

Interpret SQL
Semestrální práce z předmětu 36PJ

Martin Tůma, 4/44

10.1.2006

Obsah

1	Zadání	3
1.1	Jazyk	3
1.2	Interpretace	4
2	Implementace	4
2.1	Lexikální analyzátor	4
2.2	Syntaktický a sémantický analyzátor	6
2.3	Databázový modul	7
3	Instalace a použití interpretu	8
4	Závěr	8

1 Zadání

Úkolem semestrální práce je navrhnout interpret podmnožiny jazyka SQL.

1.1 Jazyk

Jazyk byl zvolen tak, aby obsahoval všechny hlavní příkazy SQL, jmenovitě tedy: CREATE, INSERT, SELECT, UPDATE, DELETE, DROP. Jejich syntaxe je však maximálně zjednodušena.

Zvolená množina jazyka je daná následující EBNF:

sql_příkaz ::=

```
CREATE TABLE jméno_tabulky
  '('jméno_sloupce typ_sloupce ['(velikost_sloupce)']
  '{, jméno_sloupce typ_sloupce ['(velikost_sloupce)']}'')';

| INSERT INTO jméno_tabulky
  ['(jméno_sloupce {, jméno_sloupce}')]
VALUES
  '('výraz {, výraz}');

| UPDATE jméno_tabulky
  SET jméno_sloupce = výraz {, jméno_sloupce = výraz}
  [WHERE podmínka];

| DELETE FROM jméno_tabulky
  [WHERE podmínka];

| SELECT
  * | jméno_sloupce {, jméno_sloupce}
FROM jméno_tabulky
  [WHERE podmínka];

| DROP TABLE jméno_tabulky;
```

výraz ::= řetězec | aritmetický_výraz | NULL

podmínka ::= jméno_sloupce (<> | != | = | < | <= | > | >=) výraz

aritmetický_výraz ::= [-] term {(+|-) term}

term ::= faktor {(*|/) faktor}

faktor ::= celé_číslo | '('aritmetický_výraz)'

```
velikost_sloupce ::= aritmetický_výraz
typ_sloupce ::= INTEGER | CHAR
jméno_sloupce, jméno_tabulky ::= identifikátor
```

Doplňující pravidla:

- Jazyk může obsahovat komentáře. Komentář je posloupnost znaků začínající `--` a končící znakem konce řádky (jedná se tedy o „jednořádkové“ komentáře).
- Jména klíčových slov (`CREATE`, `SELECT`, ...) jsou „case-insensitive“, jména identifikátorů „case-sensitive“.
- Porovnávat lze jen obsah sloupec s výrazem shodného typu a všechny typy sloupců na rovnost a nerovnost s hodnotou `NULL`.
- Při operaci `insert` musí být zadány hodnoty všech sloupců.

