Vysoká škola báňská – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky

Úvod do databázových systémů

12. cvičení

Ing. Petr Lukáš
petr.lukas@nativa.cz
Ostrava, 2013

Opakování

- **Univerzální relační schéma**
- **Funkční závislost**
- **Armstrongovy axiomy**
- **Uzávěry množiny atributů**
- **Klíč schématu**
- **Minimální neredundantní pokrytí**

Opakování

- **Univerzální relační schéma**
  „široká“ nepřehledná tabulka, která obsahuje všechny atributy (např. modelovaného systému)
- **Funkční závislost**
  ze znalosti hodnot nějaké množiny hodnot atributů (a samozřejmě obsahu databáze) znám i množinu hodnot jiných atributů.
- **Armstrongovy axiomy**
  odvozovací pravidla pro funkční závislosti.
- **Uzávěry množiny atributů**
  hodnoty kterých všech atributů jsem schopný získat na základě určité dané množiny atributů
- **Klíč schématu**
  atributy, kterými jednoznačně identifikuji celý záznam, tzn. pokud znám hodnoty těchto atributů, umím v univerzálním schématu dohledat obsah celého záznamu
- **Minimální neredundantní pokrytí**
  rozložím FZ na elementární FZ, z levých stran odstraním nadbytečné atributy a nakonec celé nadbytečné FZ.

Pojmy

- **Redundance, konsistence, integrita**
- **Normální formy relací**
- **Automatický návrh databáze**

Redundance, konsistence, integrita

Redundance, Konzistence

login	jmeno	prijmeni	id_fakulty	nazev_fakulty
kab242	Prokop	Kabel	1	FEI
bet102	Zdislava	Betonová	1	FEI
jed135	Tomáš	Jedno	2	FBI

Redundance, Konzistence

login	jmeno	prijmeni	id_fakulty	nazev_fakulty
kab242	Prokop	Kabel	1	FEI
bet102	Zdislava	Betonová	1	~~**FEI**~~ **→ EKF**
jed135	Tomáš	Jedno	2	FBI

- **Co se stane, když změním název fakulty pro bet102?**

Redundance, Konzistence

login	jmeno	prijmeni	id_fakulty	nazev_fakulty
kab242	Prokop	Kabel	1	FEI
bet102	Zdislava	Betonová	1	~~**FEI**~~ **→ EKF**
jed135	Tomáš	Jedno	2	FBI

- **Co se stane, když změním název fakulty pro bet102?** Stejnému ID fakulty bude pokaždé odpovídat jiný název a to je zjevně nesmysl. Data budou tzv. nekonzistentní.

Redundance, Konzistence

login	jmeno	prijmeni	id_fakulty	nazev_fakulty
kab242	Prokop	Kabel	1	FEI
bet102	Zdislava	Betonová	1	~~FEI~~ → EKF
~~jed135~~	~~Tomáš~~	~~Jedno~~	~~2~~	~~FBI~~

- **Co se stane, když změním název fakulty pro bet102?** Stejnému ID fakulty bude pokaždé odpovídat jiný název a to je zjevně nesmysl. Data budou tzv. nekonzistentní.

- **Co se stane, když smažu záznam pro jed135?**

Redundance, Konzistence

login	jmeno	prijmeni	id_fakulty	nazev_fakulty
kab242	Prokop	Kabel	1	FEI
bet102	Zdislava	Betonová	1	~~FEI~~ → EKF
~~jed135~~	~~Tomáš~~	~~Jedno~~	~~2~~	~~FBI~~

- **Co se stane, když změním název fakulty pro bet102?** Stejnému ID fakulty bude pokaždé odpovídat jiný název a to je zjevně nesmysl. Data budou tzv. nekonzistentní.

- **Co se stane, když smažu záznam pro jed135?** Pokud nebudu mít k dispozici jinou tabulku s fakultami, nadobro ztratím informace o fakultě FBI.

Redundance, Konzistence

- **Redundance**
  Znamená obecně nadbytečnost dat. Stejná data jsou v databázi uchovávána vícekrát a to je obvykle **nežádoucí jev**.

- **Konzistence**
  Když už redundance nastane (což se prakticky může stát), musíme zajistit, aby data nebyla nekonzistentní (viz příklad na předchozím slidu). Tzn. konzistence je naopak **žádoucí jev**.
  Nekonzistence je důsledkem redundance.

