Databázové systémy

Relační Model

Petr Krajča

Katedra informatiky
Univerzita Palackého v Olomouci

Organizační informace

- email: petr.krajca@upol.cz
- konzultační hodiny
  - čtvrtek 13:00 – 14:30,
  - pátek 13:15 – 14:45 (termíny dle rozvrhu)
- www: `http://phoenix.inf.upol.cz/~krajca/`
- slidy budou k dispozici online

Literatura

- Hronek J. *Databázové systémy. 2007*
- Date C.J. *Relational theory for computer professionals.* O'Reilly, 2013.
- Date C.J. *An introduction to database systems.* Pearson Addison Wesley, 2004.
- Date C.J. *SQL and Relational Theory: How to Write Accurate SQL Code.* O'Reilly, 2011.
- Garcia-Molina, H. Ullman J.D., Widom J. *Database Systems: The Complete Book*, Prentice Hall, 2008.

Plán přednášek

1. Základní pojmy, modely dat. Relační databázové systémy: relační model dat, základní pojmy (atributy, domény, ntice, relační schémata, relace). SQL: domény, tabulky.

2. Operace relační algebry: množinové operace, projekce, selekce, dělení, spojení a jeho typy. Vzájemné vztahy relačních operací.SQL: SELECT, pohledy.

3. Fyzická organizace dat, zpracování dotazu. Systém řízení báze dat: služby, architektura, transakční zpracování dat, uzamykací protokoly.SQL: indexy, explain.

4. Konceptuální modelování, návrh relační databáze, normalizace. Referenční integrita v relačním modelu dat. Funkční závislosti: definice, pravdivost v datech, modely, sémantické vyplývání.

5. Spolupráce SQL s jinými jazyky (Java, C#, PHP). Procedury a triggery. Prostředky pro administraci relačního databázového systému.

Co je databázový systém

databáze

- *kolekce perzistentních dat* používaných aplikacemi nějakého subjektu
- *perzistentní* – data přetrvají proces, který je vytvořil
- např. subjekt = výrobní podnik, aplikace = skladové hospodářství

databázový systém

- systém pro organizaci, definici, manipulaci a dotazování nad perzistentními daty, který lze popsat jako množinu algoritmů pracujících s daty v určeném tvaru
- často chápán dvojím způsobem:
  1. jako teorie, tj. **formální model** (přesně definovaný, a který lze zkoumat)
  2. jako konkrétní **softwarová implementace** vycházející z teorie, viz bod 1

Co je model dat?

(formální) model dat

- *Množina abstraktních a soběstačných formálních definic datových struktur a operací s daty (případně dalších operací, omezení, apod.), které dohromady tvoří formální výpočetní model, se kterým mohou uživatelé interagovat.*
- existuje několik různých formálních modelů
- je potřeba rozlišovat formální model a jeho implementaci

model dat (určitého subjektu)

- přesněji: **model databáze**
- návrh nebo implementace organizace dat určitého subjektu
- např. návrh organizace dat "skladové hospodářství" subjektu "firma"

Kategorie jazyků související s formálními modely

- **dotazovací jazyk** (query language, QL) – jazyk pro vyjadřování dotazů pro získávání dat z databáze
- **jazyk pro definici dat** (data definition language, DDL) – jazyk pro popis typu a struktury dat, která budou v databázi uložena
- **jazyk pro modifikaci dat** (data modification language, DML) – jazyk pro vkládání, aktualizaci a mazání dat v databázi

Poznámky:

- jazyky používají (typicky) uživatelé různých rolí:
  - administrátor databáze – jazyk pro definici dat
  - uživatel – dotazovací jazyk, jazyk pro modifikaci dat
- k jednomu formálnímu modelu může existovat více jazyků dané kategorie (např. pro relační model existuje SQL, QUEL, Tutorial D, . . .)

Systém řízení báze dat (SŘBD)

Systém řízení báze dat (Database Management system, DBMS) = programový celek implementující databázový systém vycházející z určitého formálního modelu dat a poskytující následující služby.

