# ：初识MySql

## 一：数据库介绍

引入：

我们之前使用的数据都是存储在内存中的！比如说我们写一个注册功能。

我们首先需要在内存中创建一个对象，之后输入注册需要的用户名和密码等数据！

然后登陆时，输入注册的数据即可完成登陆！

注册成功之后 比如用户名是admin 密码是123456

出现的问题：

请问

01.用户的数据放在哪里了？

02.如果放在内存中，下次登陆还使用之前注册的用户名和密码，能登陆吗？

显然是不能的！ 因为程序一旦运行完毕，内存中的数据也随之消失！

怎么办？

01使用序列化保存对象到文件中

02.使用xml保存数据到文件中

但是把大量的数据放在文件中，我们对文件数据的读写效率是相对较慢的！

而且也不便于我们管理文件中的数据！这时候就需要有新的存储方式来代替！

就是我们所谓的数据库（存储数据的仓库）！

### 1.1：什么是数据库

定义：按照数据结构来组织,存储和管理数据的仓库,我们称之为数据库（DataBase,简称：DB）。

这些存储的数据可以是多种形式：如文字,符号,图片,视频,音频等！

从广义上来说，我们的txt文件,word文档,excel文件都可以理解成一个数据库！

在IT的世界里,数据库指的是由专业技术团队开发的用于存储数据的软件系统！

### 1.2：使用数据库的必要性

1.可以结构化的存储大量的数据信息，方便用户进行访问和操作

数据库可以对数据进行分类保存，提高我们的访问效率。

2.可以保证数据信息的一致性，完整性，降低数据的冗余

数据库中有事务（ACID特性）来保证数据信息的一致性，完整性;

数据库自身也有约束来降低数据的冗余。

数据冗余（Redundance）:数据重复的现象！

3.可以满足应用的安全和共享方面的要求

如果我们把员工的所有信息，都保存在一个txt文件中，那么员工的薪水就会暴露出来！很显然这是不允许的！

如果把数据放在数据库中，我们可以限制只有财务人员才能查询薪水！而其他员工只能查询自己的信息！

### 1.3：常见的关系型数据库

1.SQL Server 官网：<http://www.microsoft.com/zh-cn/sql-server/sql-server-2016>

是Microsoft公司的关系型数据库管理系统！

2.MySql 官网：<https://www.mysql.com/>

是一种开放源代码的关系型数据库管理系统！

3.Oracle 官网：<http://www.oracle.com/technetwork/database/index.html>

是Oracle公司的关系型数据库管理系统！

注意点：

虽然数据库不一样，但是对数据的操作都大同小异，有的操作命令都是一致的！

关系型数据库：是指建立在关系模型基础上的数据库，借助于集合代数等数学概念和方法来处理数据库中的数据！

### 1.4：关系型数据库和非关系型数据库

常见的非关系型数据库：

Mongo DB

Redis

HBase

二维表的介绍：

在[关系模型](https://baike.baidu.com/item/%E5%85%B3%E7%B3%BB%E6%A8%A1%E5%9E%8B" \t "https://baike.baidu.com/item/%E4%BA%8C%E7%BB%B4%E8%A1%A8/_blank)中，数据结构表示为一个二维表，一个关系就是一个二维表（但不是任意一个二维表都能表示一个关系），二维表名就是关系名。表中的第一行通常称为属性名，表中的每一个[元组](https://baike.baidu.com/item/%E5%85%83%E7%BB%84" \t "https://baike.baidu.com/item/%E4%BA%8C%E7%BB%B4%E8%A1%A8/_blank)和属性都是不可再分的，且元组的次序是无关紧要的。

常用的[关系术语](https://baike.baidu.com/item/%E5%85%B3%E7%B3%BB%E6%9C%AF%E8%AF%AD" \t "https://baike.baidu.com/item/%E4%BA%8C%E7%BB%B4%E8%A1%A8/_blank)如下：

记录 二维表中每一行称为一个记录，或称为一个元组。

字段 二维表中每一列称为一个字段，或称为一个属性。

域 即属性的[取值范围](https://baike.baidu.com/item/%E5%8F%96%E5%80%BC%E8%8C%83%E5%9B%B4" \t "https://baike.baidu.com/item/%E4%BA%8C%E7%BB%B4%E8%A1%A8/_blank)。

