Databázové systémy

Relační model dat

Vilém Vychodil

KMI/DATA1, Přednáška 2

Databázové systémy

Přednáška 2: Přehled

1. Základní pojmy relačního modelu:
   - přehled Coddova modelu,
   - komponenty relačního modelu,
   - datové tabulky, první normální forma.
2. Formalizace pojmů z relačního modelu:
   - atributy, typy, relační schémata,
   - obecné kartézské součiny,
   - relace nad relačními schématy,
   - relační proměnné, klíče, relační přiřazení.
3. Jazyky Tutorial D a SQL:
   - skalární, n-ticové a relační typy,
   - definice relačních proměnných,
   - manipulace s daty (relační přiřazení),
   - vybrané nerelační rysy jazyka SQL.

Něco úvodem

"A hundred years from now, I'm quite sure, database systems will still be based on Codd's relational foundation." — C. J. Date

"The relational approach really is rock solid, owing (once again) to its basis in mathematics and predicate logic." — C. J. Date

"The point about principles is this: they endure. By contrast, products and technologies (and the SQL language, come to that) change all the time—but principles don't." — C. J. Date

"I could imagine how those queries would have been represented in CODASYL by programs that were five pages long that would navigate through this labyrinth of pointers and stuff. Codd would sort of write them down as one-liners." — D. Chamberlin

Relační model dat (E. F. Codd, 1969)

- logická nezávislost dat
- integritní omezení
- manipulativní formalismy
- dotazovací formalismy
- základí pojem = relace

Významní lidé

Codd

Darwen

Date

Fagin

Maier
a mnoho dalších: Vardi, Beeri, ...

Relační model dat (C. J. Date & H. Darwen)

relační model dat, angl.: *relational model of data*

Relační model se skládá z pěti komponent:

1. kolekce **skalárních typů** zahrnující typ „pravdivostní hodnota“ (angl.: *boolean*);
2. systém pro **generování relačních typů**;
3. prostředky pro definici **relačních proměnných** daných relačních typů;
4. operátor pro **relační přiřazení**, tj. přiřazení hodnot relačním proměnným;
5. kolekce generických **relačních operací** pro vyjadřování relací z jiných relací.

poznámky o rolích komponent:

- kolekce skalárních typů a relačních operací jsou „otevřené“ (ne pevně dané)
- relace = hodnoty; relační proměnné = jména (nabývající hodnot)
- prostředky pro definici dat (1–3), modifikaci dat (4–5) a dotazování (3, 5)

Struktury relačního model dat (neformálně)

Neformálně lze chápat (instanci) relační databáze jako kolekci pojmenovaných datových tabulek spolu s množinou integritních omezení.

tabulky mají dvě části:

1. **záhlaví** (*metadata tabulky*), definuje:
   - jména sloupců
   - typy hodnot přípustných ve sloupcích
2. **tělo** (*data tabulky*)
   - skládá se z řádků (n-tic hodnot)
   - průsečíky řádků a sloupců obsahují hodnoty

jmeno	id	rodne-cislo
Adams	12345	571224/4023
Black	33355	840525/6670
Chang	66066	891117/1024

poznámky:

- ne každá tabulka splňující předchozí je „uspokojivou strukturou“ v RM
- cílem je používat tabulky, které jsou formalizovatelné jako n-ární relace

Požadavky na tabulky

motto:

> *A table is normalized—equivalently, it is in the first normal form, 1NF—if and only if it is a direct and faithful representation of some relation. — C. J. Date*

první normální forma, angl.: *first normal form*

Tabulka je v první normální formě (1NF), pokud splňuje následující podmínky:

1. neuvažuje se **žádné uspořádání řádků** shora-dolů,
2. neuvažuje se **žádné uspořádání sloupců** zleva-doprava,
3. v tabulce se nevyskytují **žádné duplicitní řádky**,
4. v každém vnitřním poli tabulky je **právě jedna hodnota daného typu**,
5. všechny **atributy jsou regulární**.

poznámka: regularita = řádky (skrytě) neobsahují žádnou extra informaci, která je dostupná (pouze) prostřednictvím speciálních funkcí

Chybné chápání 1NF

V literatuře se často vyskytují pseudodefinice:

„Tabulka je v 1NF, když jsou všechny její hodnoty jsou atomické.“

problémy:

- co je „atomická hodnota“ je vágní pojem (*ad absurdum*: jeden bit?)
- nepokrývá patologické případy (např. duplicitní řádky nebo absenci hodnot)
- z dnešního pohledu je atomicita směšná (řetězce, multimediální data, . . .)
- cílem „pseudodefinic“ někdy je „zamezit možnosti uvažovat vnořené tabulky“ (to jest tabulky jako hodnoty); principiálně ale není důvod to zakazovat

definice, kterou preferujeme:

Darwen H.: Relation-Valued Attributes; or, Will the Real First Normal Form Please Stand Up?
In: Date C. J., Darwen H., *Relational Database Writings 1989–1991*
Addison-Wesley 1992, ISBN 978–0201543032

