# 二 mysql高级

## 1 视图

对于复杂的查询，在多次使用后，维护是一件非常麻烦的事情，可以使用视图查询，视图本质就是对查询的一个封装。

定义视图

create view stuscore as

select students.\*,scores.score from scores

inner join students on scores.stuid=students.id;

查询

select \* from stuscore;

修改视图

将create改为alter

## 2 存储过程

### 2.1 创建存储名为test的过程

create procedure test()

begin

select Avg(product\_price) as priceavg from products;

end;

注意：默认mysql分隔符为 ; ，如果使用的是mysql命令行，要解析存储过程中自身的 ; ，会出现语法错误，解决办法：使用delimiter关键字。

delimiter //

create procedure test()

begin

select Avg(product\_price) as priceavg from products;

end //

这里使用 //代替 ;，除了 / 外，都可以作为分隔符。

### 2.2 使用test存储过程

call test();

### 2.3删除test存储过程

drop procedure test;

如果不存在该存储过程，会出现错误。

### 2.4 带参数的存储过程

create procedure productpriceing(

out p1 DECIMAL(8,2),

out ph DECIMAL(8,2),

out pa DECIMAL(8,2)

)

begin

select Min(product\_price) into p1 from products;

select Max(product\_price) into ph from products;

select Avg(product\_price) into pa from products;

end;

解析：该存储过程接收3个参数，p1存储最低价格，ph存储最高价格，pa存储平均价格，每个参数必须具有指定的类型，这里是十进制。关键字out指出对应参数用来从存储过程中传出一个值给调用者。

in：传递给存储过程

out：传出给调用者

inout：传入和传出

使用带参数存储过程：

call productpricing(

@pricelow,

@pricehigh,

@priceavg

);

mysql的变量使用@，该语句并不现实任何数据，只返回以后可以显示的变量，如果药检所产品的平均价格，如下进行：

select @priceavg;

## 2 事务

### 2.1 事务简介

当一个业务逻辑需要多个sql完成时，如果其中某条sql语句出错，则希望整个操作都退回，使用事务可以完成退回的功能，保证业务逻辑的正确性。

事务四大特性(简称ACID)：

原子性(Atomicity)：

事务的全部操作在数据库中不可分割，要么全部完成，要么均不执行；

一致性(Consistency)：

几个并行执行的事务，其执行结果必须与按某一顺序串行执行的结果相一致；

隔离性(Isolation)：

事务的执行不受其他事务的干扰，事务执行的中间结果对其他事务必须是透明的

持久性(Durability)：

对于已提交事务，系统必须保证该事务对数据库的改变不被丢失，即使数据库故障；

### 9.2 事务使用前提

要求：表的类型必须是innodb或bdb类型，才可以对此表使用事务，一般默认是innodb。

inndb原理：对当前操作的行加锁；适合插入增删改（写入速度快）；

其他引擎常见的是对整张表加锁，查询速度很快。

### 9.3 事务的使用

查看表的创建语句

show create table students;

修改表的类型

alter table '表名' engine=innodb;

事务语句

开启begin;

开始对一个表进行增删改写入操作；

提交commit;

开启一个内存级临时表，将写入操作写入该临时表；

回滚rollback;

放弃begin后的所有操作；

### 9.4 示例

示例1

步骤1：打开两个终端，连接mysql，使用同一个数据库，操作同一张表

终端1：

select \* from students;

------------------------

终端2：

begin;

insert into students(sname) values('张飞');

步骤2

终端1：

select \* from students;

步骤3

终端2：

commit;

------------------------

终端1：

select \* from students;

示例2

步骤1：打开两个终端，连接mysql，使用同一个数据库，操作同一张表

终端1：

select \* from students;

------------------------

终端2：

begin;

insert into students(sname) values('张飞');

步骤2

终端1：

select \* from students;