1.2 Interpretace

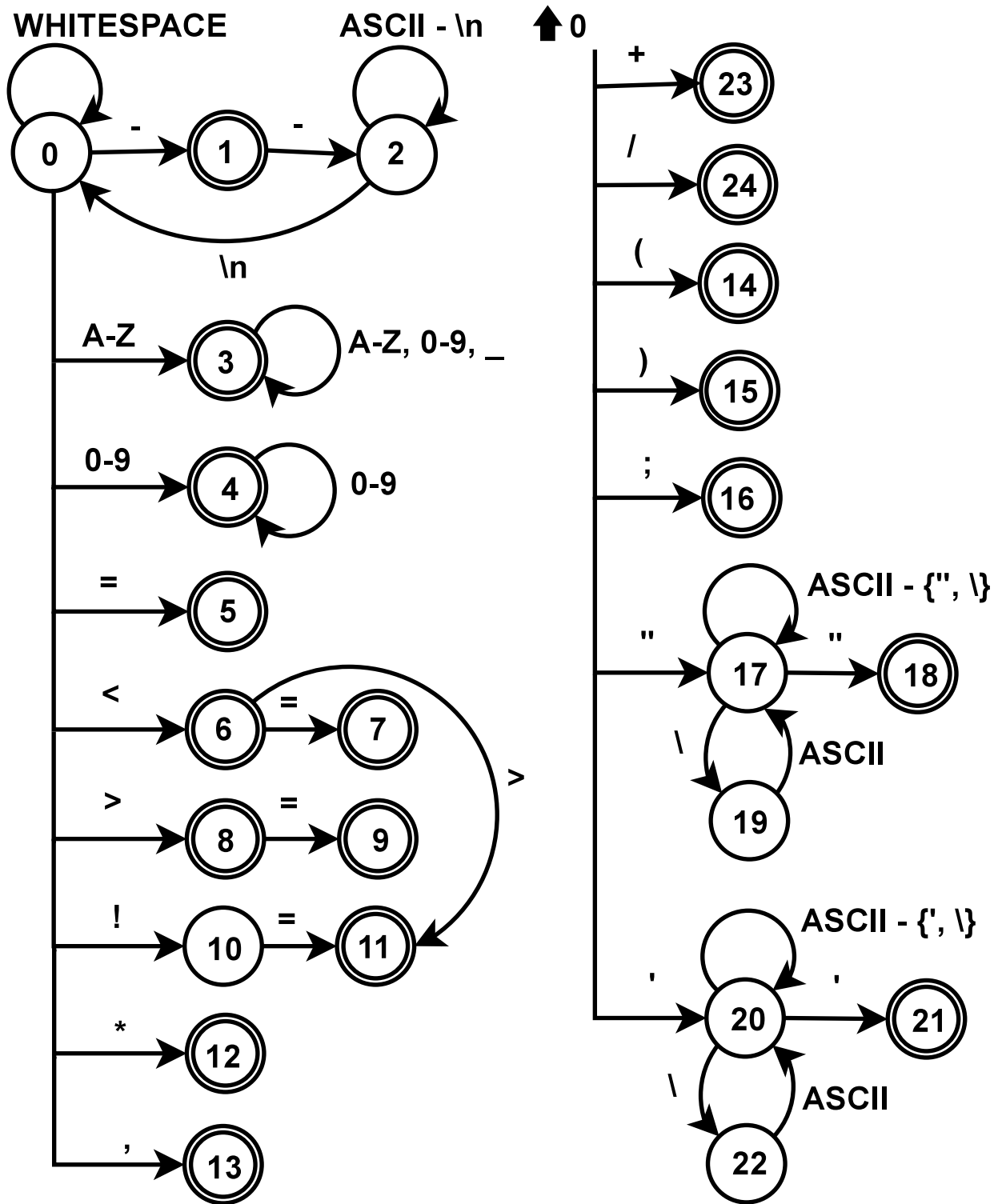
Nad výše popsaným jazykem je implementována syntaktická a sémantická analýza, skutečně interpretované jsou však pouze příkazy `CREATE`, `INSERT`, `SELECT` a `DROP`. Příkazy `UPDATE` a `DELETE` interpretovány (prováděny) nejsou. Je to z toho důvodu, že implementace diskových operací nad daty je poměrně náročná a jelikož překračuje rámec předmětu, je zvolena nejmenší možná množina příkazů, na které je možné „smysluplně“ ověřit funkci interpretu.

2 Implementace

2.1 Lexikální analyzátor

Lexikální analyzátor je implementován klasickou formou konečného automatu realizovaného řídicí strukturou. Lexikální analyzátor rozpoznává tyto lexikální elementy: identifikátor, klíčová slova (součástí LA je i tabulka klíčových slov), řetězec, celé číslo, jednotlivé porovnávací a aritmetické operátory a speciální znaky (závorky, čárka...). Schéma příslušného konečného automatu je na obrázku 1.

Poznámka: Jak jména jednotlivých identifikátorů, tak řetězce mohou obsahovat pouze znaky anglické abecedy.



Obrázek 1: Schéma lexikálního analyzátoru

2.2 Syntaktický a sémantický analyzátor

Syntaktický analyzátor je zkonstruován formou rekurzivního sestupu. Použitou (LL1) gramatiku udává následující odstavec. Tučně jsou zvýrazněny neterminální symboly, italicou výstupní symboly.

$C \rightarrow \text{CREATE TABLE id (CD); CREATE}$
 $C \rightarrow \text{SELECT CL FROM id WD; SELECT}$
 $C \rightarrow \text{INSERT INTO id IC VALUES (VL); INSERT}$
 $C \rightarrow \text{DELETE FROM id WD; DELETE}$
 $C \rightarrow \text{UPDATE id SET UD WD; UPDATE}$
 $C \rightarrow \text{DROP TABLE id; DROP}$
 $C \rightarrow \varepsilon$

$CD \rightarrow \text{id DT DS CD'}$
 $CD' \rightarrow \text{,CD} \mid \varepsilon$

$DS \rightarrow \text{(E)} \mid \varepsilon$
 $DT \rightarrow \text{CHAR} \mid \text{INTEGER}$