- souběžný **víceuživatelský přístup** k databázi (neblokované zpracování dotazů)
- **transakční zpracování dat** (atomicita, konzistence, izolace, trvalost)
- **perzistentní uložení dat** a **zotavení z chyb**
- **integritní omezení** (prevence nekonzistentních dat)
- **bezpečnost přístupu k datům** (autorizace)
- ...

Poznámky:

- složitost implementace SŘBD na úrovni OS
- menší SŘBD neposkytují všechny uvedené fce.
- SŘBD typicky funguje buď jako (síťová) služba nebo jako knihovna (embedded)

Přehled paradigmat

- **souborový model** (1955) – nejstarší, data uložena jako množina záznamů stejného typu v souboru
- **síťový model** (1969) – grafový pohled na schéma databáze (dnes Neo4j)
- **hierarchický model** (1960) – zjednodušení síťového modelu, grafy jsou nahrazeny stromy (dnes renesance v podobě „XML databází")
- **relační model** (1969) – dnes mainstream, založen na pojmu n-ární relace, s úzkou vazbou na logiku
- **objektové modely** (1989) – mnoho modelů, perzistentní uložení objektů, obvykle omezené možnosti dotazovaní
- **relačně/objektové modely** (1990) – více návrhů, pokusy o rozšíření rel. modelu o objekty
- **další** – key-value databáze, dokumentové databáze (semistrukturovaná data), XML databáze, ...
- **NoSQL** databáze = buzzword pro nerelační databáze

Relační model dat

- *E.F.Codd. A Relational Model of Data for Large Shared Data Banks. 1969.*

Charakteristika

- jeden typ databázových objektů **relace**
  - formální protějšek pojmu ,,datová tabulka"
  - formalizuje **základní data**, **výsledky dotazů**, **vztahy mezi daty**
- dobrý teoretický model, který lze efektivně implementovat
- manipulativní formalismy
- dotazovací formalismy
- odpočátku formalizuje související fenomény (závislosti v datech, normalizace)
- integritní omezení
- logická nezávislost

Implementace relačního modelu

- RM implementuje velká řada SŘBD
- žádný (produkčně použitelný) SŘBD neimplementuje čistý RM

Komerční řešení

- Oracle (Oracle), SQL Server (Microsoft), DB2 (IBM), Informix (IBM), ...
- robustní software, ověřený, s podporou, nevýhodou je cena a uzavřenost systémů

Otevřená řešení

- PostgreSQL: vyzrálý databázový systém, velké množství funkcí, široké možnosti programování
- MariaDB: fork MySQL (vícerůzných způsobů uložení dat)
- SQLite: embedded databáze (nepotřebuje server)

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

Pět komponent

1. kolekce **skalárních typů** zahrnující typ pravdivostní hodnota (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

- kolekce skalárních typů a relačních operací jsou ,,otevřené" (ne pevně dané)
- relace = hodnoty; relační proměnné = jména (na něž jsou navázány relační hodnoty)
- definicie dat (1, 2, 3), modifikace (4, 5), dotazování (3, 5)

Relační model neformálně

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

Datová tabulka:

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

name	id	age
Alice	123	20
Bob	456	30
Chuck	789	15

Poznámky:

- ne každá tabulka splňující předchozí popis je ,,uspokojivou strukturou" v RM
- cílem je používat tabulky, které jsou formalizované jako n-ární relace

Požadavky na tabulky

A table is in first normal form (1NF)—equivalently, such a table is normalized—if and only if it's a direct and faithful representation of some relation. — C.J.Date

První normální forma (1NF)

Tabulka je v první normální formě, pokud splňuje všechny následující podmínky:

1. neuvažuje se **žádné uspořádání řádků** (shora-dolů)
2. neuvažuje se **žádné uspořádání sloupců** (zleva-doprava)
3. v tabulce se nevyskytují **duplicitní řádky**
4. v každém vnitřní poli tabulky je **právě jedna hodnota daného typu**
5. všechny atributy jsou **regulární** (neobsahují žádnou skrytou informaci, která je dostupná pouze prostřednictvím speciálních funkcí)

Alternativní (chybné) chápání 1.NF

V literatuře bývá uváděna (pseudo)definice:
Tabulka je 1NF, pokud jsou všechny její hodnoty atomické.