步骤3

终端2：

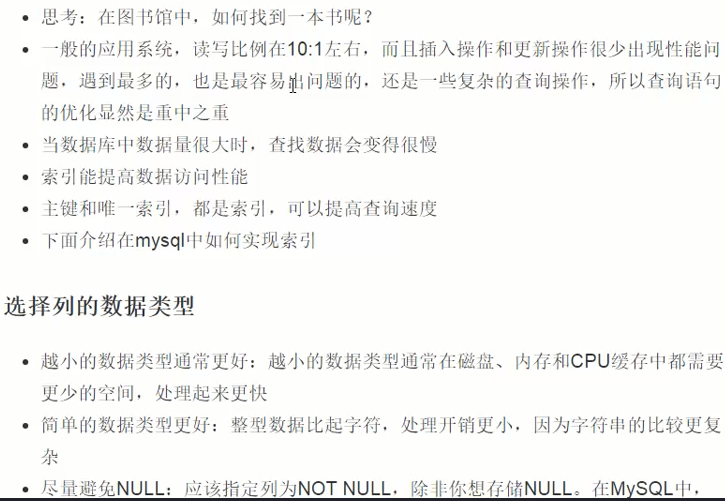
rollback;

------------------------

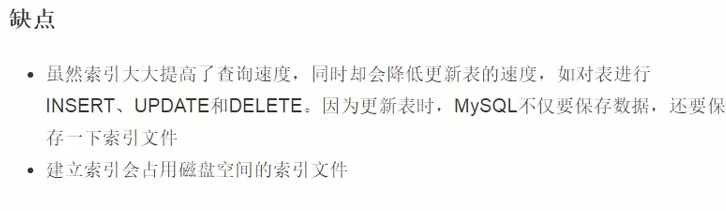
终端1：

select \* from students;

## 3 索引



数据检索如果大于100ms就可以开始考虑优化了。主要优化where后面的语句。



## 1 全文本搜索

通过正则、通配符可以实现复杂的搜索，但是随着搜索行数的增加，搜索将会变得耗时，且并不能智能的选择结果，比如：一个特殊词的搜索将会返回包含该词的所有行，而不区分包含单个匹配的行和包含多个匹配的行，类似的，一个特殊词的搜索将不会找出不包含该词但包含其他相关词的行。所有这些限制都可以通过全文检索来实现。

并非所有引擎支持全文本搜索，InnoDB不支持，MyISAM支持。

### 1.1 启用全文本搜索

为了进行全文本搜索，必须索引被搜索的列，而且要随着数据的改变不断重新索引。在堆表列进行适当设计后，MySQL会自动进行所有索引和重新设计。在索引之后，select可与Match()和Against()一起使用以实际执行搜索。

一般在创建表时启用全文本搜索：

create table products(

product\_id int not null auto\_increment,

product\_num char(10) not null,

product\_date datetime not null,

product\_text text null,

primary key(product\_id),

fulltext(product\_text)

)

engine=MyISAM;

fulltext索引单个列，如果需要也可以指定多个列。在定义之后，MySQL自动维护，增删改时，随之自动更新。

可以在创建时候指定fulltext，也可以稍后指定（此时必须所有已有数据立即索引）。

注意：更新索引需要花时间，如果正在导入数据到一个新表，此时不应该启用fulltext索引，应该先导入所有数据，然后再修改表，定义fulltext，这样有助于快速导入数据。

### 1.2 行全文本搜索

在索引后，使用Macth()和Against()执行全文本搜索，Match()指定被搜索的列，Against(0指定要使用的搜索表达式。

select product\_id from products

where Match(product\_text) Against('k');

## 6 备份恢复

备份：

进入超级管理员 sudo -s

进入mysql库目录 cd /var/lib/mysql

运行mysqldump命令 mysqldump –uroot –p 数据库名 > ~/Desktop/备份文件.sql;

恢复：

连接mysq，创建数据库

退出连接，执行如下命令：mysql -uroot –p 数据库名 < ~/Desktop/备份文件.sql

## 7 逻辑删除

对于重要数据，并不希望物理删除，一旦删除，数据无法找回，一般对于重要数据，会设置一个isDelete（有人喜欢用deleteFlag）的列，类型为bit，表示逻辑删除。

删除数据的时候，是把要删除的数据移动到另外一个库内，这个库的表结构和项目库一样，里面保存的全是删除的数据。

## 数据库安全

数据库在实际业务中绝对不可能使用root账户登录，需要给不同的使用人员分配不同权限的账户。

创建新用户：

create user mary indentified by ‘p@$$w0rd’;

设置访问权限：刚创建的用户只能登录mysql，不能看到数据，也不能执行操作。

show grants for mali; //显示用户mali有什么权限

grant select on testbase.\* to mali; //mali用户可对testbase数据库所有表使用select

// move用来撤销权限，用法同 grant

查询用户列表：

select user from user; //user表中有一个user列，用来存储登录名

重新命名一个表：

rename user mary to mali;

删除用户：

drop user mali;