Jemný úvod do SQL

Jaroslav Janda

Září 1997

verze 1

Obsah

1 Předmluva

Tato „příručka“ vznikla jako jednoduchý návod pro základní vstup do SQL (Structured Query Language – Strukturovaný dotazovací jazyk). Při kompletaci této „příručky“ byla použita, pro nastudování problému, kniha „Učíme se SQL“ od Jana Pokorného (vydavatelství Plus). Pokud tedy chcete mít stručný začátek na mnohem více stránkách a za peníze, nebo chcete mít barevnou obálku, pak si můžete tuto knihu koupit. Pokud by sem někdo chtěl něco doplnit, nechť mi pošle o svém záměru zprávu na `djanda@hotmail.com` nebo krátkou zprávu na `jaroslav.janda@sms.paegas.cz`.

2 Úvod

Jak už bylo uvedeno výše (pro ty, co nečtou předmluvy ještě jednou), SQL znamená Structured Query Language (Strukturovaný dotazovací jazyk) a slouží pro tvorbu univerzálních dotazů v databázích. Dále také umožňuje zakládání tabulek (definování dat), ošetření přístupu k datům, sdílení dat nebo třeba zabezpečení databází. SQL je výhodné používat v kombinaci s některým výkonějším programovacím jazykem nebo databází. Struktura příkazů SQL je velmi názorná, a většinou na první pohled se dá zjistit, co bude příkaz provádět. Další text vyžaduje alespoň základní znalosti o databázích a programování. Pokud to pochopí i někdo, kdo takové znalosti nemá, budu velmi rád.

Pro příklady jsou použity následující tabulky:

Struktura tabulky: TABULKA.DBF

Field	Field Name	Type	Width	Dec	Index
1	PRIJMENI	Character	20		No
2	JMENO	Character	15		No
3	TITUL	Character	10		No
4	RODNEC	Character	10		Desc
5	DIVIZE	Character	2		No
6	NASTUP	Date	8		No
7	PLAT	Numeric	12	2	No

PRIJMENI	JMENO	TITUL	RODNEC	DIVIZE	NASTUP	PLAT
Janda	Jaroslav		7707231111	2	01/07/96	4000.00
Kriz	Jiri	Ing.	7302202222	2	01/01/95	8000.00
Adamec	Milos		7508011111	3	01/05/96	6500.00
Hanus	Jan	RNDr.	7012202222	1	01/07/96	9000.00
Zikmund	Adam		7401201111	2	01/07/96	7000.00
Strasky	Lubos		7109092222	3	01/06/93	7500.00

Struktura tabulky: DIVIZE.DBF

Field	Field Name	Type	Width	Dec	Index
1	DIVIZE	Character	2		No
2	POPIS	Character	20		No

DIVIZE	POPIS
1	Reditel
2	Analytici
3	Programatori

Použité typy písma a jejich význam:

- **SELECT** – klíčová slova ve struktuře příkazu a příkladech

- *tabulka* – parametry ve struktuře příkazu
- PRIJMENI – příklady

Další typy písma jsou použity omylem.

Nejlepší přístup k této „příručce“ je, že si ji vezmeme na klín, sedneme k počítači a vše budeme zkoušet a experimentovat. Použité DBF soubory jsou standardně přiloženy k této „příručce“. Příklady byly odzkoušeny v M$ Visual FoxPro, a protože se (bohužel) jednotlivé interprety SQL liší (ač existuje norma ANSI/ISO, ale tak už to ve světě chodí), je možné že některé prostě nebudou fungovat.

3 Základní dotazy

Chceme-li pracovat z tabulkou a provádět nad ni některé operace, použijeme příkaz SELECT.