Základní pojmy RM: Atribut

základní pojem použitý pro formalizaci „záhlaví tabulky“:

atribut, angl.: *attribute*

Atributy jsou symbolická jména. Množinu všech atributů, o které předpokládáme, že je nekonečná a spočetná, označujeme Y.

poznámky:

- značení:
  1. $y, y', y_1, y_2, \ldots$ (označení obecných úvahách o RM) nebo
  2. `name`, `id`, `foo`, ... (konkrétní atributy v příkladech)
- *intuitivní interpretace:* atribut je jméno, které může být použito pro „označení sloupce“ datové tabulky
- atribut je čistě *syntaktický pojem* (sám o sobě nemá žádnou hodnotu)

Základní pojmy RM: Typ (doména)

motivace:

RM je silně typovaný, každý atribut (relace, . . .) má přiřazen svůj typ tak, aby bylo možné provádět operace pouze s hodnotami povolených typů.

typ, angl.: *type*

Typ je pojmenovaná (nejvýš spočetná) množina elementů (hodnot).

poznámky:

- ekvivalentní název pro *typ* je *doména* (historický starší, Codd)
- alternativní chápání pojmů typ/doména (někdy užitečné, my nevyužijem):
  - **typ** je *symbolické jméno* pro množinu hodnot (syntaktický pojem)
  - **doména** je *interpretace typu* – množina konkrétních hodnot (sémantický pojem)
- **rozlišitelnost hodnot**: hodnoty daného typu lze porovnávat pomocí „=“
  – porovnávat hodnoty různých typů nelze (**!!**)

Základní pojmy RM: Relační schéma

formalizace pojmu „záhlaví tabulky“:

relační schéma, angl.: *relation scheme*

Konečná množina $R = \{\langle y_1, \tau_1 \rangle, \ldots, \langle y_n, \tau_n \rangle\}$, kde $y_1, \ldots, y_n$ jsou vzájemně různé atributy z Y a $\tau_1, \ldots, \tau_n$ jsou typy, se nazývá relační schéma.

poznámky:

- značení: $R, R', R_1, R_2, \ldots, S, S', S_1, S_2, \ldots$
- relační schéma může být prázdné, to jest $R = \emptyset$ (existuje právě jedno)
- stejně jako hodnoty atributů, i relace mají svůj *typ* (relační schéma)

úmluva o zjednodušeném značení relačních schémat

Pokud typy (domény) atributů vyplývají jasně z kontextu, pak relační schémata ztotožňujeme s konečnými podmnožinami $R \subseteq Y$; typ (doménu) $y \in Y$ značíme D_y.

Intermezzo: Opakování užitých pojmů

- **množina** – chápeme naivně, budeme uvažovat nejvýš spočetné
- **uspořádaná dvojice** – značíme $\langle a, b \rangle$, pozor: $\langle 10, 20 \rangle \neq \langle 20, 10 \rangle$
- **(naivní) kartézský součin** množin A a B je množina $A \times B$ všech uspořádaných dvojic prvků jejichž první složky jsou prvky z A a druhé složky jsou prvky z B, to jest $A \times B = \{\langle a, b \rangle \mid a \in A \text{ a } b \in B\}$
- **relace mezi množinami** A a B je libovolná podmnožina $A \times B$
- **zobrazení** je množina $f \subseteq A \times B$ taková, že pro každý prvek $a \in A$ existuje právě jedno $b \in B$ tak, že $\langle a, b \rangle \in f$ (což označumeme $f(a) = b$)

poznámky:

- fakt, že $f \subseteq A \times B$ je zobrazení píšeme $f \colon A \to B$
- $f \subseteq A \times B$ je zobrazení p. k. následující podmínky jsou (obě) splněny:
  1. pro každé $a \in A$ existuje $b \in B$ tak, že platí $\langle a, b \rangle \in f$;
  2. pro každé $a \in A$ a $b_1, b_2 \in B$ platí: pokud $\langle a, b_1 \rangle \in f$ a $\langle a, b_2 \rangle \in f$, pak $b_1 = b_2$.

Intermezzo: Obecný kartézský součin

Relace v RM jsou formalizovány jako konečné podmnožiny obecných kartézských součinů, ve kterých jsou n-tice hodnot formalizované zobrazeními.

Kartézský součin, angl.: *Cartesian product*

Mějme systém množin A_i, které jsou indexovány prvky z množiny I (tzv. **indexy**). **Kartézský součin množin** A_i ($i \in I$) je množina $\prod_{i \in I} A_i$ všech zobrazení $f : I \to \bigcup_{i \in I} A_i$ takových, že $f(i) \in A_i$ platí pro každý index $i \in I$.