Integrita

id_republiky	nazev	hlavni_mesto	Prezident
1	Česká republika	Praha	Václav Havel
2	USA	Washington	Arnold Schwarzenegger
3	Německo	Bonn	Richard von Weizsäcker

- **Co je na datech na první pohled v nepořádku?**

Integrita

id_republiky	nazev	hlavni_mesto	Prezident
1	Česká republika	Praha	Václav Havel
2	USA	Washington	Arnold Schwarzenegger
3	Německo	Bonn	Richard von Weizsäcker

- **Co je na datech na první pohled v nepořádku?**
  Při troše základního vzdělání minimálně víme, že dnes, v roce 2013, Václav Havel určitě už není současný prezident ČR, Arnold není prezidentem USA (i když by si to asi přál) a Bonn není současným hlavním městem Německa. Data nejsou integritní.

Integrita

- **Integrita**
  Integrita dat znamená soulad se zobrazovanou realitou. Integrita je **žádoucí jev**.

  Integritou také rozumí, že jsou dodržena všechna definovaná integritní omezení v databázi, tzn. např. neexistuje cizí klíč, který by se odkazoval na neexistující ID.

login	jmeno	prijmeni	id_fakulty
kab242	Prokop	Kabel	1
bet102	Zdislava	Betonová	2
jed135	Tomáš	Jedno	**3**

id_faulkty	nazev_fakulty
1	FEI
2	FBI

Normální formy relací

1. normální forma

Relace je v 1. normální formě, jestliže obsahuje pouze atomické (dále nedělitelné) atributy.

login	jmeno	adresa
kab242	Prokop	Za Mostem 23, Ostrava-Svinov
bet102	Zdislava	710 00, Slezská Ostrava, Bohumínská 22
jed135	Tomáš	U točny 2, Psojedy, Česká republika

- Ukázková relace porušuje pravidlo pro 1. normální formu. Pokud bychom např. chtěli hromadně předtisknout obálky, asi bychom s takto uvedenou adresou moc dobře nepořídili.
- Na druhou stranu co je to „atomický“ atribut není jednoznačně dáno. Co má např. obsahovat atribut *ulice*? Atomičnost je potřeba citlivě volit podle účelu systému.

2. normální forma

Relace je ve 2. normální formě, jestliže je v 1. NF a každý neklíčový atribut (tj. ten, který není součástí žádného klíče) je úplně závislý na každém klíči. Tzn. není závislý na žádném podklíči.

hodina	**ucebna**	**predmet**	**kapacita**
10:45	A1033	UDBS	16
7:15	G317a	DAIS	20
9:30	A1033	MAIT	16

- Ukázka opět porušuje 2. NF, protože platí funkční závislost **ucebna → kapacita**, ale klíčem relace jsou dohromady **hodina** a **ucebna**. Tzn. kapacita je závislá na podklíči a to je špatně.

3. normální forma

Relace je ve 3. normální formě, jestliže je ve 2. NF a neexistují netriviální závislosti mezi neklíčovými atributy.

login	**jmeno**	**prijmeni**	**id_fakulty**	**nazev_fakulty**
kab242	Prokop	Kabel	1	FEI
bet102	Zdislava	Betonová	1	FEI
jed135	Tomáš	Jedno	2	FBI

- Název fakulty a její id jsou dva neklíčové atributy, mezi kterými existuje funkční závislost **id_fakulty → nazev_fakulty**. Tato relace tedy není ve 3. normální formě.

Boyce-Coddova normální forma

Relace je v BCNF, jestliže pro každou netriviální funkční závislost $\boldsymbol{X} \rightarrow \boldsymbol{Y}$ platí, že $\boldsymbol{X}$ je klíč nebo nadklíč.

sluzba	**zakaznik**	**tarif**
internet	Petr	10 GBit/s
internet	Jan	100 GBit/s
telefon	Petr	SUPER
telefon	Pavel	EXTRA

- V uvedeném příkladu je problém v tom, že existuje funkční závislost **tarif → sluzba**. Tarify *SUPER* a *EXTRA* jsou vždy pro telefonní službu, zatímco *10 GBit/s* a *100 GBit/s* se vždy týkají internetu. V relaci tedy existuje závislost, kde levá strana netvoří klíč nebo nadklíč.

