Relační model, relační algebra

Michal Valenta

Katedra softwarového inženýrství
Fakulta informačních technologií
České vysoké učení technické v Praze

BI-DBS, LS 2021/2022

https://courses.fit.cvut.cz/BI-DBS/

Relační model dat (Codd 1970)

Odkud vychází, co přináší?

- Formální abstrakce textových souborů.
- Relační kalkul a relační algebra (dotazovací prostředky).
- Metodika pro posuzování kvality relačního schématu.
- Metodika pro návrh kvalitního relačního schématu.

Relační algebra – ukázka na začátek

```
filmy1(jmeno_f, reziser, rok)
kina1(nazev_k, adresa)
predstaveni1(nazev_k, jmeno_f, datum)
obsazeni12(jmeno_f,herec)
obsazeni90(jmeno_f,herec)
```

1. $\{filmy1(rok < 1990)[jmeno\_f] * predstaveni1\}[nazev\_k]$
2. $filmy1 < * predstaveni1$
3. $obsazeni12[herec] \cup obsazeni90[herec]$
4. $obsazeni12[herec] \cap obsazeni90[herec]$
5. $obsazeni12[herec] - obsazeni90[herec]$
6. $filmy1 \times kina1$
7. $predstaveni1[nazev\_k, jmeno\_f] \div \{filmy1(reziser = 'Woody\ Allen')[jmeno\_f]\}$

Souvislosti

Souvislost relační algebry (RA) a jazyka SQL

- RA je POUZE dotazovací formalismus
- každý výraz v RA lze přeložit na SELECT příkaz (viz DBS portál)
- části SQL jazyka - DDL, DML(mimo SELECT), TCL, DCL v RA nejsou
- zavedeme si (v této přednášce) notaci pro vyjádření relačního schématu (abychom se měli nad čím dotazovat)

Souvislost relačního modelu dat (RMD) a RDBMS

- RDBMS implementují relační model dat
- nedělají to úplně přesně - viz dále v této přednášce
- intuitivní a ilustrativní náhled - relační databáze je sada tabulek, jak je známe z tabulkových procesorů (Libre Office, MS Excel) - není úplně přesný

doplňková vsuvka – Jazyk SQL

- Říkáme, co chceme získat, ne jak se to má technicky dělat.
- Intuitivně srozumitelný zápis.
  Připomíná jednoduché anglické věty.
  Nepřipouští se žádné zkratky.
- Podpora odsazování, lámání na více řádek = snadná čitelnost.
- Klíčová slova a názvy objektů nejsou case sensitive, porovnání řetězců ano (pokud není v implementaci na úrovni session řečeno jinak – MySQL)
- Standardizace: 1986, **1992**, 1999, 2003, 2005, 2008, 2011 ...
- Nový standard obvykle zahrnuje předchozí.
- Standardy postupně “nabalují“ další rysy (OO, XML, ...)
- Poslední čistě relační standard byl 1992.
- Implementace mají svoje odchylky – většinou podstatná část standardu 92.

doplňková vsuvka – Rozdělení SQL

- Jazyk pro definici dat (Data Definition Language, DDL)
  - Jazyk pro definice pohledů
  - Jazyk pro definice integritních omezení
- Jazyk pro manipulaci dat a dotazování (Data Manipulation Language, DML)
- Jazyk pro přiřazení přístupových práv (Data Control Language, DCL)
- Jazyk pro řízení transakcí (Transaction Control Language, TCL)
- Data Dictionary
- Jazyk modulů – packages, procedury, triggery

Relace – pojmy

- jména **atributů** $[A_1, A_2, A_3, \ldots, A_n]$
- **domény** atributů D_i nebo $dom(A_i)$
  1.NF požaduje, aby atributy byly atomické.
- **n-tice** $(a_1, a_2, \ldots, a_n)$ (prvek relace)
- množina n-tic $\subset D_1 \times D_2 \ldots \times D_n$ je **relace**
- **jméno relace** R
- **schéma relace** $R(A_1 : D_1, A_2 : D_2, \ldots, A_n : D_n)$,
  zkráceně $R(A)$

