

project2: 简易sql parser

Authored by: BryanHe

一、扩展功能

1. 读取外部输入文件

使用文件格式

```
./sql [-h] [fil1.sql file2.sql .... ]
```

- [-h] 输入-h, 输入文件的语句就不会显示在终端
- [fil1.sql file2.sql] 程序会按顺序执行fil1.sql file2.sql中的sql语句

e.g.

=====test/file1.sql=====

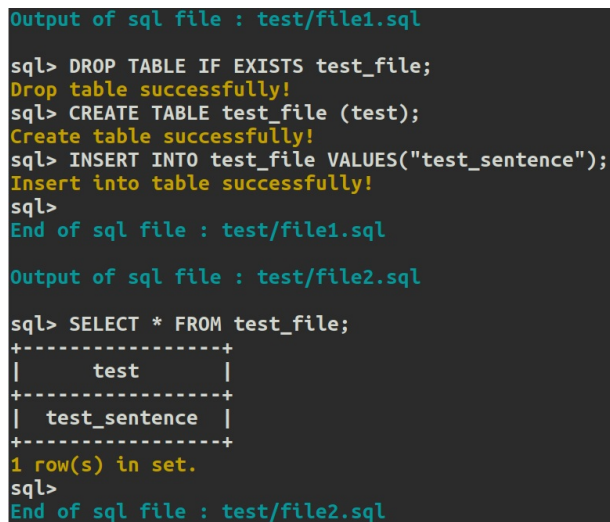
```
DROP TABLE IF EXISTS test_file;  
CREATE TABLE test_file (test);  
INSERT INTO test_file VALUES("test_sentence");
```

=====test/file2.sql=====

```
SELECT * FROM test_file;
```

执行

```
./sql file1.sql file2.sql
```



```
Output of sql file : test/file1.sql  
sql> DROP TABLE IF EXISTS test_file;  
Drop table successfully!  
sql> CREATE TABLE test_file (test);  
Create table successfully!  
sql> INSERT INTO test_file VALUES("test_sentence");  
Insert into table successfully!  
sql>  
End of sql file : test/file1.sql  
  
Output of sql file : test/file2.sql  
sql> SELECT * FROM test_file;  
+-----+  
|      test      |  
+-----+  
| test_sentence |  
+-----+  
1 row(s) in set.  
sql>  
End of sql file : test/file2.sql
```

2. 语句颜色标志(示例如上图)

- 白色语句: 普通语句
- 蓝色语句: 文件读入标志语句
- 黄色语句: 语句成功执行提示语句
- 红色语句: 异常执行语句

3. 支持多行输入

4. 完善的类型系统

当前支持: BOOLEAN, SMAILINT, INTEGER, BIGINT, FLOAT, DOUBLE, VARCHAR(SIZE) 7种数据类型

在执行函数运算时会自动进行类型转换

```
e.g.  
  
./sql test/extend_1.sql
```

5.支持多表联合查询: 理论上支持不限个数表格联合查询

e.g.

```
./sql test/extend_2.sql
```

p_a, p_b的结构都是

col
1
2
3
4

4 row(s) in set.

```
...> SELECT * FROM p_a,p_b;
```

p_a.col	p_b.col
1	1
1	2
1	3
1	4
2	1
2	2
2	3
2	4
3	1
3	2
3	3
3	4
4	1
4	2
4	3
4	4

16 row(s) in set.