Každé zobrazení $f \in \prod_{i \in I} A_i$ se nazývá ***n*-tice** (angl.: *tuple*).

poznámka:

- pro $f \in \prod_{i \in I} A_i$ se $f(i) \in A_i$ nazývá *hodnota i v n-tici f*
- český pojem n-tice je matoucí, protože n nemá vztah k I ani k A_i (**!!**)
- pokud je $I = \{1, \ldots, n\}$, pak lze psát $\prod_{i=1}^{n} A_i$
- důležité: indexy z množiny I nemají žádné „pořadí“ (**!!**)

Příklad (Příklad kartézského součinu)

Pro $A_1 = \{5, 7, 9\}$ $A_2 = \{\mathsf{a}, \mathsf{b}\}$ $A_3 = \{\clubsuit, \diamondsuit, \heartsuit, \spadesuit\}$ a $I = \{1, 2, 3\}$ máme:

$$\begin{aligned}\prod_{i \in I} A_i = \{&\{\langle 1,5\rangle, \langle 2,\mathsf{a}\rangle, \langle 3,\clubsuit\rangle\}, \{\langle 1,7\rangle, \langle 2,\mathsf{a}\rangle, \langle 3,\clubsuit\rangle\}, \{\langle 1,9\rangle, \langle 2,\mathsf{a}\rangle, \langle 3,\clubsuit\rangle\},\\ &\{\langle 1,5\rangle, \langle 2,\mathsf{b}\rangle, \langle 3,\clubsuit\rangle\}, \{\langle 1,7\rangle, \langle 2,\mathsf{b}\rangle, \langle 3,\clubsuit\rangle\}, \{\langle 1,9\rangle, \langle 2,\mathsf{b}\rangle, \langle 3,\clubsuit\rangle\},\\ &\{\langle 1,5\rangle, \langle 2,\mathsf{a}\rangle, \langle 3,\diamondsuit\rangle\}, \{\langle 1,7\rangle, \langle 2,\mathsf{a}\rangle, \langle 3,\diamondsuit\rangle\}, \{\langle 1,9\rangle, \langle 2,\mathsf{a}\rangle, \langle 3,\diamondsuit\rangle\},\\ &\{\langle 1,5\rangle, \langle 2,\mathsf{b}\rangle, \langle 3,\diamondsuit\rangle\}, \{\langle 1,7\rangle, \langle 2,\mathsf{b}\rangle, \langle 3,\diamondsuit\rangle\}, \{\langle 1,9\rangle, \langle 2,\mathsf{b}\rangle, \langle 3,\diamondsuit\rangle\},\\ &\{\langle 1,5\rangle, \langle 2,\mathsf{a}\rangle, \langle 3,\heartsuit\rangle\}, \{\langle 1,7\rangle, \langle 2,\mathsf{a}\rangle, \langle 3,\heartsuit\rangle\}, \{\langle 1,9\rangle, \langle 2,\mathsf{a}\rangle, \langle 3,\heartsuit\rangle\},\\ &\{\langle 1,5\rangle, \langle 2,\mathsf{b}\rangle, \langle 3,\heartsuit\rangle\}, \{\langle 1,7\rangle, \langle 2,\mathsf{b}\rangle, \langle 3,\heartsuit\rangle\}, \{\langle 1,9\rangle, \langle 2,\mathsf{b}\rangle, \langle 3,\heartsuit\rangle\},\\ &\{\langle 1,5\rangle, \langle 2,\mathsf{a}\rangle, \langle 3,\spadesuit\rangle\}, \{\langle 1,7\rangle, \langle 2,\mathsf{a}\rangle, \langle 3,\spadesuit\rangle\}, \{\langle 1,9\rangle, \langle 2,\mathsf{a}\rangle, \langle 3,\spadesuit\rangle\},\\ &\{\langle 1,5\rangle, \langle 2,\mathsf{b}\rangle, \langle 3,\spadesuit\rangle\}, \{\langle 1,7\rangle, \langle 2,\mathsf{b}\rangle, \langle 3,\spadesuit\rangle\}, \{\langle 1,9\rangle, \langle 2,\mathsf{b}\rangle, \langle 3,\spadesuit\rangle\}\}.\end{aligned}$$

poznámka: každá množina $\{\langle 1, \ldots\rangle, \langle 2, \ldots\rangle, \langle 3, \ldots\rangle\}$ zeprezentuje jedno zobrazení

$$f\colon \{1, 2, 3\} \to \{5, 7, 9, \mathsf{a}, \mathsf{b}, \clubsuit, \diamondsuit, \heartsuit, \spadesuit\}$$

takové, že $f(1) \in \{5, 7, 9\}$, $f(2) \in \{\mathsf{a}, \mathsf{b}\}$ a $f(3) \in \{\clubsuit, \diamondsuit, \heartsuit, \spadesuit\}$.

Příklad (Speciální případy kartézských součinů)

kartézský součin dvou množin, $|I| = 2$:

- pro $I = \{1, 2\}$, každé $f \in \prod_{i \in I} A_i$ formalizuje *dvojici hodnot* z množin A_1 a A_2
- *pozor:* $A_1 \times A_2$ (*naivní kartézský součin*) je množina uspořádaných dvojic, kdežto $\prod_{i \in \{1,2\}} A_i$ je množina *zobrazení* reprezentujících tyto dvojice

kartézský součin jedné množiny, $|I| = 1$:

- každá $f \in \prod_{i \in \{x\}} A_i$ je *konstantní funkce*, kde $f(x) \in A_x$ pro $I = \{x\}$
- f je funkce vybírající prvek z množiny A_x,
- někdy *ztotožňujeme* $\prod_{i \in \{x\}} A_i = \{\{\langle x, a \rangle\} \mid a \in A_x\}$ a A_x (!!)

