后盾网 人人做后盾

www.houdunwang.com

MYSQL事务、视图、存储过程

后盾网 2011-2013 v3.0

MyISAM

- MyISAM引擎的表存储空间受硬盘大小限制，不支持事务，查询操作是性能强悍，索引压缩，各平台间文件通用，支持表级锁

InnoDB

- 提供更高级的事务处理及外键约束，支持行级锁，如果有事务需求比如说金融股票类项目使用InnoDB存储引擎

区别

- MySQL支持同一时间多个用户访问数据表，MyISAM引擎是表级锁，所以多个用户进行写操作会带来性能下降，InnoDB采用行级锁，可以支持更大的并发操作

存储引擎

事务

- 事务是一组原子性的查询语言，也就是说在一组操作中，要么全部执行，要么一个也不执行，如果任何一个操作失败，则所有操作都失效，保证数据的完整性如银行转帐系统
- 保证数据在操作时不会被更改，如对同一个表执行一次INSERT再执行一次DELETE，虽然执行DML操作可以锁定表，但在两条语句执行中间可能有其他客户端执行了DML操作，有可能造成当前CLIENT第2个SQL有误

存储引擎

- MYSQL支持事件的存储引擎有InnoDB、NDB和BDB

查看MYSQL支持的存储引擎命令

- show engines

修改表引擎

- alter table demo engine= innodb

事务

Mysql的工作模式

- Mysql默认是自动模式，即一条命令执行就马上生效（这一条命令可以看成一个事务）。为了执行事务，应该关闭自动提交，将多个语句组合成一个事务，最后执行提交或回滚操作。

关闭自动提交模式，启动手动提交（开启事务）

- start transaction或BEGIN
- set autocommit=0 不自动提交

开启自动提交

- set autocommit=1;

注：

- 以start transaction开启事务时，如果原来是自动提交模式，那么当执行完commit后，系统将自动切换到自动提交模式。而set autocommit=0则不会
- set autocommit、commit、begin、alter、create、rollback、等命令会自动提交事务

事务

事务的准备工作

- 选择支持事务的数据引擎
- 通过以上一种方式开启事务模式

事务的回滚rollback

- create table d(name char(30)) engine innodb;
- set autocommit=0;关闭自动提交模式
- insert into d(name) values(22);
- rollback;回滚

事务的提交

- create table d(name char(30)) engine innodb;
- set autocommit=0;关闭自动提交模式
- insert into f(name) values(22);
- commit;提交事务

事务实例

视图

- 视图是一张虚表，是一组查询记录
- 将部分表字段分配到视图中，重要字段不分配，提高数据库安全性

查看视图

- show table status where comment= ‘VIEW’ ;

创建视图

- CREATE VIEW stuview AS select name from stu;

删除视图

- drop view stuview

修改视图

- alter view stuview as select id,name,age from stu;

视图

声明变量

SET

- 定义全局变量
- set @name= ‘后盾网’;

DECLARE(函数或过程中使用)

- DECLARE var_name[,...] type [DEFAULT value]
- 函数与存储过程中通过DECLARE声明局部变量,局部变量的作用范围在它被声明的BEGIN ... END块内
- DECLARE a int; #声明局部变量a为int数据类型
- DECLARE webname char(10) default ‘后盾‘; #声明局部变量，默认值为’后盾’

SELECT ... INTO ..把选定的列直接存储到变量。

- SELECT id,data INTO x,y FROM test.t1 LIMIT 1;
- 注意: SQL变量名不能和列名一样

存储过程

- 把代码进行封装，便于多次使用或多种应用程序共享使用
- 使用存储过程可以明显提高系统运行效率
- 储存过程没有返回值
- 执行结果可以返回结果集
- 不能用在SQL语句中，只能使用CALL调用

结束符

- 由于存储过程内部以分号结束，所以在定义存储过程前要将MYSQL的结束符进行修改，可以采用任何不冲突字符，如下
- delimiter $$或delimiter EOF
- \d $$也可以更换结束符

存储过程

- [begin_label:] BEGIN
- [statement_list]
- END [end_label]

存储子程序使用BEGIN ... END复合语句来包含多个语句。
statement_list 代表一个或多个语句的列表。statement_list之内每个语句都必须用分号（；）来结尾。