Následující pomocné příkazy vykonává příkaz SELECT v uvedeném pořadí (žádný z uvedených nemusí být použit, podrobnější popis a příklady použití jsou uvedeny níže):

1. FROM (výběr zdrojové tabulky)
2. WHERE (filtrovací podmínka)
3. DISTINCT (odstranění duplicitních řádek)
4. ORDER BY (setřídění)

Zde je uvedena struktura příkazu SELECT, převzata z nápovědy ve Foxce:

```
SELECT [ALL | DISTINCT] [TOP nExpr [PERCENT]]
        [Alias.] Atribut [AS Jméno_sloupce]
        [, [Alias.] Atribut [AS Jméno_sloupce] ...]
FROM [FORCE] [Jméno_databáze!]Tabulka [Lokální_alias]
        [[INNER | LEFT [OUTER] | RIGHT [OUTER] | FULL [OUTER] JOIN
                [Jméno_databáze!]Tabulka [Lokální_alias]
                [ON Připojovací_podmínka ...]
[[INTO Cílová_tabulka]
        | [TO FILE Jméno_souboru [ADDITIVE] | TO PRINTER [PROMPT]
        | TO SCREEN]]
[PREFERENCE PreferenceName]
[NOCONSOLE]
[PLAIN]
[NOWAIT]
[WHERE Výběrová_podmínka [AND | OR Výběrová_podmínka2 ...]]
[GROUP BY Skupinový_sloupec [, Skupinový_sloupec ...]]
[HAVING Filtrovací_podmínka]
[UNION [ALL] SELECTCommand]
[ORDER BY Třídící_položka [ASC | DESC] [, Třídící_položka [ASC | DESC] ...]]
```

3.1 Vypsání všech sloupců tabulky: FROM

SELECT * FROM Tabulka

Příkaz provede vypsání všech sloupců tabulky.

```
PRIJMENI  JMENO     TITUL  RODNEC      DIVIZE  NASTUP       PLAT
Janda     Jaroslav         7707231111  2       01/07/96   4000.00
```

Kriz	Jiri	Ing.	7302202222	2	01/01/95	8000.00
Adamec	Milos		7508011111	3	01/05/96	6500.00
Hanus	Jan	RNDr.	7012202222	1	01/07/96	9000.00
Zikmund	Adam		7401201111	2	01/07/96	7000.00
Strasky	Lubos		7109092222	3	01/06/93	7500.00

Symbol ***** v SQL znamená symbol pro všechny sloupce.

3.2 Vypsání vyjmenovaných sloupců

SELECT `Prijmeni, Jmeno, Titul, RodneC` **FROM** `Tabulka`
Příkaz provede vypsání všech vyjmenovaných sloupců.

PRIJMENI	JMENO	TITUL	RODNEC
Janda	Jaroslav		7707231111
Kriz	Jiri	Ing.	7302202222
Adamec	Milos		7508011111
Hanus	Jan	RNDr.	7012202222
Zikmund	Adam		7401201111
Strasky	Lubos		7109092222

3.3 Dynamický sloupec

SELECT `Prijmeni, Jmeno, Titul, Plat, (Plat/30)` **FROM** `Tabulka`
Příkaz nám vydefinuje sloupec, který se spočte z některého z ostatních sloupců. Tento sloupec se pak přidá k vypisované tabulce.

PRIJMENI	JMENO	TITUL	PLAT	EXP_5
Janda	Jaroslav		4000.00	133.33
Kriz	Jiri	Ing.	8000.00	266.67
Adamec	Milos		6500.00	216.67
Hanus	Jan	RNDr.	9000.00	300.00
Zikmund	Adam		7000.00	233.33
Strasky	Lubos		7500.00	250.00

V tomto příkladě se přidá plat v dolarech, nebo-li plat dělený 30. Nový sloupec tabulky je pojmenován automaticky dle konvencí použité databáze. K definici vypočteného sloupce lze použít i volání některého složitějšího programu, který daný sloupec vypočte.

3.3.1 Pojmenování nového sloupce: `AS`

SELECT `Prijmeni, Jmeno, Titul, Plat, (Plat/30)` **AS** `'Dolary'` **FROM** `Tabulka`
Pokud máme potřebu si nově založený sloupec pojmenovat, uděláme to pomocí výše uvedeného postupu.