Problémy

- Co je to ,,atomická hodnota"?
- problém s řetězci, multimédii, ...
- nepokrývá patologické případy (duplicitní řádky, absence hodnot)
- postranní úmysly definice (např. zamezení "vnořených tabulek")

Námi používaná definice:
Darwen, Hugh. "Relation-Valued Attributes; or, Will the Real First Normal Form Please Stand Up?", in C. J. Date and Hugh Darwen, Relational Database Writings 1989-1991 (Addison-Wesley, 1992).

Základní pojmy RM: Atribut

Atribut = symbolické jméno. Množinu všech atributů, o které předpokládáme, že je konečená a spočetná, označujeme Y.

Poznámky:

- (intuitivně) atribut je jméno, které může být použito pro označení sloupce datové tabulky
- *syntaktický pojem* (nemá hodnotu)
- značení: $y, y_1, y_2, \ldots$ (obecné označení v rámci RM), `foo`, `id`, `name`, ... (konkrétní atributy)

Základní pojmy RM: Typ (doména)

Typ = pojmenovaná (nejvýše spočetná) množina elementů (hodnot).

Poznámky:

- RM je silně typový, každý atribut (relace, atd.) má přiřazen svůj typ.
- ekvivalentní název pro *typ* je *doména*
- někdy uvažován rozdíl
  - **doména** – množina konkrétních hodnot (sémantický pojem)
  - **typ** – symbolické jméno pro doménu (syntaktický pojem)
- **rozlišitelnost hodnot** – hodnoty stejného typu lze porovnat $=$. Hodnoty různých typů nelze porovnat!

Základní pojmy RM: Relační schéma

Relační schéma je 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.

Poznámky:

- formalizace pojmu záhlaví datové tabulky
- relační schéma představuje *typ* relace (*relační typ*)
- značení: $R, R_1, R_2, \ldots, S_1, S_2, \ldots$
- relační schéma může být prázdné

Dohoda o značení:
Pokud typ vyplývá jednoznačně z kontextu, pak

- relační schémá ztotožňujeme s konečnými podmnožinami $R \subseteq Y$
- doménu (typ) atributu $y \in Y$ značíme D_y

Opakování pojmů

- **množina** – chápeme intuitivně (budeme uvažovat nejvýše spočetné)
- **uspořádaná dvojice** – $\langle a, b \rangle$
- **(naivní) kartézský součin** – $A \times B$ je množina všech uspořádaných dvojic takových, že $A \times B = \{\langle a, b \rangle \mid a \in A, b \in B\}$
- **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 značíme $f(a) = b$

Poznámka:

- fakt, že $f \subseteq A \times B$ je zobrazení píšeme $f : A \to B$

Obecný kartézský součin

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).

Poznámky:

- pro $f \in \prod_{i \in I} A_i$ se $f(i) \in A_i$ nazývá hodnota i v n-tici f
- pokud je $I = \{1, \ldots, n\}$, pak lze psát $\prod_{i=1}^{n} A_i$
- indexy z I nemají žádné pořadí

Kartézský součin (příklad)

$I = \{1, 2, 3\}$, $A_1 = \{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$, $A_2 = \{5, 6, 7, 8\}$, $A_3 = \{\alpha, \beta\}$