Automatický návrh databáze

Automatický návrh databáze

- **Univerzální schéma**
  Ať už zvolíme kterýkoli algoritmus pro návrh databáze, prvním krokem je vždy sestavení univerzálního schématu $\boldsymbol{RU}$, tj. předpisu pro tabulku s komplet všemi atributy.

- **Dekompozice univerzálního schématu**
  Existují dva základní algoritmy
  - **Algoritmus dekompozice do BCNF**
  - **Algoritmus dekompozice do 3.NF** (často též nazýván jako algoritmus syntézy)

Dekompozice

Dekompozice relačního schématu $\boldsymbol{R(A)}$ je množina relačních schéma $\boldsymbol{RO} = \{\boldsymbol{R_1(A_1)}, \boldsymbol{R_2(A_2)}, \ldots \boldsymbol{R_n(A_n)}\}$, přičemž platí, že $\boldsymbol{A_1} \cup \boldsymbol{A_2} \cup \cdots \cup \boldsymbol{A_n} = \boldsymbol{A}$.

Student (<u>login</u>, jmeno, id_katedry, nazev)

Jeden z možných rozkladů může vypadat např.:

O_1 (login, jmeno, id_katedry)
O_2 (id_katedry, nazev)

Dekompozice

- Při dekompozici samozřejmě nesmíme přijít o žádný atribut, nicméně po dekompozici požadujeme trochu více:

1. Nesmíme porušit tzv. **zákon zachování informace**
2. Nesmíme porušit tzv. **zákon zachování množiny funkčních závislostí**

Zákon zachování informace

1. Mějme původní schéma $\boldsymbol{R(A)}$
2. Množinu funkčních závislostí $\boldsymbol{F}$
3. Rozklad $\boldsymbol{RO = \{R_1(A_1), R_2(A_2)\}}$

Zákon zachování informace

Ke ztrátě informace nedojde, jestliže pro každou relaci $\boldsymbol{R}$ platí, že $\boldsymbol{R = R_1 \bowtie R_2}$.

R (idZam, jméno, firma, název)

idZam	jméno	firma	název
1	Petr	1	VŠB
2	Pavel	2	Tieto
3	Lukáš	1	VŠB
4	Jakub	2	Tieto

$=$

R_1 (idZam, jméno, firma)

idZam	jméno	firma
1	Petr	1
2	Pavel	2
3	Lukáš	1
4	Jakub	2

$\bowtie$

R_2 (firma, název)

firma	název
1	VŠB
2	Tieto

Zákon zachování množiny FZ

1. Mějme původní schéma $\boldsymbol{R(A)}$
2. Množinu funkčních závislostí $\boldsymbol{F}$
3. Rozklad $\boldsymbol{RO} = \{\boldsymbol{R_1(A_1)}, \boldsymbol{R_2(A_2)}\}$

Zákon zachování množiny FZ

Ke ztrátě množiny FZ nedojde, jestliže z funkčních závislostí pokrytých ve schématech $\boldsymbol{R_1}$ a $\boldsymbol{R_2}$ lze odvodit (např. pomocí armstrongových axiomů) původní množinu funkčních závislostí $\boldsymbol{F}$.

Zákon zachování množiny FZ

R (idZam, jméno, firma, název)

idZam	jméno	firma	název
1	Petr	1	VŠB
2	Pavel	2	Tieto
3	Lukáš	1	VŠB
4	Jakub	2	Tieto

idZam → jméno, firma

firma → název

R_1 (idZam, jméno)

idZam	jméno
1	Petr
2	Pavel
3	Lukáš
4	Jakub

$F[R_1]$: idZam → jméno

R_2 (idZam, firma)

idZam	firma
1	1
2	2
3	1
4	2

$F[R_2]$: idZam → firma

R_3 (firma, název)

firma	název
1	VŠB
2	TIeto

$F[R_3]$: firma → název

Zákon zachování množiny FZ

R (idZam, jméno, firma, název)

idZam	jméno	firma	název
1	Petr	1	VŠB
2	Pavel	2	Tieto
3	Lukáš	1	VŠB
4	Jakub	2	Tieto

idZam → jméno, firma

firma → název

R_1 (idZam, jméno)

idZam	jméno
1	Petr
2	Pavel
3	Lukáš
4	Jakub

$F[R_1]$: idZam → jméno

R_2 (firma, název)

firma	název
1	VŠB
2	TIeto

$F[R_2]$: firma → název

Ztratila se nám závislost mezi idZam a firmou.