mezní případ pro $I = \emptyset$:

- $f = \emptyset$ je jediné zobrazení $f : \emptyset \to \emptyset$, protože $f \subseteq \emptyset \times \emptyset = \emptyset$
- $\prod_{i \in \emptyset} A_i = \{\emptyset\}$, $\emptyset$ lze interpretovat jako „nultici hodnot“ (!!)

poznámka: lze uvažovat i nekonečené I (uplatnění v algebře, my nebudeme používat)

Základní pojmy RM: Relace nad relačním schématem

Definice (relace nad relačním schématem, angl.: *relation*)

Mějme relační schéma $R \subseteq Y$ a nechť D_y ($y \in Y$) označují domény atributů $y \in R$. **Relace** $\mathcal{D}$ nad relačním schématem R je libovolná konečná podmnožina $\prod_{y \in R} D_y$. Číslo $|\mathcal{D}|$ se nazývá **velikost relace** $\mathcal{D}$, číslo $|R|$ se nazývá **stupeň relace** $\mathcal{D}$.

značení:

- relace označujeme $\mathcal{D}$, $\mathcal{D}'$, $\mathcal{D}_1$, $\mathcal{D}_2, \ldots$

poznámky:

- $\prod_{y \in R} D_y$ je *kartézský součin domén* (indexy $=$ atributy z R)
- $r \in \prod_{y \in R} D_y$ je n-*tice*, to jest $r(y)$ je prvek z D_y
- každá n-tice $r \in \mathcal{D}$ reprezentuje jeden „řádek“ v tabulce odpovídající $\mathcal{D}$
- zřejmě: tabulka je 1NF p. k. reprezentuje relaci na relačním schématu
- *nulární* relace (stupně 0), *unární* relace (stupně 1), *binární* relace (stupně 2), ...

Příklad (Matematické relace × relace nad relačními schématy)

$A = \{1, 2, \dots\}$ (množina přirozených čísel),
$B = \{\texttt{single}, \texttt{married}, \texttt{divorced}, \texttt{widowed}\}$ (doména nominálních hodnot),
C je množina všech řetězců nad abecedou znaků.

Příklad *ternární relace* mezi množinami A, B, C (v tomto pořadí):

$$\{\langle 3, \texttt{single}, \texttt{"Abbe"}\rangle, \langle 2, \texttt{married}, \texttt{"Blangis"}\rangle,$$
$$\langle 3, \texttt{married}, \texttt{"Curval"}\rangle, \langle 4, \texttt{divorced}, \texttt{"Durcet"}\rangle\} \subseteq A \times B \times C$$

Pokud chápeme A, B a C jako domény atributů CHILDREN, STATUS a NAME, pak lze předchozí formalizovat jako relaci $\{t_1, t_2, t_3, t_4\}$ na schématu
$R = \{\texttt{CHILDREN}, \texttt{STATUS}, \texttt{NAME}\}$, kde

$t_1(\texttt{CHILDREN}) = 3$,	$t_1(\texttt{STATUS}) = \texttt{single}$,	$t_1(\texttt{NAME}) = \texttt{"Abbe"}$,
$t_2(\texttt{CHILDREN}) = 2$,	$t_2(\texttt{STATUS}) = \texttt{married}$,	$t_2(\texttt{NAME}) = \texttt{"Blangis"}$,
$t_3(\texttt{CHILDREN}) = 3$,	$t_3(\texttt{STATUS}) = \texttt{married}$,	$t_3(\texttt{NAME}) = \texttt{"Curval"}$,
$t_4(\texttt{CHILDREN}) = 4$,	$t_4(\texttt{STATUS}) = \texttt{divorced}$,	$t_4(\texttt{NAME}) = \texttt{"Durcet"}$.

Příklad (Vizualizace relace z předchozího příkladu)

relace z předchozího příkladu zapsaná jako tabulka:

NAME	STATUS	CHILDREN
Abbe	single	3
Blangis	married	2
Curval	married	3
Durcet	divorced	4

$=$

CHILDREN	NAME	STATUS
2	Blangis	married
3	Abbe	single
4	Durcet	divorced
3	Curval	married

$= \cdots$

množinová reprezentace předchozí relace:

$$\begin{aligned}\mathcal{D} = \{&\{\langle \texttt{NAME}, \texttt{"Abbe"}\rangle, \langle \texttt{CHILDREN}, 3\rangle, \langle \texttt{STATUS}, \texttt{single}\rangle\},\\ &\{\langle \texttt{NAME}, \texttt{"Blangis"}\rangle, \langle \texttt{CHILDREN}, 2\rangle, \langle \texttt{STATUS}, \texttt{married}\rangle\},\\ &\{\langle \texttt{NAME}, \texttt{"Curval"}\rangle, \langle \texttt{CHILDREN}, 3\rangle, \langle \texttt{STATUS}, \texttt{married}\rangle\},\\ &\{\langle \texttt{NAME}, \texttt{"Durcet"}\rangle, \langle \texttt{CHILDREN}, 4\rangle, \langle \texttt{STATUS}, \texttt{divorced}\rangle\}\}\end{aligned}$$

