存储过程，从新的视角看待数据库应用程序开发（游标，变量，结果集）

> by ColdZoo 2015年03月26日

通常情况下，在操作数据库里数据的时候，我们会选择一种编程语言（Java，C++, et al.），然后利用它们实现我们的业务逻辑。这种方法非常繁琐，并且在表结构发生修改的时候需要重写整个数据库访问模块。 实际上，现代的数据库系统已经为我们准备了另一套更加优雅的解决方案，也就是本文要讲到的——“存储过程”。 关于存储过程的定义，这里有详细的介绍。我们就不再赘述。

本文主要介绍MySQL中存储过程的几个关键而基础的语法（Syntax），涉及游标，结果集，变量，查询语句，DML，DDL。由于水平有限，如有谬误敬请指正

与其他教程不同，笔者决定不从语法开始，而是在工程中碰到问题的时候才讲解语法。因此，如果你需要语法方面的信息，请随时参考[这里]。下面来看看在***实际***的工程中，如何使用存储过程。

需求

我们需要对如下数据表中的数据进行处理：

数据表说明

表名	说明
d_annotation	保存的是标注信息，主键是AnnotaId
c_anp	保存的是ANP(Adj. Noun Pair)中Adj.和Noun的Id，主键Anpid
c_vnp	保存的是VNP(Verb. Noun Pair)中Verb.和Noun的Id，主键Vnpid
b_verb	保存的是Verb的Id和Keyword（英文）
b_adj	保存的是Adj.的Id和Keyword
b_noun	保存的是Noun的Id和Keyword

此外，为了便于处理，我们准备了如下视图：

视图名	说明
useful_data	经过预处理后的可用数据，内有两个字段，AnnotaId 和 FileName，分别代表数据文件名和相应的标注Id。

关键字段

表	字段	含义
d_annotation	AnnotaId	标注信息的Id，每个标注信息均不相同，每个文件Id对应唯一一个AnnotaId
d_annotation	AnpId* (*为1...8)	该标注中第*个Anp的Id

需求

我们的需求是 对视图useful_data中给定的每一行数据进行处理，将每个文件名对应的一个Annotation中的ANP,VNP信息改为Adj，Verb信息。

> 例如，图片0001.jpg原来的标注信息是：VNP1(Cover Face)+VNP2(Smile Man)+ANP1(Green Grassland) 我们要把以上信息转化为：Verb1(Cover)+Verb2(Smile)+Adj1(Green)

技术分析

要实现以上需求，我们需要执行下列步骤

1. 循环遍历视图useful_data取出AnnotaId
2. 根据AnnotaId 查询表 d_annotation, 取出ANPID 和 VNPID (有多个)
3. 对每一个取出的ID，查询表c*anp和表*cvnp，取出Adj.和Verb.的ID,然后在b*verb 和 b*adj中得到动词和形容词的英文形式。综合之后便可以得到每个文件名对应的动词和形容词了。

实现

具体实现如下：

声明变量

以上三步中各个步骤的中间结果均需要保存到变量里，SQL的存储过程中可以用如下语法声明变量。

```
declare variable_name type default default_value
```

其中type为数据类型，常用的有

> | INT | VARCHAR | CHAR |....

我们声明了如下变量

```
DECLARE Done INT DEFAULT 0;
declare annoid int;
declare fn varchar(45);


declare anp1 int;
declare anp2 int;
declare anp3 int;
declare anp4 int;
declare anp5 int;
declare anp6 int;
declare anp7 int;
declare anp8 int;

declare vnp1 int;
declare vnp2 int;
declare vnp3 int;
declare vnp4 int;
declare vnp5 int;
declare vnp6 int;
declare vnp7 int;
declare vnp8 int;

declare adj1 varchar(45);
declare adj1id varchar(45);
declare adj2 varchar(45);
declare adj2id varchar(45);
declare adj3 varchar(45);
declare adj3id varchar(45);
declare adj4 varchar(45);
declare adj4id varchar(45);
declare adj5 varchar(45);
declare adj5id varchar(45);
declare adj6 varchar(45);
declare adj6id varchar(45);
declare adj7 varchar(45);
declare adj7id varchar(45);
```

```
declare adj8 varchar(45);
declare adj8id varchar(45);

declare verb1 varchar(45);
declare verb1id varchar(45);
declare verb2 varchar(45);
declare verb2id varchar(45);
declare verb3 varchar(45);
declare verb3id varchar(45);
declare verb4 varchar(45);
declare verb4id varchar(45);
declare verb5 varchar(45);
declare verb5id varchar(45);
declare verb6 varchar(45);
declare verb6id varchar(45);
declare verb7 varchar(45);
declare verb7id varchar(45);
declare verb8 varchar(45);
declare verb8id varchar(45);
```

利用游标循环遍历数据表

声明了以上变量之后我们可以开始进行第一步，之前接触过存储过程的朋友肯定知道，存储过程中可以使用循环语句，那么我们要用循环语句来遍历数据表么？答案是否定的。

我们使用游标来遍历数据表，游标的语法如下：

```
DECLARE rs CURSOR FOR SELECT AnnotaId,FileName from useful_data;

DECLARE CONTINUE HANDLER FOR SQLSTATE '02000' SET Done = 1;
```

> 第一句用于声明游标，rs为游标的名称，FOR 后面为我们游标作用的语句。 第二句用于设置Handler,其作用是在游标移动到表的末尾时，设置Done变量为1，也就是退出我们下面这个循环

```
repeat
    ......
    -- some other code;
    ......
    fetch rs into annoid,fn;
until Done end repeat;
```