Algoritmus dekompozice do BCNF

- Najdeme FZ $\boldsymbol{X} \rightarrow \boldsymbol{Y}$, která porušuje podmínku BCNF, a provedeme rozklad schématu $\boldsymbol{R}$ na dvě schémata $\boldsymbol{R_1}$ a $\boldsymbol{R_2}$.
  Schéma $\boldsymbol{R_1}$ bude obsahovat všechny atributy $\boldsymbol{R}$ bez atributů $\boldsymbol{Y}$, druhé schéma $\boldsymbol{R_2}$ bude obsahovat dohromady atributy $\boldsymbol{X}$ a $\boldsymbol{Y}$.
- Rozkládáme tak dlouho, dokud existují relace a FZ, které porušují podmínku BCNF.
- Bohužel, záleží na tom, v jakém pořadí vytahujeme jednotlivé FZ a může dojít ke **ztrátě informace** nebo **ztrátě funkční závislosti**.

Algoritmus dekompozice do BCNF

R (A, B, C, D, E, F, G)

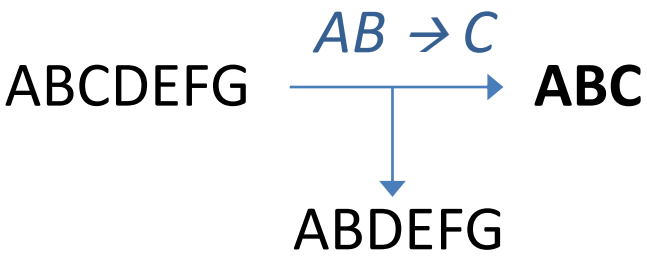
F: {AB → C, C → D, B → E, E → F, C → G}

ABCDEFG

Algoritmus dekompozice do BCNF

R (A, B, C, D, E, F, G)

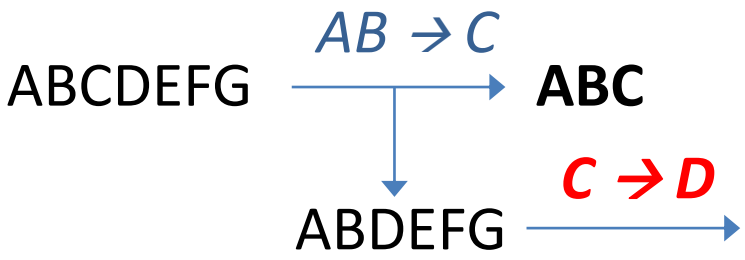
F: {AB → C, C → D, B → E, E → F, C → G}



Algoritmus dekompozice do BCNF

R (A, B, C, D, E, F, G)

F: {AB → C, C → D, B → E, E → F, C → G}

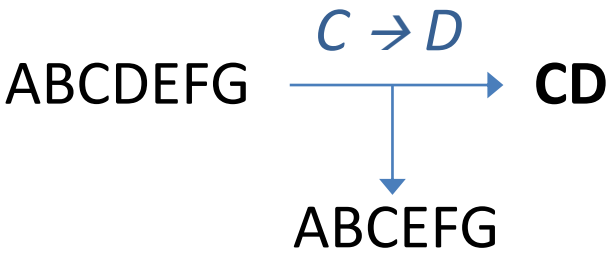


Problém, pokud v prvním kroku použiju neuváženě **AB → C**, dostanu se do pasti, protože už nebudu moct nikde uplatnit **C → D**. To znamená, že jsem přišel o funkční závislost a poruším tedy zákon o zachování množiny FZ.

Algoritmus dekompozice do BCNF

R (A, B, C, D, E, F, G)

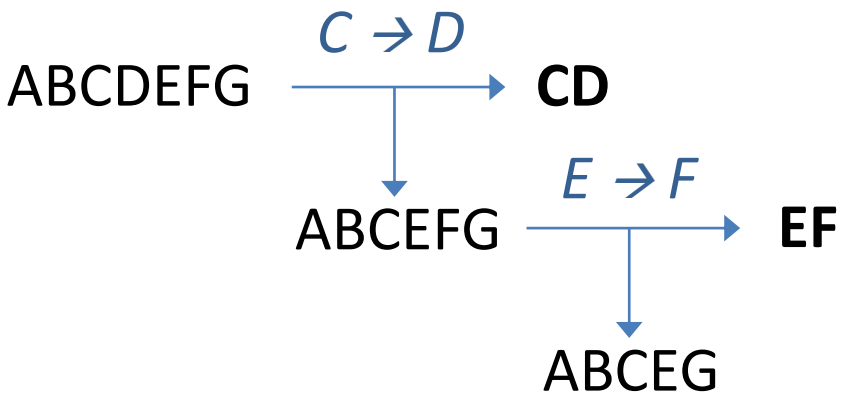
F: {AB → C, C → D, B → E, E → F, C → G}



Algoritmus dekompozice do BCNF

R (A, B, C, D, E, F, G)