BEGIN ... END

储存过程的参数

IN　　向储存过程传入值(如果没有指定此为默认值）

OUT　向out参数赋值，调用结束后可以访问到这个变量

INOUT　　传入一个值，并且将值返回给外部

```
1. create procedure getcid(inout p_name char(10) charset utf8)
2. begin
3. select cid into p_name from hd_user where uname=p_name ;
4. End
5. $
6. set @name='李四’ $
7. call getcid(@name)$                    #调用储存过程
8. select @name$                                  #输出结果
```

存储过程

```
IF search_condition THEN statement_list
    [ELSEIF search_condition THEN statement_list] ...
    [ELSE statement_list]
END IF
```

示例

```
\d $
create procedure hd(in arg tinyint)
begin
declare age tinyint default 0;
set age = arg;
IF age<20 THEN
select "年轻人";
ELSEIF age<40 THEN
select "青年人";
END IF;
end
$
```

IF语句

```
1. \d $
2. create procedure hd(inout arg int)
3. begin
4.     declare i int default 0;
5.     set i = arg;
6.     case i
7.     when 1 then
8.     select "sina";
9.     when 2 then
10.    select "baidu";
11.    else
12.    select "houdunwang";
13.    end case;
14.end;
15.$
16.\d ;
17.set @s = 2;
18.call hd(@s);
```

case语句

```
1. \d $
2. create procedure createStu(in num int)
3. begin
4. declare i int default 0;
5. while num>0 do
6. select  i;
7. set num=num-1;
8. end while;
9. end
10.$
```

while语句

存储过程

创建存储过程

```
1. Delimiter $$
2. create procedure pro_login()
3. begin
4. select * from hd_stu;
5. End;$$
```

查看存储过程

```
SHOW PROCEDURE STATUS\G
```

调用存储过程

```
call pro_login()$$
```

删除存储过程

```
DROP PROCEDURE pro_login;
```

注：如果储存过程只有一行语句，begin 与end可以省略，但建议一直采用，哪怕只有一行SQL时

添加用户

- \d $
- create procedure add_stu(in _sname varchar(100) charset utf8,in _sex int)
- begin
- 	insert into stu set sname=_sname,sex=_sex;
- end;
- $
- \d ;
- call add_stu('向军',1);

存储过程

存储函数

创建储存函数

1. \d $
2. CREATE FUNCTION hello (s CHAR(20))　　创建储存函数hello
3. RETURNS CHAR(50) reads sql data　　返回值数据类型
4. RETURN CONCAT(‘Hello’ ,s, ‘!’); 返回结果
5. $
6. SELECT hello(‘houdunwang’);

注：函数必须至少有一个return

删除储存函数

- drop function 储存函数名

显示储存函数

- show function status\G

通过存储函数求得学生id

```
1. \d $
2. create function func_geid(username char(30))
3. returns int
4. reads sql data
5. begin
6.     if username is null then
7.     return null;
8.     else
9.     return (select id from stu where name=username);
10.    end if;
11.end
12.$
```

存储函数

当表执行INSERT、UPDATE、DELETE执行前或执行后，通过触发器完成一些语句的自动执行

触发器的作用

- 检测插入数据的正确性
- 可以将执行结果赋值给列，做为默认值使用
- 可以控制数据的完整性

查看触发器

- SHOW TRIGGERS\G

删除触发器

- drop trigger 触发器名称

触发器

```
1. create trigger afterDelete_hd_class after delete on hd_class
2. for each row
3. Begin
4. delete from hd_user where cid=old.cid;
5. End
6. $
```

以上触发器实现，在删除班级表记录时，将此班级中的学生一并删除操作

使用new.列名表示，UPDATE、INSERT操作时的新数据

使用old.列名给示，UPDATE、INSERT操作时的旧数据

创建触发器

```
1. \d $
2. create trigger ib_hduser before insert on hd_user
3. for each row
4. Begin
5. if new.uname is null then
6. set new.uname='后盾';
7. end if ;
8. End;
9. $
```

创建触发器

事务

1. 模拟银行交易打款的事务过程，如果李四向张三打3000元，那么只有当李四帐户钱减少成功且张三确实收到钱的执行事务
2. 否则事务执行失败

存储过程

1. 创建删除班级的存储过程
2. 实现删除班级时一并删除此班级中的学生
3. 调用方式call del_class(1);

触发器

1. 创建文章表含标题、作者、发布时间字段
2. 使用触发器完成，如果只添加了标题，发布时间字段自动设置为当前时间，作者字段设置为后盾网

作业