第一章 MySQL的基本概念和安装操作

1. 数据库是“按照数据结构来组织、存储和管理的仓库”

响应和提供数据

应用程序

数据库

数据库服务器

操作和查询

1. 什么是MySQL？
2. MySQL是一个小型关系数据库管理系统，开发者为瑞典MySQL AB公司，属于Oracle旗下产品。
3. 应用于：中小型网站，其体积小，速度快，总体拥有成本低，开放源码

* **Oracle**  **例：银行**
  + 应用广泛，功能强大，分布式数据库系统
  + “关系----对象” 型数据库
* **MySQL** **例：中小型网站**
  + 快捷，可靠，容易使用
  + 开源、免费、体积小、速度快、成本低
* **SqlServer**  **例：只适用Windows操作系统**
  + 针对不同用户群体的五个特殊的版本
  + 易用性好
* **DB2**
  + 应用于大型应用系统，具有较好的可伸缩性

1. 安装与配置

（1）解压安装包到安装目录（一定要修改配置文件）

配置文件：

basedir=mysql安装的目录

datadir=mysql安装目录\data

（2）随机密码登录

随机密码：.err后缀名文件

初始化mysql： mysql --initialize

安装mysql服务： mysqld --install

使用初始密码登录： mysql -uroot -p (若使用初始密码登录出错)

修改加密规则： alter user ’root’ @ ’localhost’ identified by ’password’ expire never

修改初始化密码： alter user ’root’ @ ’localhost’ identified with mysql \_native \_password by ’新密码’;

进行登录： mysql -uroot -p密码

退出： Ctrl+Z

关闭MySQL服务器： net stop mysql

初始化MySQL：生成data目录 ，删掉，重新执行初始化命令

（3）删除服务

黑窗口：sc delete MySQL 删除mysql

注册表 开始--搜索--regedit--注册编辑表--编辑查看（mysql）--删除跟mysql相关的文件

（4）安装目录

目录里面不能出现空格，不能出现中文

（5）环境配置

此电脑--属性--高级系统设置--环境变量--系统变量--path(新建 mysql的安装路径\bin)

（6）服务设置

此电脑--管理--服务--找到MySQL服务 可以启动关闭服务

（7）安装版的MySQL

傻瓜式安装

4、图形化工具

Navicat的安装（注意：把破解文件复制到Navicat安装目录下）

1.安装 傻瓜式安装（一直点下一步就可以）

2.破解 x64 复制到 包含有navicat.exe目录下

3.连接测试

连接--mysql--连接名（随便起）--用户名root--密码自己设置的--连接测试

第二章数据库及数据表相关操作

1数据库的概念

库：将物品有条理的存放

数据库：存放数据的一个仓库

2.检测mysql命令行

注释 #

select current\_user; #查询当前登录的用户

select now(); #查询当前的时间

show databases; #显示所有数据库

select version(); #检测MySQL安装的版本

3.数据库命令行

create database 数据库名; #创建数据库

drop database 数据库名； #删除数据库

drop database if exists 数据库名 ; #判断是否存在再进行删除

4.mysql数据类型

在java中的数据类型：基本数据类型 整数 byte short int long、小数 float double 、布尔boolean 、字符char

引用数据类型 字符串 String 数组 对象

Mysql常用的数据类型：数值类型 int tinyint、字符串类型 char varchar、日期时间类型 datetime 、小数 float double

（1）数值列类型，存储数值型的数据记录

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **类型** | **大小** | **范围(有符号)** | **范围(无符号)** | **用途** |
| TINYINT | 1字节 | (-128,127) | (0,255) | 小整数值 |
| SMALLINT | 2字节 | (-32768,32767) | （0,65535) | 大整数值 |
| MEDIUMINT | 3字节 | (-8388608,8088607) | (0,4294967295) | 大整数值 |
| Int或integer | 4字节 | (-2147483648,2147483647) | (0,4294967295) | 大整数值 |
| Bigint | 8字节 | (-9233372036854775808,  9233372036854775807) | (0,18446744073709551615) | 极大整数值 |
| float | 4字节 | E -38),0,(1.175494351E -38,3402823466351 E + 38) | 0,(1.175494351 E - 38,3402823466 E + 38) | 单精度浮点数值 |
| double | 8字节 | (-1.7976931348623157 E + 308,-2.2250738585072014 E -308),0,(2.2250738585072014 E -308,1.7976931348623157 E +308) | 0,(2.2250738585072014 E -308,1.797693134 862 315 7E +308) | 双精度浮点数值 |
| decimal | 对decimal(M,D),如果M>D,为M+2否则为D+2 | 依赖M和D的值 | 依赖M和D的值 | 小数值 |