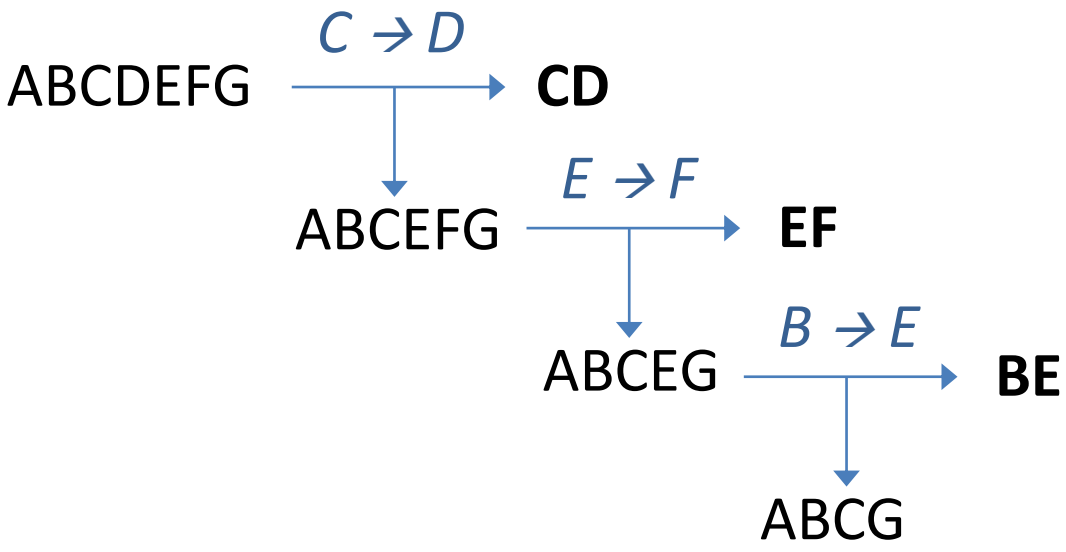
F: {AB → C, C → D, B → E, E → F, C → G}



Algoritmus dekompozice do BCNF

R (A, B, C, D, E, F, G)

