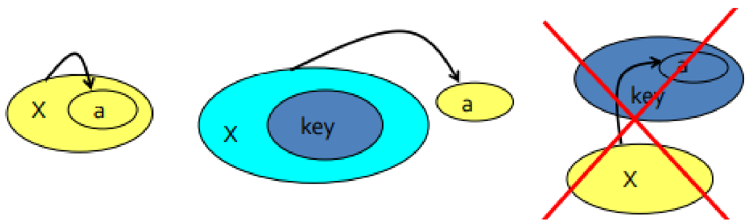
Boyce-Coddova normální forma (BCNF)

Každá atribut je (netranzitivně) závislý na klíči.

Přesnějim v daném schématu R(A,F) je splněna alespoň jedna podmínka pro každou FD $X \rightarrow a$ (kde $X \subseteq A$, $a \in A$):

- **FD je triviální**
- **X je superklíč**

- stejné jako 3NF bez poslední možnosti (a je klíčový atribut)



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*

both schemas are in BCNF

Destination	Pilot
Paris	cpt. Oiseau
Berlin	cpt. Vogel

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