Zajímavé případy relací

singleton – jednoprvková unární relace (to jest $|\mathcal{D}| = 1$, $|R| = 1$)

$\mathcal{D} = \{\{\langle y, d\rangle\}\}$

y
d

kde $R = \{y\}$ a $d \in D_y$

prázdná relace – relace velikosti 0 (to jest $|\mathcal{D}| = 0$)

$\mathcal{D} = \emptyset$

y_1	y_2	$\cdots$	y_n

kde $R = \{y_1, y_2, \ldots, y_n\}$

relace nad prázdným schématem – relace stupně 0 (to jest $R = \emptyset$);

1. $\mathcal{D}_\top = \{\emptyset\}$ (`TABLE_DEE`, relační reprezentace *logické pravdy*)
2. $\mathcal{D}_\bot = \emptyset$ (`TABLE_DUM`, relační reprezentace *logické nepravdy*)

značení: názvy `TABLE_DEE` a `TABLE_DUM` zavedl C. J. Date, viz

Carroll L.: *Through the Looking-Glass, and What Alice Found There*, Macmillan 1871

Příklad (Relace jako hodnoty v jiných relacích)

NAME	YEAR	SUBJ	SCORE	
			VAL	DATE
Abbe	2011	KMI/DATA1	34%	18/12/11
			84%	25/01/12
Blangis	2010	KMI/DATA1	VAL	DATE
			25%	19/12/10
			34%	13/01/11
			57%	24/01/11
Curval	2010	KMI/DATA1	VAL	DATE
			21%	19/12/10
Curval	2013	KMI/DATA1	VAL	DATE
Durcet	2010	KMI/DATA2	VAL	DATE
			95%	18/01/13
			34%	25/06/13

tabulka je v 1NF pokud:

- NAME je typu „*řetězec*“
- YEAR je typu „*rok*“ (*číslo*)
- SUBJ je typu „*řetězec*“
- SCORE je typu relace nad schématem $\{\texttt{VAL}, \texttt{DATE}\}$
- VAL je typu „*procento*“
- DATE je typu „*datum*“

Skalární, n-ticové a relační typy

tři kategorie typů v relačním modelu:

1. **relační typy** (angl.: *relation types*)
   - jsou definované relačními schématy
   - hodnoty jsou relace nad relačními schématy
2. **n-ticové typy** (angl.: *tuple types*)
   - jsou definované relačními schématy (případně pouze syntakticky odlišené)
   - hodnoty jsou n-tice nad relačními schématy
3. **skalární typy** (angl.: *scalar types*)
   - všechny ostatní typy a jejich hodnoty

poznámka:

Tradiční chápání „skalárních“ a „neskalárních“ typů a hodnot (hodnoty skalárního typu nemají pro uživatele žádné „viditelné složky“) je analogicky vágní jako pseudodefinice 1NF.

Relační proměnná

formalizace „jmen a typů tabulek“ v RM:

relační proměnná, angl.: *relation variable*, zkráceně RELVAR

Relační proměnná daného typu je (perzistentní) proměnná, která může nabývat hodnot, jimiž jsou relace daného typu.

rozlišujeme:

- **základní** relační proměnné (angl.: *base relvar*)
  - mají definovaný svůj *typ* (relační schéma) a *množinu klíčů* (viz dále)
- **pohledy** (angl.: *virtual relvar, view*)
  - hodnoty vznikají vyhodnocením relačních výrazů (v daném čase)
  - jejich typ je určen typem daného relačního výrazu
  - budeme se jimi zabývat později

poznámka: různé jazyky (SQL, Tutorial D) řeší koncept „relační proměnné“ různě

Klíč základní relační proměnné a relační přiřazení

klíč, angl.: *key*

Uvažujme relační proměnnou X typu $R = \{y_1, \ldots, y_n\}$. Množina klíčů proměnné X je libovolná neprázdná množina $\{K_1, \ldots, K_n\}$ jejíž prvky jsou podmnožiny R a splňují podmínku, že $K_i \not\subseteq K_j$ pro každé $i \neq j$.

relační přiřazení, angl.: *relational assignment*

Mějme relční proměnnou X typu R a nechť $\{K_1, \ldots, K_n\}$ je množina klíčů proměnné X. Pak relaci $\mathcal{D}$ typu R lze **přiřadit jako hodnotu** proměnné X pokud je splněna následující podmínka: Pro každé $i = 1, \ldots, n$ a libovolné $r_1, r_2 \in \mathcal{D}$ platí:

$$\text{pokud } r_1(y) = r_2(y) \text{ pro každý } y \in K_i, \text{ pak } r_1 = r_2.$$

V opačném případě říkáme, že $\mathcal{D}$ porušuje integritní omezení dané některým klíčem relační proměnné X.

poznámka: relační přiřazení = fundamentální *prostředek pro manipulaci s daty*

Jazyk Tutorial D

rysy:

- programovací jazyk, staticky a silně typovaný (bez koercí)
- relace jsou objekty prvního řádu (se vším všudy)
- specifikace se průběžně aktualizuje

implementace:

- Rel (Java, Berkeley DB), `http://dbappbuilder.sourceforge.net/`

zdroje:


Darwen H.: *An Introduction to Relational Database Theory*
Hugh Darwen and Ventus Publishing ApS, ISBN 978–8776815004


Date C. J., Darwen H.: *Foundation for Object/Relational Databases*
Addison-Wesley Professional 1998, ISBN 978–0201309782


Date C. J., Darwen H.: *Databases, Types and the Relational Model*
Addison Wesley 2006, ISBN 978–0321399427

Jazyk Tutorial D: Skalární typy

zabudované skalární typy:

- `INTEGER` (zkráceně `INT`) – celá čísla
- `RATIONAL` (zkráceně `RAT`) – čísla s pohyblivou řádovou čárkou
- `CHARACTER` (zkráceně `CHAR`) – řetězce (libovolné délky)
- `BOOLEAN` (zkráceně `BOOL`) – pravdivostní hodnoty (`TRUE` a `FALSE`)

definice skalárních typů:

`TYPE` ⟨*jméno-typu*⟩ `POSSREP {`
⟨$komponenta_1$⟩ ⟨typ_1⟩`,`
⋮ ⋮
⟨$komponenta_n$⟩ ⟨typ_n⟩`};`

poznámka:

- výraz (angl.: *expression*) × příkaz (angl.: *statement*) – končí středníkem (**!!**)

Příklad (Tutorial D: Skalární typy a výrazy)

```
10                ⟹ 10
-20.5e2           ⟹ -20.5e2
"Ahoj, svete!"    ⟹ "Ahoj, svete!"
TRUE              ⟹ TRUE
FALSE             ⟹ FALSE

10 = 20           ⟹ FALSE
10 = 20.0         /* error: cannot compare INTEGER and RATIONAL */
TRUE = FALSE      ⟹ FALSE
FALSE = FALSE     ⟹ TRUE

10 = 20 OR 10 = 10      ⟹ TRUE
10 = 10 OR 10 = 10.0    /* error: cannot compare */
OR {10 = 20, 10 = 10}   ⟹ TRUE
OR {}                   ⟹ FALSE
```

Příklad (Tutorial D: Příkazy pro definici skalárních typů)

```
TYPE Name POSSREP {string CHAR};
TYPE Point POSSREP {x RATIONAL, y RATIONAL};
TYPE Score POSSREP {x INTEGER CONSTRAINT x >= 1 AND x <= 5};

TYPE IPv4_Addr POSSREP {
 a INTEGER, b INTEGER, c INTEGER, d INTEGER
 CONSTRAINT a >= 0 AND a <= 255 AND b >= 0 AND b <= 255 AND
            c >= 0 AND c <= 255 AND d >= 0 AND d <= 255};

DROP TYPE IPv4_Addr;

TYPE IPv4_Addr POSSREP {
 a INTEGER, b INTEGER, c INTEGER, d INTEGER
 CONSTRAINT AND {a >= 0, b >= 0, c >= 0, d >= 0,
                 a <= 255, b <= 255, c <= 255, d <= 255}};
```

Příklad (Tutorial D: Práce s hodnotami nových skalárních typů)

```
Name ("Blangis")                 ⟹ Name ("Blangis")
Point (10.5, 20.3)               ⟹ Point (10.5, 20.3)
Score (5)                        ⟹ Score (5)
IPv4_Addr (158, 194, 80, 20)     ⟹ IPv4_Addr (158, 194, 80, 20)

THE_a (IPv4_Addr (158, 194, 80, 20)) ⟹ 158
THE_x (Point (10.5, 20.3))           ⟹ 10.5
THE_string (Name ("Blangis"))        ⟹ "Blangis"

Score (3) = Score (5) ⟹ FALSE
Score (3) = Score (3) ⟹ TRUE

VAR myadr IPv4_Addr;                       /* not persistent */
myadr := IPv4_Addr (158, 194, 80, 20); /* assignment */
WRITELN (THE_a (myadr));                   /* print value */
```

Jazyk Tutorial D: N-ticové typy a výrazy

n-ticový typ:

TUPLE {
$\langle atribut_1 \rangle$ $\langle typ_1 \rangle$,
$\vdots$ $\vdots$
$\langle atribut_n \rangle$ $\langle typ_n \rangle$}

n-ticový výraz:

TUPLE {
$\langle atribut_1 \rangle$ $\langle výraz_1 \rangle$,
$\vdots$ $\vdots$
$\langle atribut_n \rangle$ $\langle výraz_n \rangle$}

poznámka:

- n-ticový typ se používá např. při deklaraci n-ticové proměnné

Příklad (Tutorial D: N-ticové typy a výrazy)

```
TUPLE {}                          ⟼ TUPLE {}
TUPLE {x 10, y 20, z 30}          ⟼ TUPLE {x 10, y 20, z 30}
TUPLE {name "Abbe", score 1}      ⟼ TUPLE {name "Abbe", score 1}

TUPLE {name Name ("Abbe"), score Score (1)}
  ⟼ TUPLE {name Name ("Abbe"), score Score (1)}

x FROM TUPLE {x 10, y 20, z 30} ⟼ 10
y FROM TUPLE {x 10, y 20, z 30} ⟼ 20

name FROM TUPLE {name Name ("Abbe"), score Score (1)}
  ⟼ Name ("Abbe")
score FROM TUPLE {name Name ("Abbe"), score Score (1)}
  ⟼ Score (1)

THE_string (name FROM TUPLE {name Name ("Abbe"), score Score (1)})
  ⟼ "Abbe", score Score (1)}
```