（2）字符串类型，存储字符串型的数据记录

|  |  |  |
| --- | --- | --- |
| **类型** | **大小** | **用途** |
| Char | 0~255字节 | 定长字符串 |
| Varchar | 0~65535字节 | 变长字符串 |
| tinybLob | 0~255字节 | 不超过255字节的二进制字符串 |
| tinytext | 0~255字节 | 短文本字符串 |
| BloB | 0~65535字节 | 二进制形式的文本数据 |
| text | 0~65535字节 | 长文本数据 |
| mediumblob | 0~16777215字节 | 二进制形式的中等长度文本数据 |
| mediumtext | 0~16777215字节 | 中等长度文本数据 |
| Longblob | 0~4294967295字节 | 二进制形式的极大文本数据 |
| longtext | 0~4294967295字节 | 极大文本数据 |

1. 日期时间类型，存储日期时间型的数据类型

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **类型** | **大小(字节)** | **范围** | **格式** | **用途** |
| data | 3 | 1000-01-01/9999-12-31 | YYYY-MM-DD | 日期值 |
| time | 3 | -838:59:59/838:59:59 | HH:MM:SS | 时间值或持续时间 |
| year | 3 | 1901/2155 | YYYY | 年份值 |
| datatime | 8 | 1000-01-01:00:00:00/9999-12-31  23:59:59 | YYYY-MM-DD  HH:MM:SS | 混合日期和  时间值 |
| timestamp | 4 | 1971-01-01 00:00:00/2038  结束时间是第2147483674秒，北京时间2038:1-19 11:14:07,格林尼治时间2038年1月19日，凌晨03:14:07 | YYYYMMDD  HHMMSS | 混合日期和时间值 时间戳 |

5.完整性约束条件

primary key 主键 唯一的

foreign key 外键

not null 不为空（非空）

unique 唯一（保证数据不重复）

auto\_increment 值自动增加，一个表中只能有一个字段用它来约束，且必为主键

default 默认值

check 检查约束 检查我们输入的数据是否满足设置的条件

6.创建表

create table 表名(

列名 数据类型 约束,

...,

列名 数据类型 约束

);

7.desc 表名; 查看表中的所有列

8.删除表

drop table 表名;

drop table if exists 表名; 先判断是否存在再进行删除

第三章 MySQL操作数据表记录

1、表中插入数据

•格式1：insert into 表名(字段1,字段2,...字段n) values(字段值1,字段值2,...字段值n);

例：desc books;

insert into books(bno,bname,author,quanitity) values (1001,”红楼梦”,”曹雪芹”,100),(1002,”西游记”,”吴承恩”,600);

•格式2：insert into 表名 values(列1的值,列2的值,...列n的值);

注意省略列的时候，后面值应与建表时列的顺序一致

例：

insert into books values(1002,”西游记”,”吴承恩”,600);

•格式3：insert into 表名 values(列1的值,列2的值,...列n的值),(列1的值,列2的值,...列n的值);

多条语句的添加，多条数据之间用逗号隔开

例：

insert into books values(1003,”西游记”,”吴承恩”,700),(1004,”三体”,”刘慈欣”,50);

2、查询表中的数据 \*代表所有列

select \* from 表名 [where条件表达式];

select 属性列表 from 表名 [where 条件表达式];