Intuitivně (ale nepřesně, viz dále):

- relace = tabulka
- schéma relace = záhlaví tabulky

Jednoduchý dotaz na tabulku v rozsahu RA

SELECT – základní syntaxe

SELECT DISTINCT specifikace_sloupců
FROM specifikace_zdroje
[WHERE podmínka_selekce]

Relace vs. databázové tabulky 1/3

Mějme tabulky:

KINO* a **FILM***

NazevK	**Adresa**
Blaník	Václ. n. 4
Vesna	Olšiny 3
Mír	Strašnická 3
Domovina	V dvorcích 7

JmenoF	**Herec**	**Rok**
Černí baroni	Vetchý	1994
Černí baroni	Landovský	1994
Top gun	Cruise	1986
Top gun	McGillis	1986
Kmotr	Brando	1972
Nováček	Brando	1990
Vzorec	Brando	1980

Dále máme relace:

KINO(NazevK, Adresa) a **FILM(JmenoF, Herec, Rok)**.

Otázky:

Jaký je vztah mezi relacemi KINO a FILM a tabulkami KINO* a FILM*?

Jak zařídit vztah – film je na programu kina?

U jednoho filmu chceme sledovat více herců. Jak to zařídit?

Relace vs. databázové tabulky 2/3

Jaký je vztah mezi relacemi KINO a FILM a tabulkami KINO* a FILM*?

Relace vs. tabulky – terminologie

relace	tabulka
schéma relace	záhlaví tabulky
jméno atributu	jméno sloupce
atribut	sloupec
n-tice	řádek tabulky

Odlišnosti:

- V relaci nezáleží na pořadí n-tic.
- Relace neobsahují duplicitní n-tice (jsou to množiny).

Relace vs. databázové tabulky 3/3

U jednoho filmu chceme sledovat více herců. Jak to zařídit?

Nápad 1: Více atributů herec.

NázevF	**Herec1**	**Herec2**	**Rok**
Černí baroni	Vetchý	Landovský	1994
Kmotr	Brando		1972

Co když je více než dva herci? Když naopak méně $\rightarrow$ prázdné sloupce.

Nápad 2: Vícehodnotový atribut.

NázevF	**Herec**	**Rok**
Černí baroni	Vetchý, Landovský	1994
Kmotr	Brando	1972

Vícehodnotové atributy se v relačním modelu nepřipouští. Není v 1NF.

Nápad 3: Více řádek pro jeden film.

NázevF	**Herec**	**Rok**
Černí baroni	Vetchý	1994
Černí baroni	Landovský	1994

Zbytečná redundance, update anomálie. Není ve 2NF.

Detailní diskuse a “čisté” relační řešení – přednáška o normalizaci schématu.

Relační model dat

Je nutné zajistit, aby se do relací dostala pouze "správná data" – přípustné n-tice.

Def.: Schéma (relační) databáze:

(R, I) je **schéma relační databáze**, kde:

- $R = \{R_1, R_2, .., R_n\}$ je množina relací a
- I je množina integritních omezení.

Def.: Přípustná (relační) databáze:

Přípustná relační databáze se schématem (R, I) je množina (konkrétních) relací $\{R_1^*, R_2^*, .., R_n^*\}$ takových, že jejich n-tice vyhovují tvrzením v I.

Klíč schématu

Def.: Klíč schématu

Klíč K schématu $R(A)$ je taková **minimální** podmnožina atributů z A, která jednoznačně určí každou n-tici (konkrétní) relace R^*.

Pozn.: minimalita klíče (viz předchozí definice)

Požadavek na **minimalitu**, lze formulovat takto: neexistuje takové $K' \subset K$, které jednoznačně určuje každou n-tici $R(A)$.