注意上面那句

> fetch rs into annoid,fn 意思是将游标指向的数据赋值给指定变量。

嵌套查询

遍历之后，我们得到了AnnotaId，我们还需要进行第二步的查询，那么在循环体内又该如何查询新的表呢?

```
select AnpId1,AnpId2,AnpId3,AnpId4,VnpIdA,VnpIdB,VnpIdC,VnpIdD from
d_annotation where AnnotaId=annoid into
anp1,anp2,anp3,anp4,vnp1,vnp2,vnp3,vnp4;
```

使用的select ... into ... 到这里我们有了ANP/VNP 的ID

下面我们用相似的步骤得到剩下变量的值:

```
        select AdjId from c_anp where AnpId = anp1 into adj1id;
        select Keyword from b_adj where SynId=adj1id into adj1;
        select AdjId from c_anp where AnpId = anp2 into adj2id;
        select Keyword from b_adj where SynId=adj2id into adj2;
        select AdjId from c_anp where AnpId = anp3 into adj3id;
        select Keyword from b_adj where SynId=adj3id into adj3;
        select AdjId from c_anp where AnpId = anp4 into adj4id;
        select Keyword from b_adj where SynId=adj4id into adj4;

        select VerbId from c_vnp where VnpId=vnp1 into verb1id;
        select Keyword from b_verb where SynId=verb1id into verb1;
        select VerbId from c_vnp where VnpId=vnp2 into verb2id;
        select Keyword from b_verb where SynId=verb2id into verb2;
        select VerbId from c_vnp where VnpId=vnp3 into verb3id;
        select Keyword from b_verb where SynId=verb3id into verb3;
        select VerbId from c_vnp where VnpId=vnp4 into verb4id;
        select Keyword from b_verb where SynId=verb4id into verb4;
```

在查询中使用DML和DDL

进行到这一步，我们已经得到了动词和名次，我们需要用DML（data manipulate language）语言,插入结果到新的表中

```
insert into
t_word_from_anno(AnnotaId,verb1_id,verb1,verb2_id,verb2,verb3_id,verb3,v
erb4_id,verb4,adj1_id,adj1,adj2_id,adj2,adj3_id,adj3,adj4_id,adj4,FileNa
me)
values(annoid,verb1id,verb1,verb2id,verb2,verb3id,verb3,verb4id,verb4,ad
j1id,adj1,adj2id,adj2,adj3id,adj3,adj4id,adj4,fn);
```

结果展示

```
select * from t_word_from_anno

# id, AnnotaId, verb1_id, verb1, verb2_id, verb2, verb3_id, verb3,
verb4_id, verb4, adj1_id, adj1, adj2_id, adj2, adj3_id, adj3, adj4_id,
adj4, FileName

'16842', '45', 'SID-02148788-V', 'show', NULL, NULL, NULL, NULL, NULL,
NULL, 'SID-01048406-A', 'happy', NULL, NULL, NULL, NULL, NULL, NULL,
'1379559307320.gif'
```

大功告成。

常见错误

Syntax error: missing 'end'

请检查声明语句和其他语句的顺序是否正确，在SQL的存储过程中，声明语句必须放在其他所有语句之前。

```
delete from t_word_from_anno where id >0;
delete from t_verbs where 1>0;
delete from t_adjs where 1>0;

DECLARE Done INT DEFAULT 0;
```

这样写就是错误的

Syntax error: 'from' is not a valid input at this position

这个错误估计是Mysql独有的， 按照SQL的标准，从游标读取数据的语法为

```
fetch next from cursor_name into variable
```

我查阅了mysql 官方的document， 上面写的是

```
FETCH [[NEXT] FROM] cursor_name INTO var_name [, var_name] ...
```

也就是说可以省略Next From， 然而，在MySQL workbench 6.2 （目前的最新版）中会提示上述错误。改为

```
fetch cursor_name into variable
```

后正常。这个是MYSQL WORKBENCH的一个BUG.

遍历表的时候只读出了一条记录（循环提前终止）

按照我们上文的代码，循环终止的条件是Done=1。而 Done变量由一个Handler负责维护，设置的条件是

```
SQLSTATE '02000'
```

简单的说，是在查询出错的时候设置变量。若Repeat循环查询过程没出错，则可以正常终止循环。否则，就会因为Repeat内部新查询的出错而提前终止。详细信息可以参考这个官方说明。

解决方法很简单，既然可能在内部出错（我们的例子中某些位置上ANP,VNP可能是空的），我们就在Repeat结束之前手动设置一下。为了保证循环正常终止，把Fetch移动到最后进行。

```
set Done=0;
fetch rs into annoid,fn;
until Done end repeat;
```

参考文献&致谢

本文写作过程中得到了很多帮助，下面这些链接中也包含了这个问题的详细解析，供大家参考

- http://www.jbxue.com/db/13432.html
- http://stackoverflow.com/questions/19679594/sql-syntax-error-incorrect-syntax
- http://bugs.mysql.com/bug.php?id=74834 *http://en.wikipedia.org/wiki/Stored_procedure

---

关于完整代码

完整的代码已经上传在我的Github小项目合集，文件夹名：SQL stored procedure--iterate btw rows中，如果这篇文章侥幸对你有一点帮助，请移步Github帮我点亮一颗星星呗。♥