Příklad (Tutorial D: N-ticové typy a výrazy)

```
/* notice: attribute "x" is of tuple type */
VAR t1 TUPLE {x TUPLE {x INTEGER}, y INTEGER, z INTEGER};

WRITELN (t1); /* attributes have implicit values */

t1 := TUPLE {z 100, y 50, x TUPLE {x 666}};

WRITELN (t1);

VAR t2 TUPLE {x INTEGER, y INTEGER, z INTEGER}
  INIT (TUPLE {x 1, y 2, z 3});

WRITELN (t2);
WRITELN (x FROM t2);
```

poznámka:

- skalární ani n-ticové hodnoty *nejsou perzistentní*

Jazyk Tutorial D: Relační typy a výrazy

relační typ:

RELATION {
$\langle atribut_1 \rangle$ $\langle typ_1 \rangle$,
$\vdots$ $\vdots$
$\langle atribut_n \rangle$ $\langle typ_n \rangle$}

relační výraz:

RELATION {$\langle n\text{-}tice_1 \rangle, \ldots, \langle n\text{-}tice_k \rangle$}

RELATION $\langle záhlaví \rangle$ {$\langle n\text{-}tice_1 \rangle, \ldots, \langle n\text{-}tice_k \rangle$}

výrazy pro relace nad prázdným schématem:

TABLE_DUM (zkráceně DUM) $\equiv$ RELATION {} {}

TABLE_DEE (zkráceně DEE) $\equiv$ RELATION {} {TUPLE {}}

Příklad (Tutorial D: Základní relační výrazy)

```
RELATION {
  TUPLE {x 1, y 20},
  TUPLE {x 6, y 30},
  TUPLE {x 5, y 50}} ⟹ ···

RELATION {x BOOLEAN, y INTEGER} {
  TUPLE {x TRUE, y 20},
  TUPLE {x FALSE, y 30},
  TUPLE {y 50, x TRUE}} ⟹ ···

RELATION {
  TUPLE {x 1, y TUPLE {k 2}},
  TUPLE {x 2, y TUPLE {k 3}}} ⟹ ···
```

poznámka:

- `RELATION {}` není legální výraz: není možné určit relační schéma (**!!**)

Příklad (Tutorial D: Relace jako hodnoty atributů)

```
TYPE Name POSSREP {name CHAR};
TYPE Year POSSREP {year INT};
TYPE Subj POSSREP {dept CHAR, code CHAR};
TYPE Perc POSSREP {x INT CONSTRAINT x >= 0 AND x <= 100};
TYPE Date POSSREP {day INT, month INT, year INT};

RELATION {
  TUPLE {
    name Name ("Abbe"),
    year Year (2011),
    subject Subj ("KMI", "DATA1"),
    score RELATION {
      TUPLE {val Perc (34), date Date (18, 12, 11)},
      TUPLE {val Perc (84), date Date (25, 01, 12)}}},
  TUPLE {···}, ...}
```

NAME	YEAR	SUBJ	SCORE	
			VAL	DATE
Abbe	2011	KMI/DATA1	34%	18/12/11
			84%	25/01/12
⋮	⋮	⋮	⋮	

Jazyk Tutorial D: Základní relační proměnné, přiřazení

vytvoření a zrušení (perzistentní) relační proměnné:

`VAR` ⟨*jméno*⟩ `BASE` ⟨*relační-typ*⟩ ⟨*inicializace*⟩ ⟨*klíče*⟩;

`DROP VAR` ⟨*jméno*⟩;

přitom:

- ⟨*relační-typ*⟩ lze vynechat (pokud je přítomna ⟨*inicializace*⟩)
- ⟨*inicializace*⟩ je ve tvaru `INIT` (⟨*relační-výraz*⟩) a lze ji vynechat
- ⟨*klíče*⟩ je posloupnost výrazů `KEY` {⟨$atribut_1$⟩, ⟨$atribut_2$⟩, . . . } oddělených „␣“

relační přiřazení:

⟨*relační-proměnná*⟩ := ⟨*relační-výraz*⟩;

poznámka: přiřazení *nemodifikuje hodnotu* (je konstantní), ale *proměnnou* (**!!**)

Příklad (Tutorial D: Relační proměnné a přiřazení)

```
VAR result BASE
  RELATION {
    name Name, year Year, subject Subj,
    score RELATION {val Perc, date Date}}
  KEY {name, year, subject};

result := RELATION {···};

VAR foo BASE INIT (RELATION {TUPLE {x 10, y 20, z TRUE}})
  KEY {x} KEY {y, z};

VAR bar BASE RELATION {x INT, y INT, z BOOL} KEY {x} KEY {y, z};

bar := RELATION {
         TUPLE {x 10, y 20, z TRUE},
         TUPLE {x 30, y 20, z FALSE}};
```