PRIJMENI	JMENO	TITUL	PLAT	DOLARY
Janda	Jaroslav		4000.00	133.33
Kriz	Jiri	Ing.	8000.00	266.67
Adamec	Milos		6500.00	216.67
Hanus	Jan	RNDr.	9000.00	300.00
Zikmund	Adam		7000.00	233.33
Strasky	Lubos		7500.00	250.00

Příkaz provede pojmenování nového sloupce jako „Dolary“. Popis sloupce musí odpovídat konvencím použitého programu.

3.4 Použití konstantního sloupce

SELECT `'Příjmení:'` **AS** `'Prijmeni', Prijmeni` **AS** `'HODNOTA', 'Jméno:'` `AS 'Jmeno', Jmeno` **AS** `'HODNOTA2'` **FROM** `Tabulka`
Uvedený postup umožňuje založení konstantního sloupce, nebo-li sloupec který obsahuje konstantní hodnotou, například nápis.

PRIJMENI	HODNOTA	JMENO	HODNOTA2
Příjmení:	Janda	Jméno:	Jaroslav
Příjmení:	Kriz	Jméno:	Jiri
Příjmení:	Adamec	Jméno:	Milos
Příjmení:	Hanus	Jméno:	Jan
Příjmení:	Zikmund	Jméno:	Adam
Příjmení:	Strasky	Jméno:	Lubos

3.5 Výběr řádků: `WHERE`

SELECT * **FROM** `Tabulka` **WHERE** `Jmeno='Ja'`
Pro výběr řádků, které odpovídají určitým podmínkám (podmínce), použijeme uvedený postup.

PRIJMENI	JMENO	TITUL	RODNEC	DIVIZE	NASTUP	PLAT
Janda	Jaroslav		7707231111	2	01/07/96	4000.00
Hanus	Jan	RNDr.	7012202222	1	01/07/96	9000.00

Při tomto výběru by se zobrazili i jména 'Jaromír', 'Jaroslava' atd. Pro výběr pouze jména 'Jaro' se musí použít znaménko přesně rovno „==“. Pokud chceme zkombinovat více podmínek a vytvořit složitější dotaz, použijeme operátoru `AND`, `OR` nebo `NOT`. Operátor `NOT` se vztahuje na bezprostředně následující příkaz. Přehlednější a jistější formy dosáhneme použitím závorek „(“ a „)“. Jako srovnávací operátory můžeme použít:

- = rovno
- == přesně rovno
- <>, ! = nerovno (může se lišit podle použité aplikace)
- < menší než, <= menší nebo rovno
- > větší než, >= větší nebo rovno

Pro výběr podle datumu použijeme:
SELECT * **FROM** `Tabulka` **WHERE** `Nastup <{01.06.96}`
Zobrazí všechny pracovníky z tabulky, kteří nstoupili před „01.06.1996“.

PRIJMENI	JMENO	TITUL	RODNEC	DIVIZE	NASTUP	PLAT
Kriz	Jiri	Ing.	7302202222	2	01/01/95	8000.00
Adamec	Milos		7508011111	3	01/05/96	6500.00
Strasky	Lubos		7109092222	3	01/06/93	7500.00

3.5.1 Zobrazení hodnot v určitém rozpětí: `BETWEEN`

SELECT * **FROM** `Tabulka` **WHERE** `Plat` **BETWEEN** `1000` **AND** `7000`
`BETWEEN` slouží k zobrazení hodnot sloupce, které se nacházejí v určitém rozpětí.

PRIJMENI	JMENO	TITUL	RODNEC	DIVIZE	NASTUP	PLAT
Janda	Jaroslav		7707231111	2	01/07/96	4000.00
Adamec	Milos		7508011111	3	01/05/96	6500.00
Zikmund	Adam		7401201111	2	01/07/96	7000.00

Výsledkem je seznam pracovníků, kteří mají plat v rozmezí 1000 a 7000. To samé bychom mohli samozřejmě zapsat bez `BETWEEN`:
SELECT * FROM `Tabulka` **WHERE** `Plat>=1000` **AND** `Plat<=7000`

3.5.2 Otestování některého znaku: `LIKE`

SELECT * FROM `Tabulka` **WHERE** `Jmeno` **LIKE** `"____s%"`
Uvedený postup slouží k otestování existence některého znaku na libovolném místě řetězce.