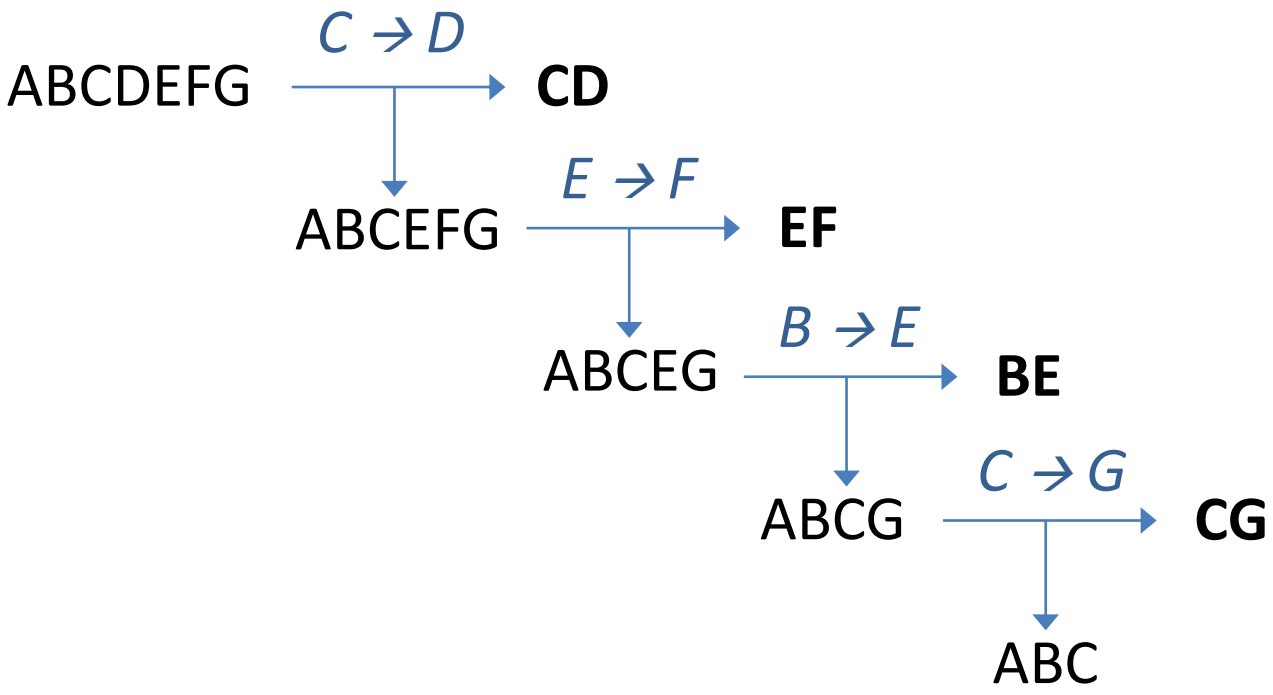
F: {AB → C, C → D, B → E, E → F, C → G}



Algoritmus dekompozice do BCNF

R (A, B, C, D, E, F, G)