Tvrzení 1:

Nechť K je klíč schématu $R(A)$. Pak pro každou přípustnou relaci R^* platí: jsou-li u a v dvě různé n-tice z R^*, potom $u[K] \neq v[K]$.

Tvrzení 2:

Relace R může mít několik (alternativních) klíčů.

Příklad: řekněme, že R(a,b,c) a R(a,b,c), pak ale už ne R(a,b,c),
protože a,b není minimální!

Integritní omezení

KINO (NazevK, Adresa)
FILM (JmenoF, Herec, Rok)
MA_NA_PROGRAMU(NazevK, JmenoF, Datum)

Integritní omezení:

- IO1 : primární klíče (podtržené)
- IO2 : cizí klíče:
  $MA\_NA\_PROGRAMU[JmenoF] \subseteq FILM[JmenoF]$
  $MA\_NA\_PROGRAMU[NazevK] \subseteq KINO[NazevK]$
- IO3 : V kině se nehraje více než dvakrát týdně.
- IO4 : Jeden film se nehraje více než ve třech kinech.

Integritní omezení a jejich ošetřování

1. Jak definovat IO nad schématem úložiště?
2. Jak zajistit dodržování daného IO?

Možností vyjádření (a kontroly) IO:

1. deklarativní:
   při vytvoření schématu (relace),
   hlídá je automaticky DBMS,
   výše jsme zavedli PK a FK,
   později v SQL přibudou UK, NOT NULL a CHECK.
2. procedurální na straně serveru
3. procedurální na straně klienta

Dotaz nad schématem

Def.: Dotaz

Databázový **dotaz** nad schématem S je výraz, který vrací odpověď se schématem T. Přičemž:

- definičním oborem jsou všechna úložiště se schématem S,
- oborem hodnot jsou všechny relace se schématem T,
- data v odpovědi pocházejí z databáze,
- odpověď nezávisí na fyzickém uložení dat.

Def.: Dotazovací jazyk

Dotazovací jazyk je množina všech použitelných výrazů pro konstrukci dotazu.

Manipulace s relacemi

- Vložení n-tice do dané relace.
- Zrušení/změna daných n-tic v dané relaci.
  $\Rightarrow$ Algoritmus manipulačních operací zahrnuje kontrolu dodržovaní (deklarativních) IO.
- Zadání dotazu:
  $\Rightarrow$ relační algebra,
  $\Rightarrow$ relační kalkul,
  $\Rightarrow$ SQL.

Relační model dat – shrnutí

- Relace (tabulky) jsou v 1NF,
  Domény atributů jsou atomické (toto omezení překračují ORDBMS).
- Jedinečné n-tice (řádky) v rámci relace.
  Relace jsou (definovány jako) množiny. V implementacích v RDBMS a ORDBMS toto neplatí.
- Vyhodnocení dotazu nezávisí na fyzickém uložení dat.
- Přístup k prvkům relace (řádkům tabulky) dle obsahu, ne podle pořadí.
- Silné prostředky pro dotazování.
  (relační algebra, relační kalkul, v implementacích RDBMS SELECT).
- Existují metody návrhu schématu úložiště v relační databázi, které vedou na schémata ”dobrých vlastností”.
  Tomu se bude věnovat přednáška o normalizaci schématu.

Selekce a projekce

- **selekce** (restrikce) relace R dle podmínky φ
  Značení: $R(\varphi)$. **Definice:** $R(\varphi) =_{def} \{u \mid u \in R \wedge \varphi(u)\}$,
  kde φ je (složený) logický výraz typu $(t_1 \Theta t_2)$ a/nebo $(t \Theta a)$,
  kde t zastupuje atribut, a konstantu a Θ relační operátor.
- **projekce** relace R na množinu atributů C, kde $C \subseteq A$
  Značení: $R[C]$. **Definice:** $R[C] =_{def} \{u[C] \mid u \in R\}$.