PRIJMENI	JMENO	TITUL	RODNEC	DIVIZE	NASTUP	PLAT
Janda	Jaroslav		7707231111	2	01/07/96	4000.00
Adamec	Milos		7508011111	3	01/05/96	6500.00
Strasky	Lubos		7109092222	3	01/06/93	7500.00

V tomto případě uvedený postup vyfiltruje všechny, kteří mají na pátém místě ve jméně písmenko „s“. Znaky z filtrů, používané při dotazu na jména souborů ? a *, jsou nahrazeny znaky _ (podtržítko) a % (procento).
Toho samého efektu můžeme dosáhnout pomocí:
SELECT * FROM `Tabulka` **WHERE SUBS**`(Jmeno,5,1)='s'`

3.5.3 Výpis řádek s prázdným sloupcem: `EMPTY()`

SELECT * FROM `Tabulka` **WHERE EMPTY**`(Titul)`
Slouží pro vypsání všech řádků tabulky, které nemají uvedenou prázdnou položku.

PRIJMENI	JMENO	TITUL	RODNEC	DIVIZE	NASTUP	PLAT
Janda	Jaroslav		7707231111	2	01/07/96	4000.00
Adamec	Milos		7508011111	3	01/05/96	6500.00
Zikmund	Adam		7401201111	2	01/07/96	7000.00
Strasky	Lubos		7109092222	3	01/06/93	7500.00

Příkaz vypíše všechny pracovníky, kteří mají nějaký titul. Pokud naopak chceme řádky, v nichž položka uvedená je:
SELECT * FROM `Tabulka` **WHERE NOT EMPTY**`(Titul)`:

PRIJMENI	JMENO	TITUL	RODNEC	DIVIZE	NASTUP	PLAT
Kriz	Jiri	Ing.	7302202222	2	01/01/95	8000.00
Hanus	Jan	RNDr.	7012202222	1	01/07/96	9000.00

3.6 Výpis výsledku na tiskárnu: `TO PRINTER`

SELECT * FROM `Tabulka` **TO PRINTER**
Provede vypsání výběru nejen na obrazovku, ale i na tiskárnu. Pokud bychom nechtěli výpis na obrazovku, použijeme:
SELECT * FROM `Tabulka` **TO PRINTER NOCONSOLE**

3.7 Tříděný výpis: `ORDER BY`

SELECT * FROM `Tabulka` **ORDER BY** `Jmeno`
Pro výpis tabulky, setříděné podle některého sloupce slouží klauzule `ORDER BY`.

PRIJMENI	JMENO	TITUL	RODNEC	DIVIZE	NASTUP	PLAT
Zikmund	Adam		7401201111	2	01/07/96	7000.00
Hanus	Jan	RNDr.	7012202222	1	01/07/96	9000.00
Janda	Jaroslav		7707231111	2	01/07/96	4000.00

Kriz	Jiri	Ing.	7302202222	2	01/01/95	8000.00
Strasky	Lubos		7109092222	3	01/06/93	7500.00
Adamec	Milos		7508011111	3	01/05/96	6500.00

Tabulka bude setříděna podle jména a to vzestupně. Pro sestupné třídění se použije klauzule `DESCENDING` (zkráceně lze použít `DESC`), pro implicitní vzestupné třídění `ASCENDING` (zkráceně `ASC`).
SELECT * FROM `Tabulka` **ORDER BY** `Jmeno` **DESCENDING**

PRIJMENI	JMENO	TITUL	RODNEC	DIVIZE	NASTUP	PLAT
Adamec	Milos		7508011111	3	01/05/96	6500.00
Strasky	Lubos		7109092222	3	01/06/93	7500.00
Kriz	Jiri	Ing.	7302202222	2	01/01/95	8000.00
Janda	Jaroslav		7707231111	2	01/07/96	4000.00
Hanus	Jan	RNDr.	7012202222	1	01/07/96	9000.00
Zikmund	Adam		7401201111	2	01/07/96	7000.00

Výsledkem je setříděná tabulka sestupně podle jména.