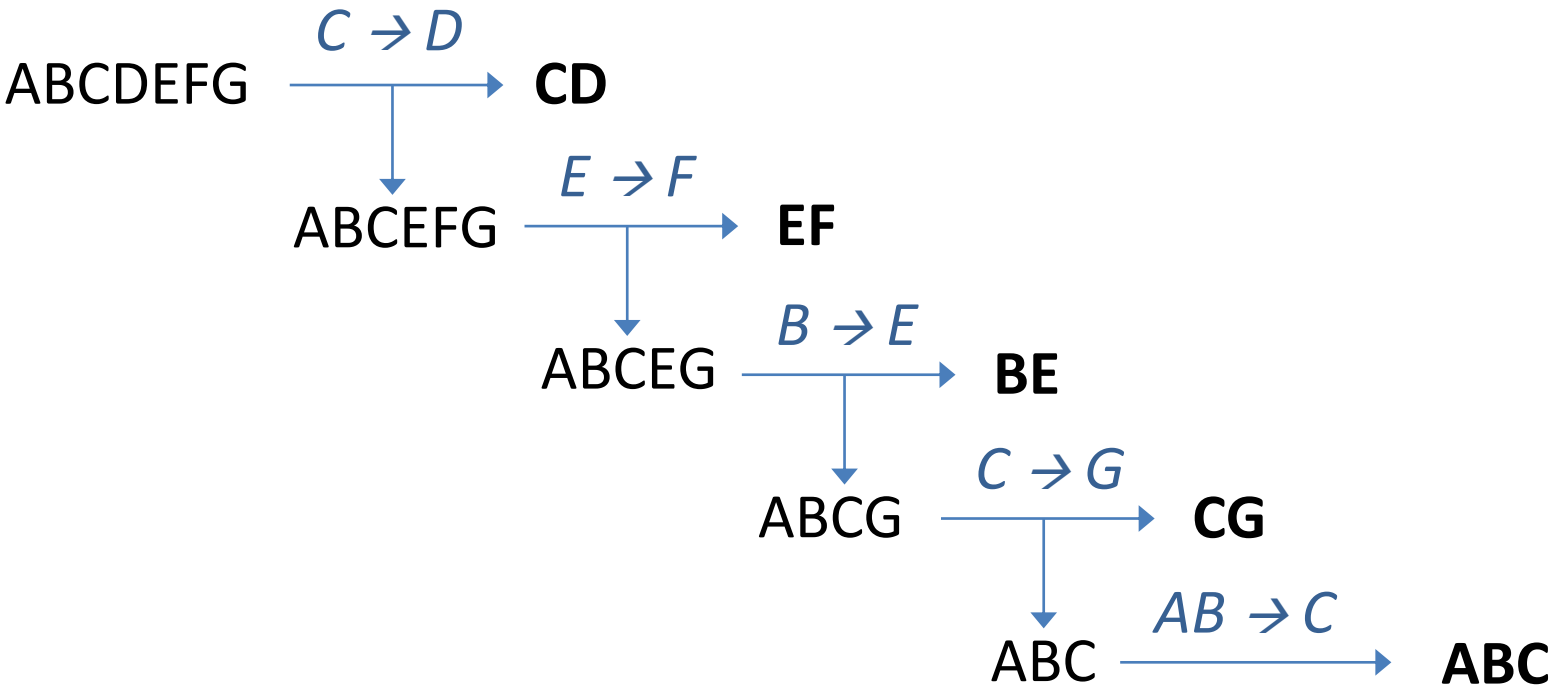
F: {AB → C, C → D, B → E, E → F, C → G}



Algoritmus dekompozice do BCNF

R (A, B, C, D, E, F, G)

F: {AB → C, C → D, B → E, E → F, C → G}



Algoritmus dekompozice do BCNF

Získali jsme tedy rozklad

$$\boldsymbol{R_1(CD), R_2(EF), R_3(BE), R_4(CG), R_5(ABC)}$$

- Je jasné, že výsledná schémata musí být v BCNF, protože jestliže existovala FZ, která porušovala BCNF, pak podle této FZ musel proběhnout rozklad.

Algoritmus dekompozice do 3.NF

1. Sestavíme minimální neredundantní pokrytí množiny FZ a nalezneme klíč. Pro každou FZ vytvoříme zvlášť relaci. V prvním kroku je přesně tolik „tabulek“, kolik je FZ v min. nered. pokrytí.
2. Spojíme relace které vznikly z FZ se stejnými levými stranami.
3. Spojíme relace s ekvivalntními klíči, tzn. těmi klíči, jejichž uzávěr je stejný. V tomto kroku může dojít k porušení podmínek pro BCNF!
4. Pokud existuje atribut univerzálního schématu, který dosud není zařazen v žádné ze vzniklých relací (tzn. nebyl obsažen v žádné FZ), pak jej přidáme do libovolné vzniklé relace.
5. Pokud žádná z relací neobsahuje celý klíč univerzálního schématu, pak vytvoříme novou relaci, která se bude skládat z atributů tohoto klíče.

Algoritmus dekompozice do 3.NF

R (A, B, C, D, E, F, G)

F: {AB → C, C → D, B → E, E → F, C → G}

---

1. Minimální neredundantní pokrytí a vytvoření mnoha „malých“ relací. Klíč schématu je **AB**.
   RO_1 = (R_1(ABC), R_2(CD), R_3(BE), R_4(EF) R_5(CG))
2. Spojíme relace, které vznikly z FZ se stejnými levými stranami.
   Žádné takové nejsou, takže **RO_2 = RO_1**
3. Spojíme relace s ekvivalntními klíči
   Opět žádné nejsou, takže **RO_3 = RO_2**
4. Pokud zbyl nějaký nezpracovaný atribut, přidáme jej do kterékoli z relací.
   Nezbyl, takže nic, **RO_4 = RO_3**
5. Není-li klíč obsažen v žádné z relací, pak vytvoříme novou relaci obsahující atributy klíče.
   Klíč **AB** je obsažen v **R_1**, talže **RO_5** = **RO_4** a máme výsledek
   R_1(ABC), R_2(CD), R_3(BE), R_4(EF) R_5(CG)

Shrnutí

- **Redundance**
  Nadbytečnost dat, stejná data jsou uchována vícekrát
- **Konsistence**
  Když už k redundanci dojde, musíme zajistit konzistenci dat
- **Integrita**
  Znamená soulad se zobrazovanou realitou
- **1. Normální forma**
  Všechny atributy jsou atomické
- **2. Normální forma**
  Každý neklíčový atribut je úplně závislý na klíči
- **3. Normální forma**
  Neexistují netriviální závislosti mezi neklíčovými atributy
- **BCNF**
  Pro každou FZ platí, že levá strana je klíč nebo nadklíč
- **Dekompozice**
  Musí obsahovat všechny atributy, nesmí dojít ke ztrátě informace nebo funkční závislosti
- **Algoritmus dekompozice do BCNF**
- **Algoritmus dekompozice do 3.NF**

Cvičení

www.dbedu.cs.vsb.cz

- Přihlášení přes **jednotný login** a **heslo**
- Vpravo sloupec -> *České kurzy* -> *UDBS*