R(φ) [A1,A2,..,Aj]

SELECT DISTINCT A1, A2,...,Aj
FROM R WHERE φ

Přirozené spojení a přejmenování atributu

- **přirozené spojení** relací $R(A)$ a $S(B)$ s výsledkem $T(C)$
  Značení: $R * S$. **Definice:** $R * S =_{def} \{u \mid u[A] \in R \wedge u[B] \in S\}$, kde $C = A \bigcup B$ a kde výběr n-tic pro spojení je dán **rovností** na **všech** průnikových atributech A a B.

R1 * R2

```
SELECT DISTINCT *
FROM R1 NATURAL JOIN R2
```

- **přejmenování** atributu
  Značení: $t \rightarrow alias$.

R[A1 -> B1]

```
SELECT DISTINCT A1 as B1
FROM R
```

Obecné spojení

- **obecné spojení (Θ-spojení)** $R(A)$ a $S(B)$ s výsledkem $T(C)$
  Značení: $R[t_1 \Theta t_2]S$.
  Definice: $R[t_1 \Theta t_2]S =_{def} \{u \mid u[A] \in R, u[B] \in S, u.t_1 \Theta u.t_2\}$,
  kde $\Theta \in \{<, >, =, \leq, \geq, \neq\}$, přípustné je také použití logických spojek $\{\wedge, \vee, \neg\}$ pro konstrukci složitějších podmínek spojení.
  C je "zřetězením" A a B - spol. atributy ve výsledku se opakují, nutno je přejmenovat nebo používat kvalifikaci, např. $R.A$.

R1[t1 Θ t2]R2

```
SELECT DISTINCT *
FROM R1 JOIN R2 ON (R1.t1 Θ R2.t2)
```

Množinové operace

- množinové operace:
  sjednocení – $R \bigcup S$
  průnik – $R \bigcap S$
  množinový **rozdíl** – $R \setminus S$ (někdy též klasicky $R - S$)
  kartézský součin – $R \times S$

R1 $\bigcup$ R2 | R1 $\bigcap$ R2 | R1 $\setminus$ R2 | R1 $\times$ R2

SELECT DISTINCT * from R1
UNION | INTERSECT | MINUS (EXCEPT) | CROSS PRODUCT
SELECT DISTINCT * from R2

Minimální množina operací

Minimální množina operací RA: $\{\times, \textit{selekce}, \textit{projekce}, \rightarrow, \bigcup, \setminus\}$
Ostatní operace se dají zapsat s jejich pomocí.
Např. $*$ nebo Θ-spojení pomocí $\times$, *selekce*, *projekce*.

Příklad: Θ-spojení

Mějme relace $R(A, B, C)$ a $S(B, C, D, E)$. $T := R[A < S.B]S$.

R	A	B	C
	8	2	3
	1	2	3
	2	1	4
	3	6	7
	3	8	9

S	B	C	D	E
	3	4	2	3
	3	3	3	3
	1	4	5	6
	2	3	4	7

T	A	R.B	R.C	S.B	S.C	D	E
	1	2	3	3	4	2	3
	1	2	3	3	3	3	3
	1	2	3	2	3	4	7
	2	1	4	3	4	2	3
	2	1	4	3	3	3	3

Příklad: selekce

Rozeberme **po krocích** dotaz:

Hvězdy (herci), které hrají ve filmech promítaných v kině Mír.

Databáze:

KINO (NazevK, Adresa)
FILM (JmenoF, Herec, Rok)
MA_NA_PROGRAMU (NazevK, JmenoF, Datum)

IO1: $MA\_NA\_PROGRAMU[NazevK] \subseteq KINO[NazevK]$

IO2: $MA\_NA\_PROGRAMU[JmenoF] \subseteq FILM[JmenoF]$

$R1 := MA\_NA\_PROGRAMU(NazevK = 'Mir')$

```
SELECT DISTINCT * from MA_NA_PROGRAMU
 WHERE NazevK = 'Mir'
```