Pokud chceme třídit tabulku podle sloupce, jehož jméno například neznáme, nebo ho nechceme uvádět, použijeme:
SELECT * FROM `Tabulka` **ORDER BY** `2` **DESCENDING**
Získáme ten samý výsledek, jako v předchozím příkladu.

3.8 Třídění podle více atributů

SELECT * FROM `Tabulka` **ORDER BY** `Prijmeni, Jmeno, Titul`
Třídit lze i podle více kritérií.

PRIJMENI	JMENO	TITUL	RODNEC	DIVIZE	NASTUP	PLAT
Adamec	Milos		7508011111	3	01/05/96	6500.00
Hanus	Jan	RNDr.	7012202222	1	01/07/96	9000.00
Janda	Jaroslav		7707231111	2	01/07/96	4000.00
Kriz	Jiri	Ing.	7302202222	2	01/01/95	8000.00
Strasky	Lubos		7109092222	3	01/06/93	7500.00
Zikmund	Adam		7401201111	2	01/07/96	7000.00

Tabulka je současné setříděna nejprve podle příjmení a pak podle jména.

3.9 Výstup do textového souboru: `TO FILE`

SELECT * FROM `Tabulka` **TO FILE** `Soubor`
Provede výpis výsledku do textového souboru „Soubor“. Všechny příklady uvedené v této „příručce“, byly vytvořeny tímto postupem. Pokud chceme výstup k textovému souboru připojit:
SELECT * FROM `Tabulka` **TO FILE** `Soubor` **ADDITIVE**

3.10 Výstup do tabulky: `INTO TABLE`

SELECT * FROM `Tabulka` **INTO TABLE** `Novatabulka`
Provede výstup výsledku do tabulky „Novatabulka“. Pro zapsání do dočasné tabulky, tzv. dočasného kurzoru:
SELECT * FROM `Tabulka` **INTO CURSOR**
Pro zapsání do pole použijeme:
SELECT * FROM `Tabulka` **INTO ARRAY**
Matice se automaticky založí podle velikosti výstupu.

3.11 Zamezení vypisování duplicitních řádek: DISTINCT

SELECT DISTINCT * FROM `Tabulka`
Uvedenou klauzuli použijeme, pokud nechceme vypisovat duplicitní řádky (aby byl řádek duplicitní, musí být duplicitní ve všech vypisovaných sloupcích).

4 Agregační dotazy

Tyto typy dotazů zpracovávají hodnoty z celých sloupců tabulky, v SQL nalezneme tyto funkce:

- `SUM()` – součet numerických hodnot ve sloupci
- `MIN()` – minimální hodnota ve sloupci
- `MAX()` – maximální hodnota ve sloupci
- `COUNT()` – počet numerických hodnot ve sloupci
- `AVG()` – aritmetický průměr numerických hodnot ve sloupci

Vnořování uvedených funkcí do sebe, např. `MIN(SUM())`, není ve většině implementací SQL povoleno.

U agregačního dotazu zpracovává SQL klauzule v následujícím pořadí (některé uvedené klauzule nemusí příkaz obsahovat):

1. `FROM`
2. `WHERE`
3. `GROUP BY`
4. `HAVING`
5. výpočet hodnot řádku
6. `DISTINCT`
7. `ORDER BY`

4.1 Součet hodnot: SUM()

SELECT SUM`(Plat)` **AS** `'Celkem'` **FROM** `Tabulka`
Uvedený postup použijeme pokud chceme provést součet sloupce tabulky. Podobně postupujeme i při aplikace ostatních klauzulí.

```
  CELKEM
42000.00
```

Zobrazen je součet všech platů.

4.2 Seskupení hodnot: GROUP BY

SELECT `Divize`, **SUM**`(Plat)` **AS** `'Celkem'` **FROM** `Tabulka` **GROUP BY** `Divize`
Chceme-li provést seskupení součtů k některému sloupci, použijeme klauzuli `GROUP BY`.

```
DIVIZE    CELKEM
1        9000.00
2       19000.00
3       14000.00
```

Zobrazen je součet platů v jednotlivých divizích.