$$\begin{aligned}\prod_{i \in I} A_i = \{&\{\langle 1, \mathsf{a}\rangle, \langle 2, 5\rangle, \langle 3, \alpha\rangle\}, \{\langle 1, \mathsf{b}\rangle, \langle 2, 5\rangle, \langle 3, \alpha\rangle\}, \{\langle 1, \mathsf{c}\rangle, \langle 2, 5\rangle, \langle 3, \alpha\rangle\},\\ &\{\langle 1, \mathsf{a}\rangle, \langle 2, 5\rangle, \langle 3, \beta\rangle\}, \{\langle 1, \mathsf{b}\rangle, \langle 2, 5\rangle, \langle 3, \beta\rangle\}, \{\langle 1, \mathsf{c}\rangle, \langle 2, 5\rangle, \langle 3, \beta\rangle\},\\ &\{\langle 1, \mathsf{a}\rangle, \langle 2, 6\rangle, \langle 3, \alpha\rangle\}, \{\langle 1, \mathsf{b}\rangle, \langle 2, 6\rangle, \langle 3, \alpha\rangle\}, \{\langle 1, \mathsf{c}\rangle, \langle 2, 6\rangle, \langle 3, \alpha\rangle\},\\ &\{\langle 1, \mathsf{a}\rangle, \langle 2, 6\rangle, \langle 3, \beta\rangle\}, \{\langle 1, \mathsf{b}\rangle, \langle 2, 6\rangle, \langle 3, \beta\rangle\}, \{\langle 1, \mathsf{c}\rangle, \langle 2, 6\rangle, \langle 3, \beta\rangle\},\\ &\{\langle 1, \mathsf{a}\rangle, \langle 2, 7\rangle, \langle 3, \alpha\rangle\}, \{\langle 1, \mathsf{b}\rangle, \langle 2, 7\rangle, \langle 3, \alpha\rangle\}, \{\langle 1, \mathsf{c}\rangle, \langle 2, 7\rangle, \langle 3, \alpha\rangle\},\\ &\{\langle 1, \mathsf{a}\rangle, \langle 2, 7\rangle, \langle 3, \beta\rangle\}, \{\langle 1, \mathsf{b}\rangle, \langle 2, 7\rangle, \langle 3, \beta\rangle\}, \{\langle 1, \mathsf{c}\rangle, \langle 2, 7\rangle, \langle 3, \beta\rangle\},\\ &\{\langle 1, \mathsf{a}\rangle, \langle 2, 8\rangle, \langle 3, \alpha\rangle\}, \{\langle 1, \mathsf{b}\rangle, \langle 2, 8\rangle, \langle 3, \alpha\rangle\}, \{\langle 1, \mathsf{c}\rangle, \langle 2, 8\rangle, \langle 3, \alpha\rangle\},\\ &\{\langle 1, \mathsf{a}\rangle, \langle 2, 8\rangle, \langle 3, \beta\rangle\}, \{\langle 1, \mathsf{b}\rangle, \langle 2, 8\rangle, \langle 3, \beta\rangle\}, \{\langle 1, \mathsf{c}\rangle, \langle 2, 8\rangle, \langle 3, \beta\rangle\}\}\end{aligned}$$

Každá množina $\{\langle 1, \ldots\rangle, \langle 2, \ldots\rangle, \langle 3, \ldots\rangle\}$ reprezentuje jedno zobrazení

$$f : \{1, 2, 3\} \to \{\mathsf{a}, \mathsf{b}, \mathsf{c}, 5, 6, 7, 8, \alpha, \beta\}$$

takové, že $f(1) \in \{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$, $f(2) \in \{5, 6, 7, 8\}$, $f(3) \in \{\alpha, \beta\}$.

Základní pojmy RM: Relace nad relačním schématem

Mějme relační schéma $R \subseteq Y$ a nechť D_y $(y \in Y)$ označuje 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** $\mathcal{D}$.

Poznámky:

- značení: $\mathcal{D}, \mathcal{D}_1, \ldots$
- $\prod_{y \in R} D_y$ – kartézský součin domén (indexy jsou atributy)
- $r \in \prod_{y \in R} D_y$ je n-tice, $r(y)$ je prvek z D_y
- n-tice reprezentuje "řádek"v tabulce odpovídající $\mathcal{D}$
- tabulka je v 1NF p.k. reprezentuje relaci na relačním schématu
- nulární relace (stupeň 0), unární relace (stupeň 1), binární relace (stupeň 2), ...