例：

select \* from books;#查询图书表数据

select bname,author from books;#想要搜什么

select \* from books where quanitity>100;#查询书籍数量大于100的书

select bname from books where id=1002; 查什么 在哪里查 是否有条件

3、修改数据，更新 (不添加条件的话，会修改表中所有信息)

update 表名 set 字段名1=字段1值,字段名2=字段2值,...字段名n=字段n值 [where条件表达式];

例：#更改名为红楼梦的书籍数量为300

update books set quanitity=300 where bname=”红楼梦” ;

#所有记录被更新，不添加条件

update books set quanitity=300;

4、删除数据

delete from 表名 [where条件表达式];

例：#某个书籍下架，删除这个书籍的序号及所有信息数据

select \* from books;

delete from books where bno = 1001;

#不添加条件的话，表中所有信息都会被删除

delete from books;

1. 外键（依赖于某一张表）

在与前面表格相连中后加references 表格名(哪列) #外键约束

主要是维护表之间的关系的，保证参照完整性，如果表中的某个字段为外键字段，那么该字段的值必须来源于参照的表的主键

学生表student 外键classno 班号

班级表class 主键classno 班号

学生表中：classno int(4) references class(classno

注意：一个表中可以有多个外键约束，添加外键的值一定要来源于参照表的主键的值

删除的时候 先删子表，再删父表

第四章

一.条件查询 where

1.等值查询 =

select \* from student where id=1;

2.不等值查询 != (< >)

查询年龄不是20岁的学生信息

select \* from student where age!=20;

3.关系比较查询 < <= > >=

查询年龄不是20岁的学生信息

select \* from student where age>20;

4.逻辑运算查询 and or

and 满足所有条件 or只需要满足其中一个条件

查询年龄是小于20岁并且住址是信阳

select \* from student where age<20 and address="信阳";

查询年龄小于20岁或者住址是信阳的

select \* from student where age<20 or address="信阳";

5.区间查询 between .. and (包含边界值)

查询年龄是18岁和21岁之间

select \* from student where age between 18 and 21;

select \* from student where age>18 and age <21;

6.模糊查询 like

① %（匹配0-n个字符）

查询学生表中名字包含x的学生信息

select \* from student where stuname like '%x%';

② \_ （匹配一个字符,相当于一个占位符）

查询学生表中地址第二个字为京的信息

select \* from student where adderss like '\_京';

二.排序查询 order by asc升序(默认) desc降序

语法：select \* from 表名 （where 条件） order by 排序的字段 asc/desc;

1.查询书籍信息表按照书价从高到低排序（降序）

select \* from books order by price desc;

注意：排序条件可以是多个字段，多个字段之间使用逗号隔开，asc/desc关键字作用于前面的字段

2.limit 限制查询几条信息

limit 1个参数 limit 2个参数

第一个参数代表从哪里开始，第二个参数代表要查询几条数据（mysql中数据从0开始数）

查询前3条数据 select \* from 表名 limit 3;

查询从第2条开始的3条数据 select \* from 表名 limit 1,3;

#查询最贵的两本书 select \* from books order by price desc limit 2;

三.聚合函数 是针对于一组数据产生一个结果

1.sum() 求和

语法：select sum(字段名) from 表名;

查询所有学生年龄总和

select sum(age) from student;

1. avg() 求平均值

语法：select avg(字段名) from 表名;

查询所有学生年龄平均值

select avg(age) from student;

1. max() 求最大值

语法：select max(字段名) from 表名;

查询所有学生年龄最大的

select max(age) from student;

1. min() 求最小值

语法：select min(字段名) from 表名;

查询所有学生年龄最小的

select min(age) from student;

5.count() 统计查询

语法:查询所有数据 两种写法等效

select count(\*) from 表名;

select count(1) from 表名;

注意：如果指定字段的值为空，会自动忽略

四.分组语句 group by

语法：select 字段名 from 表名 group by 分组条件;

说明：group by 后面跟分组条件字段

查询班级信息分组显示

select className from class group by className;

书写注意事项

①只有出现在group by后面的字段，才能单独出现在select后面

②如果要在group by语句种进行结果的排序，那么排序字段也必须是group by后指定的字段和结果的字段

③如果group by子句种的字段使用了某个单行函数，那么select子句种的字段也必须使用相同的单行函数

五.having语句 对分组后的数据再次进行筛选(相当于where加条件,和聚合函数混合使用)