MA_NA_PROGRAMU	NazevK	JmenoF	Datum
	Blaník	Tog gun	29.3. 1994
	Blaník	Kmotr	8.3. 1994
	Mír	Nováček	10.3. 1994
	Mír	Top gun	9.3. 1994
	Mír	Kmotr	8.3. 1994

R1	NazevK	JmenoF	Datum
	Mír	Nováček	10.3. 1994
	Mír	Top gun	9.3. 1994
	Mír	Kmotr	8.3. 1994

Příklad: projekce

$R2 := R1[JmenoF, Datum]$

```
SELECT DISTINCT JmenoF, Datum from R1
```

R1	NazevK	JmenoF	Datum
	Mír	Nováček	10.3. 1994
	Mír	Top gun	9.3. 1994
	Mír	Kmotr	8.3. 1994

R2	JmenoF	Datum
	Nováček	10.3. 1994
	Top gun	9.3. 1994
	Kmotr	8.3. 1994

Příklad: přirozené spojení

$R3 := R2 * FILM$

```
SELECT DISTINCT * FROM R2 NATURAL JOIN FILM
```

R2	JmenoF	Datum
	Nováček	10.3. 1994
	Top gun	9.3. 1994
	Kmotr	8.3. 1994

FILM	JmenoF	Herec	Rok
	Černí baroni	Vetchý	1994
	Černí baroni	Landovský	1994
	Top gun	Cruise	1986
	Top gun	McGillis	1986
	Kmotr	Brando	1972
	Nováček	Brando	1990
	Vzorec	Brando	1980

R3	JmenoF	Datum	Herec	Rok
	Nováček	10.3. 1994	Brando	1990
	Top gun	9.3. 1994	Cruise	1986
	Top gun	9.3. 1994	McGillis	1986
	Kmotr	8.3. 1994	Brando	1972

Spojení bylo provedeno přes rovnost na atributech `JmenoF`.

Příklad: projekce a přejmenování

$R4 := R3[Herec \rightarrow Hvezda]$

```
SELECT DISTINCT Herec as Hvezda from R3
```

R3	JmenoF	Herec	Rok	Datum
	Nováček	Brando	1990	10.3. 1994
	Top gun	Cruise	1986	9.3. 1994
	Top gun	McGillis	1986	9.3. 1994
	Kmotr	Brando	1972	8.3. 1994

R4	Hvezda
	Cruise
	Brando
	McGillis

Poznámka:
Díky "množinovému" chápání RMD dojde **automaticky k vyloučení duplicit** v relaci R4.
SQL to ale samo neudělá (proto máme ve všech příkladech DISTINCT).

Příklad: přirozené spojení, selekce, projekce a přejmenování

Hvězdy (herci), které hrají ve filmech promítaných v kině Mír.

V relační algebře:

$\{\{MA\_NA\_PROGRAMU(NazevK = 'Mir')[JmenoF, Datum]\} * FILM\}[Herec \rightarrow Hvezda]$

lze zapsat jednodušeji (viz následující slide):

$\{MA\_NA\_PROGRAMU * FILM\}(NazevK = 'Mir')[Herec \rightarrow Hvezda]$

V SQL (pouze zjednodušená varianta):

```
SELECT DISTINCT Herec AS Hvezda
FROM MA_NA_PROGRAMU NATURAL JOIN FILM
WHERE NazevK = 'Mir'
```

Vyhodnocování výrazu v RA – priority

- vyhodnocuje se zleva doprava
- operace selekce a projekce mají přednost před binárními operacemi
- pro změnu priority používáme složené závorky

Hvězdy (herci), které hrají ve filmech promítaných v kině Mír

$\{\{MA\_NA\_PROGRAMU(NazevK = 'Mir')[JmenoF, Datum]\} * FILM\}$
$[Herec \rightarrow Hvezda]$

lze zapsat i takto:

$\{MA\_NA\_PROGRAMU(NazevK = 'Mir')[JmenoF, Datum] * FILM\}$
$[Herec \rightarrow Hvezda]$

a vysledek bude stejný jako v tomto případě:

$\{MA\_NA\_PROGRAMU * FILM\}\,(NazevK = 'Mir')[Herec \rightarrow Hvezda]$

Obecné polospojení

Předpokládejme R(A), S(B)

- **levé Θ-polospojení**
  Značení: $R <t_1 \Theta t_2] S$
  Definice: $R <t_1 \Theta t_2] S =_{def} \{R[t_1 \Theta t_2] S\}[A]$

v SQL:

```
SELECT distinct R.*
FROM R JOIN S ON (t1Θt2)
```

- **pravé Θ-polospojení**
  Značení: $R[t_1 \Theta t_2> S$
  Definice: $R[t_1 \Theta t_2> S =_{def} \{R[t_1 \Theta t_2] S\}[B]$

v SQL:

```
SELECT distinct S.*
FROM R JOIN S ON (t1Θt2)
```

Přirozené polospojení

Předpokládejme R(A), S(B)

- **levé přirozené polospojení**
  Značení: $R <* S$
  Definice: $R <* S =_{def} \{R * S\}[A]$

v SQL (pouze symbolicky, nebude fungovat v praxi)

```
SELECT distinct R.*
FROM R NATURAL JOIN S
```

- **pravé přirozené polospojení**
  Značení: $R *> S$
  Definice: $R *> S =_{def} \{R * S\}[B]$

v SQL(pouze symbolicky, nebude fungovat v praxi)

```
SELECT distinct S.*
FROM R NATURAL JOIN S
```

Využití polospojení

- Polospojení (obecné i přirozené) lze chápat jako “syntaktickou zkratku”:
- spojení následované projekcí na A resp. B
- Skutečná implementace je obvykle efektivnější než spojení a selekce.

Interpretace $R <* S$

Podmnožina n-tic z R, které jsou **spojitelné** s nějakou n-ticí z S.

- Kina, která něco hrají (jsou v programu).
- Filmy, které se hrají (jsou v programu).
- ...

Antijoin

Motivace:
Tato operace se používá v modulu query executor v Oracle.
Můžeme ji vidět v prováděcích plánech SQL dotazů při vyhodnocení klauzule NOT EXISTS.
Značení: $R \overline{<*} S$
Definice: $R \overline{<*} S =_{def} R \setminus \{R <* S\}$

Interpretace $R \overline{<*} S$

Podmnožina n-tic z R, které **nejsou spojitelné** s žádnou n-ticí z S.

Příklad: Najděte kina, která nic nejhrají.

$KINO \overline{<*} MA\_NA\_PROGRAMU$
Bez operace antijoin: $KINO \setminus \{KINO <* MA\_NA\_PROGRAMU\}$

Příklady na relační algebru 1/6

D1: Názvy kin, která něco hrají.

$MA\_NA\_PROGRAMU[NazevK]$

D2: Seznam kin, která nic nehrají.

$KINO[NazevK] \setminus \{MA\_NA\_PROGRAMU[NazevK]\}$

D3: Názvy kin, která hrají film Top Gun.

$MA\_NA\_PROGRAMU(JmenoF = 'TopGun')[NazevK]$

D4: Jména filmů, které hraje kino s adresou Zvonková.

$\{KINO(Adresa = 'Zvonkova')[NazevK] * MA\_NA\_PROGRAMU\}[JmenoF]$

Příklady na relační algebru 2/6

D5: Názvy kin, která hrají něco s hercem Brando.

$$D5 := \{FILM(Herec = \text{'Brando'})[JmenoF] * MA\_NA\_PROGRAMU\}[NazevK]$$

D6: Názvy kin, která nedávají žádný film s Brando.

Přeformulujme dotaz D6 na **D6':**

Názvy všech kin kromě těch, která hrají něco s hercem Brando.

$$KINO[NazevK] \setminus D5$$

D6B: Názvy kin, která hrají, ale nehrají žádný film s Brando.

$$MA\_NA\_PROGRAMU[NazevK] \setminus D5$$

Příklady 3/6: negace ($\neg$) a exist. kvant. ($\exists$)

D7: Názvy kin, ve kterých se **ne**hraje **některý** film s Brando.