4.3 Kritéria pro zařazení: `HAVING`

SELECT `Divize,` **SUM**`(Plat)` **AS** `'Celkem'` **FROM** `Tabulka` **GROUP BY** `Divize` **HAVING SUM**`(Plat)>10000`
Klauzule omezuje rozsah tabulky tím, že z agregačních řádků vyřadí ty, které neodpovídají uvedené podmínce.

DIVIZE	CELKEM
2	19000.00
3	14000.00

Výsledkem je součet platů v jednotlivých divizích, kdy plat přesahuje hodnotu 3500.

5 Dotazy na více tabulek

5.1 Výpis svázaných informací z druhé tabulky

SELECT `Prijmeni, Jmeno, Titul, Popis` **FROM** `Tabulka, Divize` **WHERE** `Tabulka.Divize=Divize.Divize`
Provede vypsání názvů divizí k jednotlivým pracovníkům podle vazby přes společný atribut 'Divize'.

PRIJMENI	JMENO	TITUL	POPIS
Janda	Jaroslav		Analytici
Kriz	Jiri	Ing.	Analytici
Adamec	Milos		Programatori
Hanus	Jan	RNDr.	Reditel
Zikmund	Adam		Analytici
Strasky	Lubos		Programatori

Podmínka `WHERE` je nutná, jinak bychom získali kombinaci všech prvků z obou tabulek (přes prvek 'Divize'), bez této podmínky by navíc příkaz postrádat smysl.
Pro vypsání všech atributů z tabulky 'Divize' můžeme použít:
SELECT `Prijmeni, Jmeno, Titul, Divize.*` **FROM** `Tabulka, Divize` **WHERE** `Tabulka.Divize=Divize.Divize`

PRIJMENI	JMENO	TITUL	DIVIZE	POPIS
Janda	Jaroslav		2	Analytici
Kriz	Jiri	Ing.	2	Analytici
Adamec	Milos		3	Programatori
Hanus	Jan	RNDr.	1	Reditel
Zikmund	Adam		2	Analytici
Strasky	Lubos		3	Programatori

5.2 Sloučení navzájem nekonzistentních tabulek: `UNION`

Tímto příkazem se tato „příručka“ nezabývá, neboť je v běžné praxi nepotřebný. Možná v další verzi.

5.3 Samosloučení

SELECT `Tabulka.Prijmeni, Tabulka2.Prijmeni` **FROM** `Tabulka, Tabulka Tabulka2` **WHERE** `Tabulka.Prijmeni<Tabulka2.Prijmeni`
Výpis seznamu pro zápas ve stolním tenise, stylem „každý z každým“. SQL umožňuje otevřít pomocí příkazu `SELECT` dvakrát tu samou tabulku, je pouze nutné druhé tabulce přiřadit jiný lokální alias.

PRIJMENI_A	PRIJMENI_B
Janda	Kriz
Janda	Zikmund
Janda	Strasky
Kriz	Zikmund
Kriz	Strasky
Adamec	Janda
Adamec	Kriz
Adamec	Hanus
Adamec	Zikmund
Adamec	Strasky
Hanus	Janda
Hanus	Kriz
Hanus	Zikmund
Hanus	Strasky
Strasky	Zikmund

6 Složené dotazy

6.1 Poddotaz s porovnávacími operátory

Kromě níže uvedených predikátů pro konstrukci poddotazů, lze použít i porovnávací operátory.

6.2 Poddotaz s `IN`

Pokud upravíme tabulku „Tabulka“, že vyprázdníme „Divizi“ u pracovníka „Janda“, tak aby nám **SELECT * FROM** `Tabulka` vypsal:

PRIJMENI	JMENO	TITUL	RODNEC	DIVIZE	NASTUP	PLAT
Janda	Jaroslav		7707231111		01/07/96	4000.00
Kriz	Jiri	Ing.	7302202222	2	01/01/95	8000.00
Adamec	Milos		7508011111	3	01/05/96	6500.00
Hanus	Jan	RNDr.	7012202222	1	01/07/96	9000.00
Zikmund	Adam		7401201111	2	01/07/96	7000.00
Strasky	Lubos		7109092222	3	01/06/93	7500.00