语法：select 字段 from 表名 where 条件 group by 分组条件 having 条件;

查询借书卡信息表种班级人数大于1的班级

select class from card group by class having count(cno)>1; #先对班级进行一个分组，在使用聚合函数计算班级人数

注意：①和聚合函数混合使用只能用having语句

②当使用where或having都能满足一个需求时，优先使用where

六.起别名

语法：① select 字段名1 as 新名字,字段名2 as 新名字... from 表名;

② select 字段名1 新名字,字段名2 新名字... from 表名;

③ select 字段名1 "新名字",字段名2 "新名字"... from 表名;

七.其他

1.表达式 计算总价

创表的时候 设计价格这个字段

将float类型改为decimal(数值1,数值2) 数值1代表小数点前面整数位为多少位，数值2代表小数点后面保留几位小数

查询总价格

select 单价\*数量 from 表名;

1. 去重 distinct（如果遇到重复的去除掉，只显示一遍）

语法：select distinct 字段 from 表名;

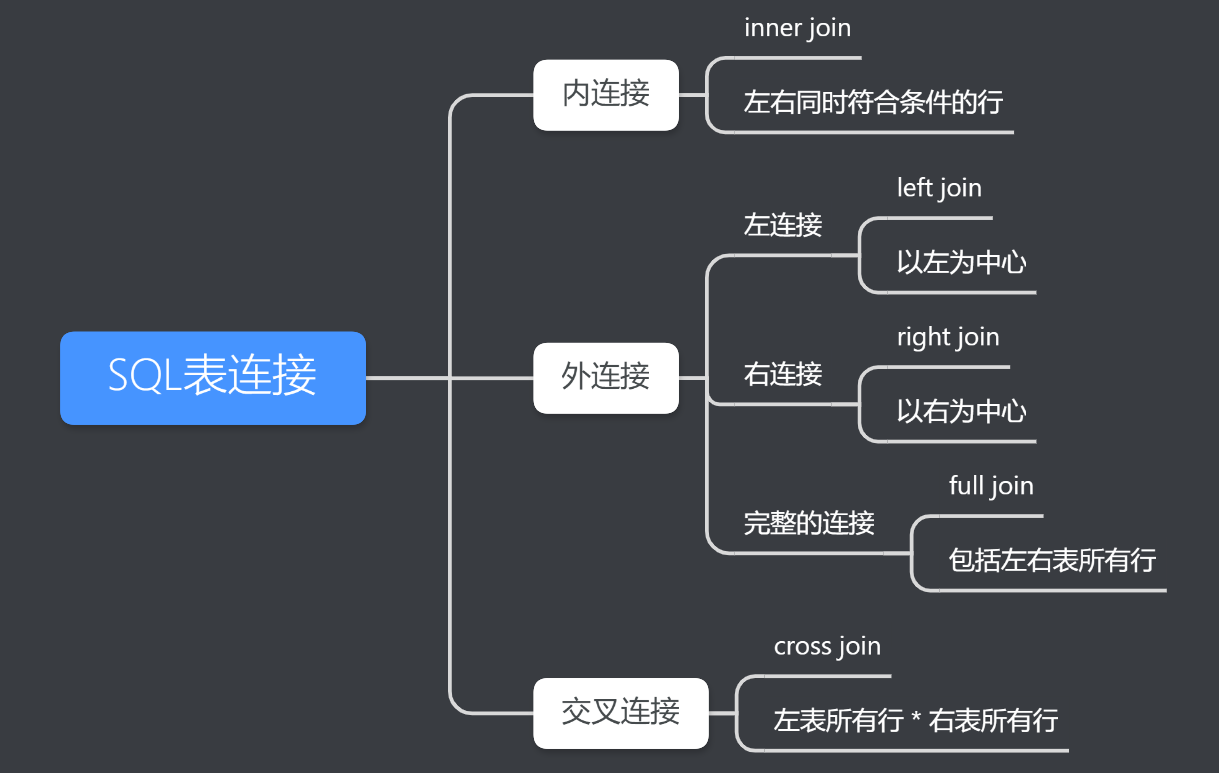
查询借阅卡表中包含的班级有哪些

select distinct class from card;