- 1.krok :
  $\Rightarrow$ Univerzum pro množinu dvojic {kino,“film s Brando”}.
  $U := KINO[NazevK] \times \{FILM(Herec = \ 'Brando')[JmenoF]\}$
- 2.krok :
  $\Rightarrow$ Získejme reálnou množinu dvojic {kino, film }.
  $R := MA\_NA\_PROGRAMU[NazevK, JmenoF]$
- 3.krok :
  $\Rightarrow$ Odečtením výše zkonstruovaných relací získáme množinu v programu nezrealizovaných dvojic {kino,“film s Brando”}.
  $N := U \setminus R$
- 4.krok :
  $\Rightarrow$ Projekcí na první atribut získáme požadovaný seznam kin.
  $N[NazevK]$

Poznámka: Opravdu není nutné omezovat R na filmy s Brando.

Příklady 4a/6: všeobecný kvantifikátor (∀)

Princip formulace dotazů se všeobecným kvantifikátorem.

$$\forall x.P(x) = \neg\exists x.(\neg P(x))$$

D8: Názvy kin, ve kterých se dávají všechny filmy s Brando.

Přepis 1 **D8':**

Názvy kin, která něco hrají, a zároveň **není pravda**, že **ne**hrají **některý** film s Brando.

Přepis 2 **D8":**

Seznam kin, ve kterých něco hrají s výjimkou těch kin, která **ne**hrají **některý** film s Brando.

Přepis 3 **D8"':**

{*NazevK* | kino něco hraje} \
{*NazevK* | kino **ne**hraje **některý** film s Brando}

Příklady na relační algebru 4b/6

D8”’: {*NazevK* | kino něco hraje} \
{*NazevK* | kino **ne**hraje **některý** film s Brando}

- 1. krok: Názvy kin, která něco hrají.
  $K := MA\_NA\_PROGRAMU[NazevK]$
- 2. krok: Univerzum dvojic {“kino s programem”,“film s Brando”}.
  $U := K \times \{FILM(Herec = \text{'Brando'})[JmenoF]\}$
- 3. krok: Reálná množina dvojic {kino, film }.
  $R := MA\_NA\_PROGRAMU[NazevK, JmenoF]$
- 4.krok: Nerealizované {“kino s programem”,“film s Brando”}.
  $N := U \setminus R$
- 5. krok: Kina, která **ne**hrají **některý** film s Brando.
  $B := N[NazevK]$
- 6. krok: Konečný výsledek.
  $K \setminus B$

Příklady 4c/6: relační dělení ($\div$)

Mějme $R(x, y)$ a $S(y)$.

Značení: $R \div S$

Definice: $R \div S =_{def} R[x] \setminus \{\{R[x] \times S\} \setminus R\}[x]$

Interpretace $R \div S$

1. Výsledkem jsou všechny hodnoty x z R, které v R tvoří dvojici s **každým** prvkem y z S.
2. Pomocí prvků y z S se snažíme **diskvalifikovat** prvky x z R. Prvek x je diskvalifikován, pokud v R neexistuje ve dvojici s **každým** y z S. Výsledkem $R \div S$ jsou ty prvky x z R, které se diskvalifikovat nepodařilo.

D8”’: {*NazevK* | kino něco hraje} $\setminus$
{*NazevK* | kino **ne**hraje **některý** film s Brando}

$$MA\_NA\_PROGRAMU[NazevK, JmenoF] \div \{FILM(Herec = \text{'Brando'})[JmenoF]\}$$

Pozor na formulace dotazů

Je třeba správně porozumět dotazu, abyste nasadili správnou "šablonu". Příklady:

- Kina, která hrají některý (alespoň jeden) film s Brando.
- Kina, která nehrají žádný film s Brando. Mají/nemají být zahrnuta kina, která nehrají vůbec?
- Kina, která hrají pouze filmy s Brando. (kategorie C v semestrálce)
- Kina, která nehrají některý film s Brando.
- Kina, která hrají každý (všechny) film s Brando. (kategorie D v semestrálce)

Všechny tyto typy dotazů si vyzkoušíte na semestrálce a budou i předmětem semestrálního a zkouškového testu. Další konkrétní příklady na proseminářích.