Základní pojmy RM: Relace nad relačním schématem (příklad)

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

Příklad ternární relace $A \times B \times C$

$$\{\langle 3, \texttt{single}, \texttt{"Alice"}\rangle, \langle 2, \texttt{married}, \texttt{"Bob"}\rangle,$$
$$\langle 3, \texttt{married}, \texttt{"Chuck"}\rangle, \langle 4, \texttt{divorced}, \texttt{"David"}\rangle\} \subseteq A \times B \times C$$

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

$t_1(\texttt{CHILDREN}) = 3$	$t_1(\texttt{STATUS}) = \texttt{single}$	$t_1(\texttt{NAME}) = \texttt{"Alice"}$
$t_2(\texttt{CHILDREN}) = 2$	$t_2(\texttt{STATUS}) = \texttt{married}$	$t_2(\texttt{NAME}) = \texttt{"Bob"}$
$t_3(\texttt{CHILDREN}) = 3$	$t_3(\texttt{STATUS}) = \texttt{married}$	$t_3(\texttt{NAME}) = \texttt{"Chuck"}$
$t_4(\texttt{CHILDREN}) = 4$	$t_4(\texttt{STATUS}) = \texttt{divorced}$	$t_4(\texttt{NAME}) = \texttt{"David"}$

Vizualizace relace

Relace z předchozího příkladu jako tabulka:

CHILDREN	NAME	STATUS
3	Alice	single
2	Bob	married
3	Chuck	married
4	David	divorced

NAME	CHILDREN	STATUS
Chuck	3	married
Alice	3	single
David	4	divorced
Bob	2	married

Množinová reprezentace

$$\begin{aligned}\mathcal{D} = \{&\{\langle \texttt{CHILDREN}, 3\rangle, \langle \texttt{STATUS}, \texttt{single}\rangle, \langle \texttt{NAME}, \texttt{"Alice"}\rangle\},\\ &\{\langle \texttt{CHILDREN}, 2\rangle, \langle \texttt{STATUS}, \texttt{married}\rangle, \langle \texttt{NAME}, \texttt{"Bob"}\rangle\},\\ &\{\langle \texttt{CHILDREN}, 3\rangle, \langle \texttt{STATUS}, \texttt{married}\rangle, \langle \texttt{NAME}, \texttt{"Chuck"}\rangle\},\\ &\{\langle \texttt{CHILDREN}, 4\rangle, \langle \texttt{STATUS}, \texttt{divorced}\rangle, \langle \texttt{NAME}, \texttt{"David"}\rangle\}\}\end{aligned}$$

Relační proměnná

Relační proměnná daného typu je (perzistentní) proměnná, která může nabývat hodnot, jimiž jsou relace daného typu. (Formalizace jména tabulky v RM.)

Rozlišujeme

1. **základní** relační proměnné
   - mají definovaný (relační) typ
   - množinu klíčů
2. **pohledy**
   - hodnoty vznikají vyhodnocením výrazů v daném čase
   - jejich typ je určen typem relačního výrazu

Klíč základní relační proměnné a relační přiřazení

Uvažujme relační proměnnou X typu $R = \{y_1, \dots y_n\}$.

Množina klíčů proměnné X je libovolná neprázdná množina $\{K_1, \dots, K_n\}$, jejíž prvky jsou podmnožiny R a splňují podmínku, že $K_i \not\subseteq K_j$ pro každé $i \neq j$.

Relační přiřazení

Mějme relační proměnnou X typu R a nechť $\{K_1, \dots, K_n\}$ je množina klíčů proměnné X.

Pak relaci $\mathcal{D}$ typu R lze **přiřadit jakoho hodnotu** proměnné X, pokud je splněna podmínka, že pro každé $i = 1, \dots n$ 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 říkame, že $\mathcal{D}$ porušuje integritní omezení dané některým klíčem X.

Jazyk SQL

- Structured Query Language (SQL), 1974.
- deklarativní jazyk s procedurálními prvky, syntaxe ,,blízká běžné angličtině"

Významné rysy SQL

- široká škála skalárních typů (numerické, řetězcové, pole, kompozitní, . . .)
- protějšky relačních proměnných a relací (tabulky a jejich obsah)
- klíče: primarní klíč (jeden zvolený), nemusí být přítomen
- relace nejsou elementy prvního řádu
- odchyluje se od RM (nepodporuje jej, jde za jeho hranice)

Implementace SQL

- podpora ve vetšině SŘBD
- standardizován (ANSI/ISO)
- jednotlivé implementace se přesto navzájem liší