第五章MySQL多表连接查询

1、连接查询：同时涉及多个表的查询

连接条件：用来连接两个表的条件 表1.字段=表2.字段



2、内连接 inner join

特点：两个表交叉的部分被查询出来

语法：select \* from 表1 inner join 表2 on 表1.字段=表2.字段;

select 字段... from 表1 别名1 inner join 表2 别名2 on 连接条件;

例：查询每个学生的选课成绩

select \* from student inner join stucon on student.stuno = stucou.stuno;

select \* from student s1 inner join stucon s2 on s1.stuno = s2.stuno;

3、外连接

①左外连接 left[outer] join [outer]可以被省略

特点：左表中的记录全部出现在结果中，如果对应的某行记录在右表中没有对应关系，则右表会自动补空（值）

语法：select \* from 表1 left join 表2 on 表1.字段=表2.字段;

select 字段... from 表1 别名1 left join 表2 别名2 on 连接条件;

例：查询每个学生的选课成绩

select \* from student left join stucon on student.stuno = stucou.stuno;

select \* from student s1 left join stucon s2 on s1.stuno = s2.stuno;

②右外连接 right[outer] join

特点：右表中的记录全部出现在结果中，如果对应的某行记录在左表中没有对应关系，则左表会自动补空

语法：select \* from 表1 inner join 表2 on 表1.字段=表2.字段;

select 字段... from 表1 别名1 inner join 表2 别名2 on 连接条件;

例：查询每个学生的选课成绩

select \* from student right join stucon on student.stuno = stucou.stuno;

select \* from student s1 right join stucon s2 on s1.stuno = s2.stuno;

4、交叉连接 cross join

特点：A表中的记录一一匹配B表中的所有记录

语法：select 字段... from 表1 cross join 表2 ;

例：查询总条数

select \* from student cross join stucou;

select count(studnet.stuno) as 总条数 from student cross join stucou;

5、where外键条件多表查询 多个条件之间用and连接

特点：相当于内连接，效果一致

语法：select \* from 表1 别名1,表2 别名2,表3 别名3... where 外键条件(关系) and 条件1 and 条件2...;

例：

#查询Java课程的所有得分

select \* from course a,stucou b where a.count=b.count and couname=”Java课程”;

#查询选修Java课程的平均分

select avg(score) as Java编程的平均分 from course a,stucou b where a.count=b.count and couno=”002” couname=”Java课程”;

#查询孙悟空的数据库基础的成绩

select \* from student s1 ,course s2,stucou s3 where s1.stuno=s2.stuno and s2.couno=s3.couno and stuname=”孙悟空” and couname=”数据库基础”;

第六章MySQL常用函数

1.数值型函数

①round(x,y) 四舍五入 x为查询数值的字段，y为保留几位小数的值

特点：返回数值x带有y为小数结果的数值

语法：select round(查询数值字段,y) from 表名;

例：

#价格四舍五入

select round(price) as 价格 from books;

②floor(x) 向下取整 小数位全舍去，取整数部分

特点：返回x的向下取整的整数

语法：select floor(查询字段) from 表名;

例：

#价格四舍五入,保留1位小数

select floor(price,1) as 价格,bname from books;

③ceil(x) 向上取整 小数位全舍去，取原整数部分+1

特点：返回x的向上取整的整数

语法：select ceil(查询字段) from 表名;

例：

#价格四舍五入

select ceil(price) as 价格,bname from books;

④truncate(x,y) 数值截取 x为查询数值的字段(表达式) y为保留几位小数的值

特点：返回数值x截取y位小数的结果（不四舍五入）

select truncate(字段或表达式,数值2) from 表名;

例：

#价格打78折

select truncate(price\*0.78,1) as 打折后价格,bname from books;