$CL \rightarrow \text{*} \mid \text{CN}$
 $CN \rightarrow \text{id CN'}$
 $CN' \rightarrow \text{,CN} \mid \varepsilon$

$IC \rightarrow \text{(CN)} \mid \varepsilon$
 $VL \rightarrow \text{V VL'}$
 $VL' \rightarrow \text{,VL} \mid \varepsilon$
 $V \rightarrow \text{E} \mid \text{řetězec} \mid \text{NULL}$

$UD \rightarrow \text{id = V UD'}$
 $UD' \rightarrow \text{,UD} \mid \varepsilon$

$WD \rightarrow \text{WHERE id CO V} \mid \varepsilon$

$E \rightarrow \text{T E'}$
 $E' \rightarrow \text{+ T E'} \mid \text{- T E'} \mid \varepsilon$
 $T \rightarrow \text{F T'}$
 $T' \rightarrow \text{* F T'} \mid \text{/ F T'} \mid \varepsilon$
 $F \rightarrow \text{int} \mid \text{(E)}$

$CO \rightarrow \text{=} \mid \text{!=} \mid \text{<} \mid \text{<=} \mid \text{>} \mid \text{>=}$

Gramatika je atributovaná. Jednotlivé atributy jsou uvedeny v následující tabulce.

Tabulka atributů:

Symboly	Syntetizované atributy	Dědičné atributy
CL, CN, CN', UD, UD', IC, UPDATE, SELECT	columns, col_num	columns, col_num
UD, UD', VL, VL', V, INSERT, UPDATE,	values	values
VL, VL', V, UPDATE	val_num	val_num
DT	type	
DS	size	
WD, SELECT, UPDATE, DELETE	condition	
E, T, F	int_value	
E', T'	int_value	operand
int	int_value	
řetězec	string_value	
NULL	null_value	
CO	operator	

Jak je vidět, atributy slouží zejména k získání seznamu sloupců/hodnot pro výstupní symboly v pravidlech pro neterminál C (Command). Tam kde to lze je v implementaci neterminálů se stejným syntetizovaným a dědičným parametrem využito možnosti předávání parametrů funkcí odkazem, kdy postačuje jeden atribut pro oba směry „posunu“ informací derivačním stromem. Typickým příkladem je právě získávání seznamů sloupců/hodnot. Směrem „dolů“ jsou do seznamu přidávány nové prvky a upravována pozice pro nový prvek. Směrem „nahoru“ pak „putuje“ vytvářený seznam. Pravidla pro práci s atributy neuvádím, měly by však být zřejmé z implementace.

Kromě výše uvedených atributů, je v syntaktickém analyzátoru ještě globální (a tudíž dostupné ve všech funkcích implementujících jednotlivé neterminály) pole s definicí sloupců, jakási obdoba tabulky symbolů. Oproti tomuto seznamu jsou pak prováděny kontroly existence sloupců, datových typů hodnot a funkce `create_table()` pomocí něj vytváří novou tabulku.

2.3 Databázový modul

Poslední součástí interpretu je modul implementující jednotlivé diskové operace nad „databází“. Obsahuje implementaci pro vytvoření tabulky, získání informací (definic jednotlivých sloupců) o tabulce, vložení řádky do tabulky, výpis řádků tabulky a smazání tabulky.

3 Instalace a použití interpretu

Interpret je napsán v programovacím jazyce C a ke svému běhu potřebuje pouze pár základních knihoven, které by měly být dostupné na každém POSIX kompatibilním systému. Program lze automaticky sestavit pomocí přiloženého `Makefile`.

Použití interpretu je velice jednoduché – je-li programu zadán vstupní parametr, je interpretován soubor se jménem parametru. Je-li program spuštěn bez parametru, je vstup čten ze standartního vstupu.

Funkci programu lze vyskoušet na 2 přiložených souborech. První, `test.sql`, je komplexní test interpretu, druhý (`test2.sql`) slouží pouze k vytvoření a naplnění testovací tabulky, nad kterou lze posléze v „interaktivním“ módu provádět libovolné příkazy.

4 Závěr

Výsledný program úspěšně realizuje zvolenou podmnožinu jazyka SQL. Pokud by však měl být použitelný i jinak, než jako semestrální úloha na 36PJ, bylo by potřeba zcela jinak koncipovat způsob práce s daty. Jako praktické seznámení s problémy implementace interpretu programovacího jazyka a problémů s tím spojených nicméně implementace funguje velice dobře...