SQL: Datové typy (1/2)

- `SMALLINT` – 2 bytové celé číslo se znaménkem (-32768 až +32767)
- `INTEGER` nebo `int` – 4 bytové celé číslo se znaménkem (-2147483648 až +2147483647)
- `BIGINT` – 8 bytové celé číslo se znaménkem
- `NUMERIC(pocet-cifer,pocet-desetin-mist)` – slouží k práci s čísly s přesně definovanou přesností.
  - `pocet-cifer` – celkový počet cifer v daném čísle (před i za desetinnou čárkou)
  - `pocet-desetin-mist` – udává počet cifer za desetinnou čárkou
  - např. `NUMERIC(6,2)` umožňuje přesně pracovat s čísly ve tvaru 1234.56
- `REAL`, `DOUBLE PRECISION` – typy odpovídající číslům s plovoucí řádovou čárkou podle normy IEEE 754, tj. typům float a double z jazyka C.

Pozor

- U datových typů dle IEEE 754 není zaručena přesnost a nejsou proto vhodné pro ukládání dat, kde na přesnosti záleží, typicky u finančních operací!!!

SQL: Datové typy (2/2)

- `VARCHAR(n)` – řetezec proměnlivé velikosti mající maximálně n znaků
- `CHAR(n)` – řetezec pevné velikosti mající právě n znaků (chybějící znaky jsou doplněny mezerami)
- `BOOLEAN` – pravdivostní hodnoty (`t`, `f`)
- `DATE`, `TIME`, `TIMESTAMP` – datové typy pro reprezentaci data, času a data + času
- ...

Definice vlastního typu/domény

```
CREATE DOMAIN Salary     /* nazev typu */
   AS NUMERIC(10, 2)     /* puvodni typ */
   DEFAULT 0             /* implicitni hodnota */
   NOT NULL              /* nepovoli hodnoty NULL */
   CHECK (VALUE > 0);    /* podminka pro hodnoty */
```

SQL: Vytvoření tabulky (1/2)

```
CREATE TABLE tabulka
(sloupec1 typ-dat /* vlastnosti-sloupce */,
 sloupec2 typ-dat /* vlastnosti-sloupce */,
 ...,
 sloupecN typ-dat /* vlastnosti-sloupce */,
 /* omezeni-pro-vice-sloupcu */);
```

- `typ-dat` – viz předchozí slidy
- `vlastnosti-sloupce`:
  - `NOT NULL` – daný sloupec nesmí obsahovat hodnotu NULL (doporučuje se NULL vyhnout)
  - `UNIQUE` – hodnoty v daném sloupci musí být unikátní (tj. nesmí se opakovat)
  - `PRIMARY KEY` – určuje, že daný sloupec je primární klíč, tj. umožňuje jednoznačně identifikovat daný řádek (odpovídá `UNIQUE NOT NULL`)
  - `DEFAULT hodnota` – definuje implicitní hodnotu pro daný sloupec, pokud není uvedeno, bere se jako implicitní hodnota NULL
  - `CHECK podminka` – určuje podmínku, kterou musí hodnota v daném sloupci splňovat

SQL: Vytvoření tabulky (2/2)

- `omezeni-pro-vice-sloupcu`
  - `UNIQUE (sloupec1, ..., sloupecN)` – udává, že hodnoty v daných sloupcích musí být unikátní (tj. nesmí se opakovat)
  - `PRIMARY KEY (sloupec2, ..., sloupecN)` – dané sloupce tvoří primární klíč, tj. umožňují jednoznačně identifikovat daný řádek
  - `CHECK podminka` – umožňuje deklarovat podmínku, platící pro více sloupců

SQL: Vytvoření tabulky (příklady)

```
CREATE TABLE zamestnanci
(jmeno  VARCHAR(30) PRIMARY KEY,
 vek    INT NOT NULL CHECK (vek > 0),
 plat   Salary,
 uvazek NUMERIC(3, 2) NOT NULL
        DEFAULT 1
        CHECK ((uvazek > 0) AND (uvazek < 2)),
 funkce VARCHAR(20));

CREATE TABLE domy
(ulice          VARCHAR(40),
 cislo_popisne  INT CHECK (cislo_popisne > 0),
 mesto          VARCHAR(20),
 pocet_obyvatel INT,
 PRIMARY KEY (ulice, cislo_popisne, mesto));
```