2.字符型函数

①concat(str1,str2,str3...) 合并字符串

特点：将多个字符串合并成一个字符串

语法：select concat(字段1,字段2...) from 表名;

例：

#合并书名和作者

select concat(bname,author) as 合并显示的结果 from books;

②upper(x)

特点：用于将字母转成大写

语法：update 表名 set 字段名=upper(字段名) where 条件;

例：

#更新图书信息,把书名全部更新成原书名的大写

update books set bname=upper(bname);

③lower(x)

特点：用于将字母转成小写

语法：update 表名 set 字段名=lower(字段名) where 条件;

例：

#更新图书信息,把书名全部更新成原书名的小写

update books set bname=lower(bname) where cno=001;

④reverse(x)

特点:用于将字符串翻转

语法:update 表名 set 字段名=reverse(字段名) where 条件;

例：

#更新西游记这本图书的书名,更新成原书名的翻转之后的结果

update books set bname=reverse(bname) where bname=”西游记”;

3.时间日期函数

①curdate()

特点：返回当前日期 只返回年月日，没有时分秒

Insert into info (cno,bno,rdate) values (105,1006,curdate());

②now()

特点：返回当前日期 返回年月日 时分秒

③year(date)

特点：从日期中选择出年份

语法：select year(查询字段) from 表名 where 条件;

select year(rdate) as 年份,cno,bno from info where bno=1006;

④date\_format(date,'%Y-%m-%d')

特点：%Y-%m-%d把日期格式调整为年-月-日的格式 Y为四位数的年份 %h时 %i分 %s秒

语法：select date\_format(字段,'%Y-%m-%d') from 表名;

4.流程控制类函数

①if(v,t,f)

特点：如果v为真返回t，否则返回f

例:

查看张三是不是大数据2班的是就显示是，否则显示否

Select if(class=”大数据2班”,”是”,”否”) as 结果 from card where name=”张三”;

select if(1!=1,'1','2') from DUAL; 结果：输出2

②ifnull(v1,v2)

特点：如果v1不为null返回v1,否则返回v2

例:select ifnull(null,'1') from DUAL; 结果：输出1

③case when v then v1 end

特点：判断如果v为真返回v1

例：select case when 1=1 then 'true' else 'false' end from DUAL; 结果：输出true

5.数据库类函数

①database() 返回当前数据库名称 select database();

②version() 返回当前数据库版本 select version();

③md5(s) 返回MD5加密值

语法：假设将字段值1加密

insert into 表名(列1,列2...) values(md5(字段值1),字段值2);

例：

#插入一本书，新书的名称要经过加密

Insert into books (bname,author,price,quanitity) values (MD5(‘西游记’),’吴承恩’,20,10);

第七章MySQL高级对象

一.MySQL视图

1.定义：视图就是一条select语句执行后返回的结果集

2.使用场景：权限控制的时候，不希望用户访问表中某些含敏感信息的列，比如salary工资

3.语法（创建）：create view 视图名称

as

select 语句

例：

建立用于查询借阅记录中借阅人姓名，借阅书籍名称，借阅时间的视图

create view view\_info

as

select name as 姓名,bname as 名称,rdate as 借阅时间 from books a,card b,info c where a.bno=c.bno and b.cno=c.cno;#多条查询 起别名

#成功视图查看

show tables;

#使用 虚拟表

select \* from view\_info;

（修改）：alter view 视图名称

as

select 语句

例：姓名，书名

alter view view\_info

as

select name as 姓名,bname as 名称 from books a,card b,info c where a.bno=c.bno and b.cno=c.cno;

（删除）：drop view 视图名; 删除视图

drop view if exists 视图名称; 先判断是否存在再进行删除

4.查看 show tables;

注意：视图创建成功后可以在表展示中显示，但实际不存在，相当于一个虚拟表，使用方法按照表操作进行

二.MySQL存储过程

1.定义：存储在数据库当中可以执行特定工作（查询和更新）的一组SQL代码的程序段。类似于功能函数

2.语法（创建）：create procedure 存储过程名称(参数列表)

begin

存储操作的语句块

end

注意：当语句结束;有冲突时使用delimiter定义新的结束方法

（调用）：call 存储过程名称;