Příklad (Tutorial D: Porovnávání hodnot a množinová inkluze)

```
10 <= 20          ⟹ TRUE
10 <= 20          ⟹ FALSE
"foo" <= "bar"    ⟹ FALSE
"foo" >= "bar"    ⟹ TRUE

DUM <= DEE ⟹ TRUE
DUM >= DEE ⟹ FALSE

RELATION {x INT, y INT} {} <=
  RELATION {TUPLE {x 10, y 20}} ⟹ TRUE

RELATION {TUPLE {x 1, y 2}} <=
  RELATION {TUPLE {x 10, y 20}} ⟹ FALSE

RELATION {TUPLE {x 1, y 2}} <=
  RELATION {TUPLE {x 1, y 2}, TUPLE {x 10, y 20}} ⟹ TRUE
```

Příklad (Tutorial D: Nové ordinální typy)

```
TYPE Foo ORDINAL POSSREP {x INT, y INT};

OPERATOR LESSTHAN (a Foo, b Foo) RETURNS BOOL;
  RETURN THE_x (a) < THE_x (b) OR
           (THE_x (a) = THE_x (b) AND THE_y (a) < THE_y (b));
END OPERATOR;
```

```
Foo (10, 20) < Foo (10, 21)  ⟹ TRUE
Foo (10, 20) < Foo (10, 15)  ⟹ TRUE
Foo (10, 20) < Foo (11, 20)  ⟹ TRUE
Foo (10, 20) <= Foo (10, 20) ⟹ TRUE
Foo (10, 20) <= Foo (13, 15) ⟹ TRUE

Foo (10, 20) > Foo (10, 21)  ⟹ FALSE
Foo (10, 20) > Foo (10, 15)  ⟹ FALSE
Foo (10, 20) >= Foo (13, 15) ⟹ FALSE
```

Jazyk SQL

základní fakta:

- Structured Query Language (SQL), 1974 (D. Chamberlin, R. Boyce)
- deklarativní jazyk s procedurálními prvky, syntaxe „blízko přirozenému jazyku“

významné rysy SQL:

- široká škála skalárních typů: numerické, řetězcové, pole, kompozitní, . . .
  (ale n-tice nemá řádný protějšek)
- protějšky relačních proměnných a relací: tabulky a jejich obsah
- klíče: jedne dezignovaný *primární klíč* (nemusí být přítomen)
- s relacemi (tabulkami) nelze pracovat jako s elementy prvního řádu

významné rysy SQL:

- ⊕ podpora ve většině relačních SŘBD, standardizace (ANSI/ISO)
- ⊖ nekoncepčnost, plně nepodporuje RM (nebo jde za jeho hranice)

Příklad (SQL: Definice domén a tabulek)

```
CREATE DOMAIN PersonName AS VARCHAR (64)
  DEFAULT 'J. Doe' NOT NULL CHECK (length (VALUE) > 0);

CREATE TABLE person (
  id NUMERIC NOT NULL PRIMARY KEY,
  name PersonName,
  license_no VARCHAR (12) NOT NULL UNIQUE,
  gender VARCHAR NOT NULL CHECK (gender = 'M' OR gender = 'W'),
  bio VARCHAR NOT NULL DEFAULT '');

CREATE TABLE position (
  pid NUMERIC NOT NULL,
  year SMALLINT NOT NULL DEFAULT 2013,
  salary NUMERIC NOT NULL DEFAULT 0,
  PRIMARY KEY (pid, year));
```

Příklad (SQL: Modifikace dat)

```
/* modification by adding new tuples */
INSERT INTO person VALUES (6, 'Abbe', '23A', 'M', 'Small guy.');
INSERT INTO person (id, license_no, gender) VALUES (7, '45N', 'M');

/* violation of integrity constraint */
INSERT INTO person VALUES (8, 'Curval', '23A', 'M', 'XXX');

/* modification by updating */
UPDATE person SET id = 66 WHERE name = 'Abbe';
UPDATE person SET gender = 'W', id = id * 10 WHERE id <= 10;

/* the following is legal in SQL (notice the coercion) */
UPDATE person SET id = id / '10';

/* modification by deleting tuples */
DELETE FROM person WHERE id = 66;
DELETE FROM person;
```

Přednáška 2: Závěr

pojmy k zapamatování:

- relační model dat, atributy, relační schémata, typy (domény)
- typy v RM: skalární, n-ticové, relační
- kartézské součiny, relace, první normální forma
- relační proměnné, klíče a jejich realizace v jazycích Tutorial D a SQL

použité zdroje:


Date C. J.: *Database in Depth: Relational Theory for Practitioners*
O'Reilly Media 2005, ISBN 978–0596100124


Date C. J., Darwen H.: *Databases, Types and the Relational Model*
Addison Wesley 2006, ISBN 978–0321399427


Garcia-Molina H., Ullman J., Widom J.: *Database Systems: The Complete Book*
Prentice Hall 2008, ISBN 978–0131873254