SQL: Práce s daty – vkládání

```
INSERT INTO tabulka (sloupec1, sloupec2, ..., sloupecN)
   VALUES (hodnota1, hodnota2, ..., hodnotaN);
```

- seznam sloupců lze vynechat, v takovém případě se předpokládá, že budou zadány hodnoty pro všechny sloupce v pořadí, jak byly sloupce zadány příkazem `CREATE TABLE`
- v seznamu sloupců je možné některé sloupce vynechat, v takovém případě se pro daný sloupec použije ,,defaultní" hodnota, byla-li zadána, nebo hodnota NULL
- je možné vložit několik řádků pomocí jednoho příkazu `INSERT`, části `VALUES` jsou jednotlivé řádky odděleny čárkami

SQL: Práce s daty – vkládání (příklady)

```
/* uplny priklad */
INSERT INTO zamestnanci (jmeno, vek, plat, uvazek, funkce)
   VALUES ('Tomas Pech', 67, 50000, 1.2, 'Manager');

/* vice radku na jednou */
INSERT INTO zamestnanci (jmeno, vek, plat, uvazek, funkce)
  VALUES ('Jan Novak', 40, 16789.20, 0.8, 'Manager'),
         ('Petr Klic', 37, 23100, 0.5, 'Udrzbar');

/* pro sloupec uvazek se pouzije implicitni hodnota */
INSERT INTO zamestnanci (jmeno, vek, funkce, plat)
   VALUES ('Tomas Marny', 20, 'Kreativec', 15000);

/* chybi vycet sloupcu */
INSERT INTO zamestnanci VALUES ('Ronald McDonald', 30, 100000, 1.0, 'CEO')
```

SQL: Práce s daty – aktualizace

```
UPDATE tabulka SET sloupec1 = vyraz1 [, sloupec2 = vyraz2, ... ];
```

- aktualizuje všechny řádky tabulky tak, že `sloupec1` bude mít hodnotu `vyrazu1`, `sloupec2` bude roven `vyrazu2`, ...

```
UPDATE tabulka SET sloupec1 = vyraz1 [, sloupec2 = vyraz2, ... ]
   WHERE podminka;
```

- chová se stejně jako předchozí tvar, jen aktualizuje řádky splňující danou podmínku

Příklady

```
/* nastavi generalnimu rediteli plat 200 tis. */
UPDATE zamestnanci SET plat = 200000 WHERE  funkce = 'CEO';

/* zvysi vsem zamestnancum plat o 10% */
UPDATE zamestnanci SET plat = plat * 1.10;

/*vsem zamestnancum nad 50 let a jsou ve funkci Manager, zdvojnasobi plat
UPDATE zamestnanci SET plat = plat * 2, funkce = 'Senior Manager'
   WHERE (vek > 50) AND (funkce = 'Manager');
```

SQL: Práce s daty – odstranění řádků

```
DELETE FROM tabulka;
```

- odstraní z tabulky všechny řádky

```
DELETE FROM tabulka WHERE podminka;
```

- odstraní řádky splňující danou podmínku

Příklad

```
DELETE FROM zamestnanci WHERE vek > 50;
```

SQL: Práce s daty – získání úplného obsahu tabulky

```
SELECT * FROM tabulka;
```

Příklad

```
SELECT * FROM zamestnanci;
```

```
      jmeno       | vek |   plat    | uvazek |  funkce
------------------+-----+-----------+--------+-----------
 Tomas Pech       |  67 |  50000.00 |   1.20 | Manager
 Jan Novak        |  40 |  16789.20 |   0.80 | Manager
 Petr Klic        |  37 |  23100.00 |   0.50 | Udrzbar
 Tomas Marny      |  20 |  15000.00 |   1.00 | Kreativec
 Ronald McDonald  |  30 | 100000.00 |   1.00 | CEO
(5 rows)
```

pokračování příště...