（有参调用）：call 存储过程名称(参数1,参数2,..);

（查看所有的存储过程）：show procedure status;

（删除存储过程）：drop procedure if exists 存储过程名称;

3.参数分两种

①in 给参数传入值，定义的参数就得到了值

语法：in 参数名 表中字段的类型

②out 返回值

语法：out 参数名 返回的数据类型

4.存储过程分为

①无参无返回值的存储过程

②有参无返回值的存储过程

③无参有返回值的存储过程

④有参有返回值的存储过程

例1：

创建一个无参无返回值的存储过程,用于查询所有借阅记录

#修改sql语句结束标识符

delimiter//

create proceduce pro\_info()

begin

select \* from info;

end//

--调用存储结构

call pro\_info;

例2：

创建一个有参无返回值的存储过程，用于查询某个价格(20-200)范围内的信息

Select \* from books where price between 20 and 200;

#有参数 一个价格 2个价格

--in 名字/数据类型

delimiter//

create proceduce pro\_books(in price1 float,price2 float)

begin

select \* from books where price between 20 and 200;

end//

--有参数，call存在过程的名字(传入的参数);

call pro\_books(20,200);

例3：

创建一个无参有返回值的存储过程，用于查询所有图书的平均价格

select avg(price) as 平均价格 from books;

--out 参数的名称 数据类型

delimiter//

create proceduce pro\_books(out priceAvg float)

begin

select avg(price) into priceAvg from books;

end//

call books1(@priceAvg);

select @priceAvg;

--查看所有的存储过程

show proceduce status;

--删除 (先判断是否存在再删除)

drop proceduce if exists pro\_books;

三.MySQL触发器

1.定义：当一件事情发生的时候会引发其他相关的事件发生

只有当一个预定义的事件发生的时候，就会被MySQL自动调用

2.语法：（创建）：create trigger 触发器名称{before|after}{inster|update|delete} on 表名

for each row[针对每一条都触发]

begin

触发器执行的语句块

end

注意：当语句结束;有冲突时使用delimiter定义新的结束方法

例1：

创建insert触发器，当在card表中新增加一个借阅用户的时候，显示“新增用户成功”

detimiter//

create trigger tri\_insert\_card after insert on card

for each now

begin

--into 只是把结果赋值给了result @后面可随便写名称

select “新增用户成功” into @result;

end//

select \* from card;

insert into card(name,class) values (“唐三”,”大数据1班”);

--查询 @result结果名称

select @result;

例2：

创建insert触发器，借阅人借阅一本书会在info表中产生一条记录，此时被借阅的书籍的总数量应该减1，触发在info表中

#获取添加的借阅记录的bno(图书号)

#更新数量信息

--update books set quanitity-1 where 哪本书;

--如果我们能够拿到新插入的那条借阅记录的bno

select \* from info;

detimiter//

create trigger tri\_insert\_info after insert on info

for each row

begin

select new.bno into @bnovalue;

--拿到插入的那条借阅记录的bno

update books set quanitity-1 where @bnovalue;

--进行数据的更新

end

查三个表：

insert into info(cno,bno,rdate) values (106,1006,now());

（显示所有的触发器）：show triggers;

（删除存储过程）：drop triggers if exists 触发器名称;