Příklady na relační algebru 5/6

D9: Názvy kin, ve kterých se hrají **všechny** filmy, které jsou na programu.

D9': Názvy kin, ve kterých něco hrají, a zároveň **není** pravdou, že **ne**hrají **některý** z filmů z programu.

D9'': Názvy kin, ve kterých něco hrají s výjimkou kin, kde **ne**hrají **některý** z filmů v programu.

D9''': $\{NazevK \mid \text{kino něco hraje}\} \setminus \{NazevK \mid \text{kino } \mathbf{ne}\text{hraje } \mathbf{některý} \text{ film z programu}\}$

1. bez použití operace $\div$
   - 1. krok: $K := MA\_NA\_PROGRAMU[NazevK]$
   - 2. krok: $U := K \times \{MA\_NA\_PROGRAMU[JmenoF]\}$
   - 3. krok: $R := MA\_NA\_PROGRAMU[NazevK, JmenoF]$
   - 4. krok: $N := U \setminus R$
   - 5. krok: $B := N[NazevK]$
   - 6. krok: $K \setminus B$
2. s pomocí operace $\div$
   $MA\_NA\_PROGRAMU[NazevK, JmenoF] \div \{MA\_NA\_PROGRAMU[JmenoF]\}$

Příklady na relační algebru 6/6

D10: Název kina,
ve kterém se hrají **všechna** představení z programu.
Poznámka: *Výsledek je buď jedno kino, nebo nic.*
D10': Název kina, pro které **není** pravda,
že **ne**hraje **některé** přestavení z programu.
D10'': Názvy kin, která něco hrají s výjimkou kin,
ve kterých se **ne**hraje **některé** představení z programu.
D10''': {*NazevK* | kino něco hraje} \
{*NazevK* | kino **ne**hraje **některé** představení z programu}

$$MA\_NA\_PROGRAMU[NazevK, JmenoF, Datum] \div \{MA\_NA\_PROGRAMU[JmenoF, Datum]\}$$

Relační algebra a programování

- Programování vs. relační algebra
  - Relační algebra je “vyšší” jazyk.
  - Nespecifikujeme “jak se mají věci udělat” (typicky 3GL), ale “co má být výsledkem”.
  - Na druhou stranu je to jazyk velmi úzce specializovaný.
- Dotazovací jazyk, který umožňuje realizovat relační algebru, se nazývá **relačně úplný**.
- Komerční svět
  - SQL SELECT (je relačně úplný)
  - jazyky formulářů (nebývají relačně úplné)
  - obrázkové jazyky (nebývají relačně úplné)

Charakteristika relační algebry

- Řeší “pouze“ dotazování nikoliv DML a DDL.
- Výsledkem dotazu je relace, která může být vstupem do dalšího dotazu – dotazy lze řetězit.
- Výrazy (dotazy) se skládají z operací a operandů.
- Operandem je vždy celá relace.
- Relační algebra byla vzorem pro návrh příkazu SELECT v SQL.
- SELECT má aktuálně větší vyjadřovací možnosti než RA (agregace, vnější spojení, vnořené dotazy, řazení, ...).

Shrnutí

- Relace je množina n-tic.
- IO: klíč (primární, alternativní), cizí klíč.
- atributy v relačním modelu jsou atomické
- Relační algebra a její operace:
  - selekce, projekce, přejmenování
  - množinové operace
  - polospojení, antijoin
  - negace, kvantifikátory
  - relační dělení
- RA je dotazovací jazyk (neřeší DML a DDL)