Nyní potřebujeme získat seznam pracovníků nezařazených do divizí.
SELECT `Prijmeni, Jmeno, Titul` **FROM** `Tabulka` **WHERE** `Tabulka.Divize` **NOT IN** (**SELECT** `Divize` **FROM** `Divize`

PRIJMENI	JMENO	TITUL
Janda	Jaroslav	

S pomocí predikátu `IN` vlastně provádíme kontrolu, zda je v tabulce „Divize“ obsažena hodnota atributu „Divize“ z „Tabulka“.

6.3 Poddotaz s `EXIST`

Pokud si vezmeme úplně stejný příklad, jako předchozí s predikátem `IN`, napíšeme dotaz s pomocí `EXIST` takto:
SELECT `Prijmeni, Jmeno, Titul` **FROM** `Tabulka` **WHERE NOT EXIST** (**SELECT** `Divize` **FROM** `Divize` **WHERE** `Divize.Divize=Tabulka.Divize`)
Predikát `EXIST` provádí test existence hodnoty v tabulce.

6.4 Poddotaz s ANY

Příklad vypsání pracovníků, jejichž divize je zanesena do tabulky „Divize“, s použitím predikátu ANY:
SELECT Prijmeni, Jmeno, Titul **FROM** Tabulka **WHERE** Divize = **ANY** (**SELECT** Divize **FROM** Divize)

```
PRIJMENI  JMENO     TITUL
Kriz      Jiri      Ing.
Adamec    Milos
Hanus     Jan       RNDr.
Zikmund   Adam
Strasky   Lubos
```

Predikát ANY provádí test na existenci alespoň jedné hodnoty.

6.5 Poddotaz s ALL

Predikát ALL provádí test na shodu všech hodnot.

7 SQL tabulky

7.1 Vytváření tabulky: CREATE TABLE

CREATE TABLE Divize (Divize c(2), Popis c(20)) Jelikož máme tuto tabulku již vytvořenou, slouží tento příklad pouze jako ukázkový. Pokud bychom tedy chtěli jakoby vytvořit novou, použili bychom výše uvedený příkaz. Vznikne nám pak toto:

```
Struktura tabulky: DIVIZE.DBF
Field  Field Name  Type       Width  Dec  Index
    1  DIVIZE      Character      2       No
    2  POPIS       Character     20       No
```

Příkaz provede založení prázdné tabulky podle uvedených parametrů. Význam parametrů jednotlivých atributů je následující:

- **c***(délka)* – znakový sloupec o délce *délka*
- **n***(míst, des.míst* – číselný sloupec o počtu míst *míst*, z toho je *des.míst* desetinných, pro plovoucí desetinnou čárku se použije typ *f*
- **l** – logická hodnota (délka je automaticky 1)
- **d** – datové pole (délka je automaticky 8)
- **m** – memo pole (délka je automaticky 10)

Struktura příkazu CREATE TABLE je následující (převzato z nápovědy Foxky):

```
CREATE TABLE | DBF Jméno_tabulky [NAME Dlouhé_jméno_tabulky] [FREE]
        (Atribut Typ_atributu [(nŠířka_atributu [, nPrecision])]
               [NULL | NOT NULL]
               [CHECK Výraz [ERROR Zpráva]]
               [DEFAULT Výraz]
               [PRIMARY KEY | UNIQUE]
               [REFERENCES Jméno_tabulky [TAG TagName1]]
               [NOCPTRANS]
```

```
        [, Atribut ...]
                [, PRIMARY KEY Výraz TAG TagName2
                |, UNIQUE Výraz TAG TagName3]
                [, FOREIGN KEY Výraz TAG TagName4 [NODUP]
                        REFERENCES Jméno_tabulky [TAG TagName5]]
                [, CHECK Výraz [ERROR Zpráva]])
| FROM ARRAY Jméno_pole
```