第八章 存储引擎及图形化管理工具

一、存储引擎种类

1.分类：MyISAM、InnoDB、MEMORY、Archive

①InnoDB存储引擎（8.0）

特点：

从MySQL5.5版本之后，MySQL的默认内置存储引擎已经是InnoDB了，他的主要特点有：

（1）支持事务。

（2）使用行级锁，可以支持更高的并发；

（3）支持外键；

②MyISAM存储引擎

特点：

在5.5版本之前，MyISAM是MySQL的默认存储引擎，该存储引擎并发性差，不支持事务，所以使用场景比较少，主要特点为：

（1）不支持事务；

（2）不支持外键，如果强行增加外键，不会提示错误，只是外键不起作用；

（3）默认表级锁，所以并发度很差，加锁快，锁冲突较少，所以不太容易发生死锁；

（4）数据库所在主机如果宕机，MyISAM的数据文件容易损坏，而且难恢复；

③MEMORY存储引擎

特点：将数据库存在内存中和市场上的Redis,memcached等思想类似，为了提高数据的访问速度；

主要特点：（1）支持的数据类型有限制。比如：不支持Text和BloB类型，对于字符串类型的数据，只支持固定长度的行，varchar会被自动存储为char类型；

1. 支持的锁粒度为表级锁；
2. 由于数据是存放在内存中，所以服务器重启之后，所有数据都会丢失；

④Archive存储引擎

特点：Archive引擎适合的场景有限，由于其支持压缩故主要用来做日志，流水等数据的归档；

主要特点：（1）支持zlib压缩，数据在插入表之前，会被先压缩

1. 仅支持select和insert操作，存入的数据就只能查询做修改和删除；
2. 只支持自增键上的索引，不支持其他索引；

2.select version(); #查看版本

show engines; #查看支持的引擎

3.mysql 8.0版本默认引擎是InnoDB；

mysql 5.5版本默认引擎是MyISAM；

4.（1）在mysql5.5版本中插入一个记录，插入一个没有的人和序号，则能运行出来，那么违反了外键约束，MyISAM的引擎不支持外键；

（2）在mysql 8.0版本中插入一条记录，插入一个没有的人和序号，则能运行出来(报错),这个InnoDB支持外键；

5.什么是事务？

事务对应生活中用户的操作就是一个业务。事务是用户操作数据库的基本单元，1个事务是由1到多条sql语句组成

（组成事务的sql命令的多少取决于业务的复杂程度）。组成事务的一组sql命令，要么一起成功，要么一起失败。

事务是为了保证业务操作的完整性和数据的安全一致性。

6.事务的特性：原子性（Atomic）、一致性（Consistency）、隔离性（Isolation）、持久性（Durability）

7.表级锁和行级锁

（1）表级锁：(依次操作，不能同时操作)

直接锁定整张表，在你锁定期间，其它进程无法对该表进行写操作。如果你是写锁，则其它进程则读也不允许

（2）行级锁：

某一条记录进行锁定，两个记录不冲突，可同时操作，若有冲突，仅对指定的记录进行加锁，这样其它进程还是可以对同一个表中的其它记录进行操作。

二、修改表存储引擎

语法：

show create table 表名; #查看当前表的存储引擎

1. 修改已创建过的表：alter table 表名 ENGINE = 修改的存储引擎;

（二）创建表时指定存储引擎

CREATE TABLE 表名（

列名1 数据类型 [完整性约束条件]，

列名2 数据类型 [完整性约束条件]，

……

列名n 数据类型 [完整性约束条件]，

）ENGINE=INNODB;

注：建议创表时直接指定存储引擎

（三）1、修改MySQL8.0版本的引擎

（1）alter table 表名 engine= myisam;

运行出来报错了不允许修改，因为有外键约束

（2）用创建表示指定存储引擎修改:

CREATE TABLE 表名（

列名1 数据类型 [完整性约束条件]，

列名2 数据类型 [完整性约束条件]，

……

列名n 数据类型 [完整性约束条件]，

）engine=myisam;

1. 修改MySQL5.5版本的引擎

（1）alter table 表名 engine= InnoDB;

运行成功，没有报错

1. 验证InnoDB引擎有没有用：

先插入一条数据，插入没有的人和序号

若插入成功，说明InnoDB引擎是没有用的

注：建议创表时直接指定存储引擎

3、创建完表后再修改引擎效果很差；要在创建外键表时，直接写一个存储引擎

1. 图形化管理工具

1、种类

（1）phpMyAdmin（http://www.phpmyadmin.net/）

（2）SQLyog（http://sqlyog.en.softonic.com/ ）

（3）Database Master

（4）Navicat

2、连接

（1）连接本地MySQL（正常连接）

（2）连接远程MySQL：点ssh 然后填写信息

3、生成E-R图模型