6.支持多种双目运算符以及含括号的表达式

理论上支持不限长度的语法正确的计算表达式

当前支持的双目运算符以及其优先级

- OR 2
- AND 2
- < 3
- > 3
- <> 3
- = 3

- <= 3
- >= 3
- + 4
- - 4
- * 5
- / 5
- LIKE 6

其中，LIKE是模糊字符串匹配，支持sql通配符以及正则表达式

e.g.

```
./sql test/extend_4.sql
```

```
...> SELECT * FROM test_a;
+-----+
| a | b | c |
+-----+
| 1 | 7 | TestA1 |
| 2 | -33333 | TeestA2 |
| 3 | 0 | testA |
| 4 | 66 | #test# |
| 5 | 2 | teestA3 |
| 6 | -211 | testAA55 |
+-----+
6 row(s) in set.
```

```
...> SELECT * FROM test_a WHERE c LIKE "[Tt]estA%";
+-----+
| a | b | c |
+-----+
| 1 | 7 | TestA1 |
| 3 | 0 | testA |
| 6 | -211 | testAA55 |
+-----+
3 row(s) in set.
```

7.WHERE后允许接表达式

示例一、

```
./sql test/extend_2.sql
```

p_a, p_b, p_c, p_d的结构都是

```
+-----+
| col |
+-----+
| 1 |
| 2 |
| 3 |
| 4 |
+-----+
4 row(s) in set.
```

获得1,2,3,4全排列

```
SELECT
  p_a.col, p_b.col, p_c.col, p_d.col
FROM
  p_a, p_b, p_c, p_d
WHERE
  p_a.col <> p_b.col AND p_a.col <> p_c.col AND p_a.col <> p_d.col AND
  p_b.col <> p_c.col AND p_b.col <> p_d.col AND
  p_c.col <> p_d.col;
```

p_a.col	p_b.col	p_c.col	p_d.col
1	2	4	3
1	3	2	4
1	3	4	2
1	4	2	3
1	4	3	2
1	2	3	4
2	1	3	4
2	1	4	3
2	3	1	4
2	3	4	1
2	4	1	3
2	4	3	1
3	1	2	4
3	4	2	1
3	4	1	2
3	2	4	1
3	2	1	4
3	1	4	2
4	1	2	3
4	1	3	2
4	2	1	3
4	2	3	1
4	3	1	2
4	3	2	1

24 row(s) in set.

获得[1,4]内和为7的整数解

```

SELECT
    p_a.col,p_b.col,p_c.col,p_d.col
FROM
    p_a,p_b,p_c,p_d
WHERE
    p_a.col+p_b.col+p_c.col+p_d.col=7;

```

p_a.col	p_b.col	p_c.col	p_d.col
1	1	2	3
1	1	3	2
1	1	4	1
1	2	1	3
1	2	2	2
1	2	3	1
1	3	1	2
1	3	2	1
1	4	1	1
1	1	1	4
2	1	1	3
2	3	1	1
2	2	2	1
2	2	1	2
2	1	3	1
2	1	2	2
3	1	1	2
3	1	2	1
3	2	1	1
4	1	1	1

20 row(s) in set.

示例二、

```
./sql test/extend_4.sql
```

8.UPDATE支持多个键值修改

9.UPDATE SET允许接表达式

e.g.

```
./sql test/extend_3.sql
```

```
...> SELECT * FROM student;
+-----+-----+-----+-----+-----+
| id | name | birth_place | major_code | account |
+-----+-----+-----+-----+-----+
| 1 | 卷王 | 火星 | 1 | 100 |
| 2 | Juan Nei | NJU | 2 | 200 |
| 3 | 蒟蒻 | Nowhere | 1 | 300 |
| 4 | aaa | Nowhere | 3 | 87 |
| 5 | bbb | Nowhere | 1 | 44 |
+-----+-----+-----+-----+-----+
5 row(s) in set.
```

为major_code为1的同学account充值1000元

```
sql> UPDATE student SET account=account+1000 WHERE major_code=1;
Update table successfully!
sql> SELECT id,student.name,account FROM student;
+-----+-----+-----+
| id | student.name | account |
+-----+-----+-----+
| 1 | 卷王 | 1100 |
| 2 | Juan Nei | 200 |
| 3 | 蒟蒻 | 1300 |
| 4 | aaa | 87 |
| 5 | bbb | 1044 |
+-----+-----+-----+
5 row(s) in set.
```