7.2 Úprava struktury tabulky: ALTER TABLE

ALTER TABLE `Divize` **RENAME COLUMN** `Popis` **TO** `Nazev`

```
DIVIZE  NAZEV
1       Reditel
2       Analytici
3       Programatori
```

Provedení úpravy názvu atributu 'Popis' na 'Nazev'. Další možnosti úprav vyčteme z výpisu syntaxe příkazu:

```
ALTER TABLE Jméno_tabulky
        ADD | ALTER [COLUMN] Atribut
                FieldType [(Šířka [, nPrecision])]
                [NULL | NOT NULL]
                [CHECK Výraz [ERROR Zpráva]]
                [DEFAULT Výraz]
                [PRIMARY KEY | UNIQUE]
                [REFERENCES Jméno_tabulky [TAG TagName1]]
                [NOCPTRANS]

ALTER TABLE Jméno_tabulky
        ALTER [COLUMN] Atribut
                [NULL | NOT NULL]
                [SET DEFAULT Výraz]
                [SET CHECK Výraz [ERROR Zpráva]]
                [DROP DEFAULT]
                [DROP CHECK]

ALTER TABLE Jméno_tabulky
        [DROP [COLUMN] Atribut]
        [SET CHECK Výraz [ERROR Zpráva]]
        [DROP CHECK]
        [ADD PRIMARY KEY Výraz TAG TagName2]
        [DROP PRIMARY KEY]
        [ADD UNIQUE Výraz [TAG TagName3]]
        [DROP UNIQUE TAG TagName4]
        [ADD FOREIGN KEY [Výraz] TAG TagName4
                REFERENCES Jméno_tabulky [TAG TagName5]]
        [DROP FOREIGN KEY TAG TagName6 [SAVE]]
        [RENAME COLUMN Atribut TO Atribut]
        [NOVALIDATE]
```

7.3 Přidání řádku: INSERT INTO

INSERT INTO `Divize (Divize, Popis)` **VALUES** `('4','Uklizecka')`

```
DIVIZE  POPIS
1       Reditel
2       Analytici
3       Programatori
4       Uklizecka
```

Při přidávání řádku do tabulky nemusí být uvedeny všechny atributy tabulky, navíc jejich pořadí nemusí odpovídat pořadí, v jakém byly založeny.
Struktura příkazu `INSERT INTO` je následující (převzato z nápovědy Foxky):

```
INSERT INTO Jméno_tabulky [(Atribut1 [, Atribut2, ...])]
        VALUES (Výraz1 [, Výraz2, ...])
```

7.4 Vymazání řádku: DELETE FROM

DELETE FROM `Divize` **WHERE** `Popis='Programatori'`
Provede vymazání záznamu, ve kterém je jméno divize 'Programatori'.

```
DIVIZE  POPIS
1       Reditel
2       Analytici
4       Uklizecka
```

Struktura příkazu `DELETE FROM` je následující (převzato z nápovědy Foxky):

```
DELETE FROM [Jméno_databáze!]Jméno_tabulky
        [WHERE Výběrová_podmínka1 [AND | OR Výběrová_podmínka2 ...]]
```

7.5 Oprava dat: UPDATE

UPDATE `Divize` **SET** `Popis='Reditelka'` **WHERE** `Divize='1'`
Provede záměnu popisu divize na 'Reditelka' u všech záznamů, kde číslo divize je '1'.

```
DIVIZE  POPIS
1       Reditelka
2       Analytici
3       Programatori
4       Uklizecka
```

Struktura příkazu:

```
UPDATE [Jméno_databáze!]Jméno_tabulky
SET Jméno_sloupce = Výraz
        [, Jméno_sloupce = Výraz ...]
        WHERE Výběrová_podmínka [AND | OR Výběrová_podmínka ...]]
```

8 Závěr

Pokud se někdo dočetl až sem, pak mu gratuluji a velmi mne potěší krátký pozdrav od takového člověka. Přeji mnoho úspěchů při používání SQL. Jestli budete SQL používat a přijdete na něco, co by tato příručka měla obsahovat, tak mne prosím informujte.

Sazba systémem TEX s nadstavbou LATEX