两者的优缺点：

关系型数据库：

01.容易理解，存放在数据库中的数据就是以二维表的形式存储的！

二维表就是有行和列组成的，知道行号和列号就能定位到表中的数据！Excel就是二维表！

02.使用方便，基本通用的sql语言使得操作关系型数据库方便！

03.支持sql，所以可以进行复杂的查询！

04.易于维护，丰富的完整性（域完整性，实体完整性，参照完整性和自定义完整性）大大降低了数据的冗余，事务的ACID特性保证了数据的一致性！

05.读写性能低

06.对海量数据的处理不如非关系型数据库

07.分布式不够完善

非关系型数据库：

01.不支持sql，省去了解析sql的步骤，读写性能高

02.基于键值对的形式，数据没有耦合性，扩展性强

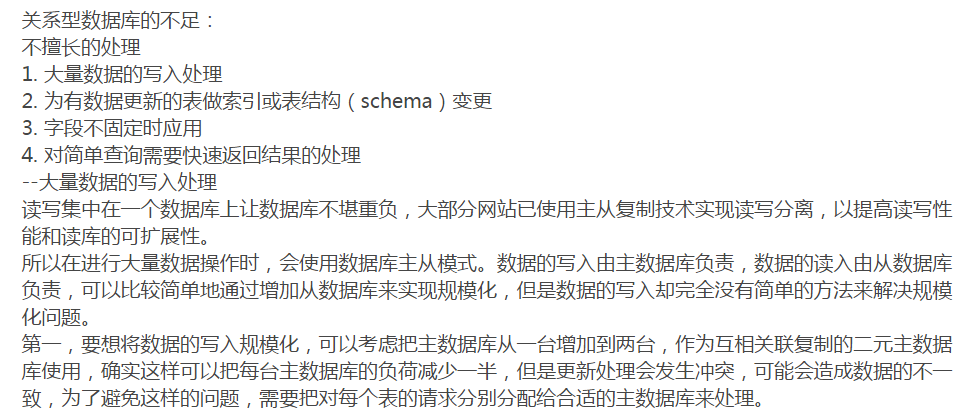
03.可以实现数据的分布式处理

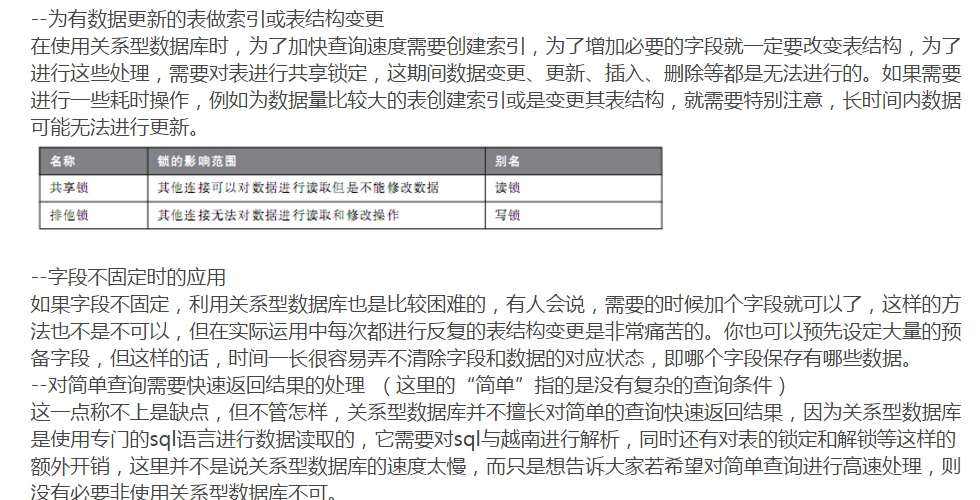
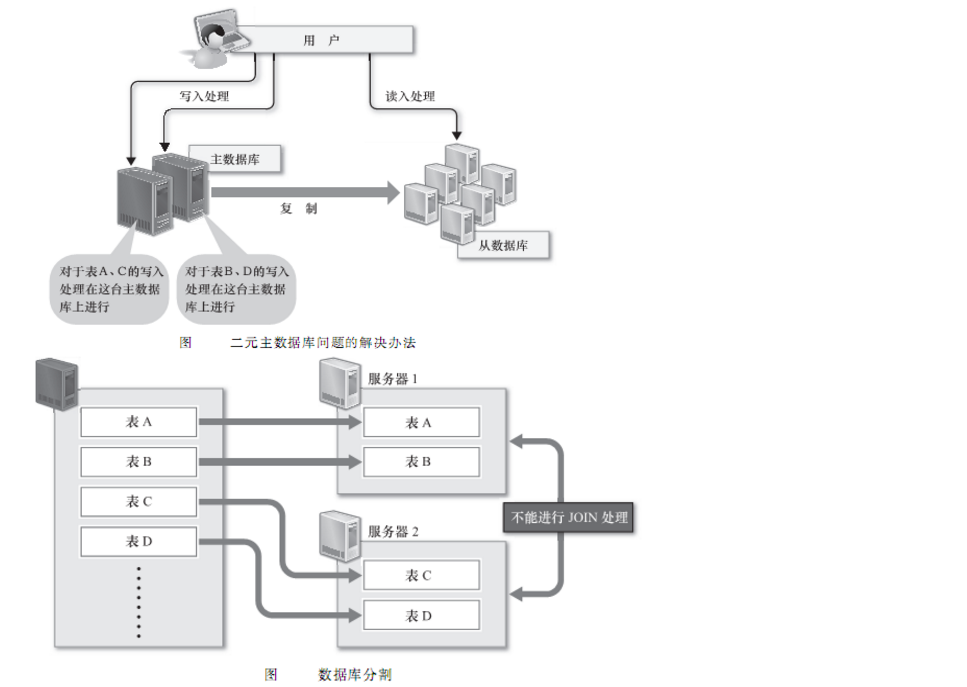
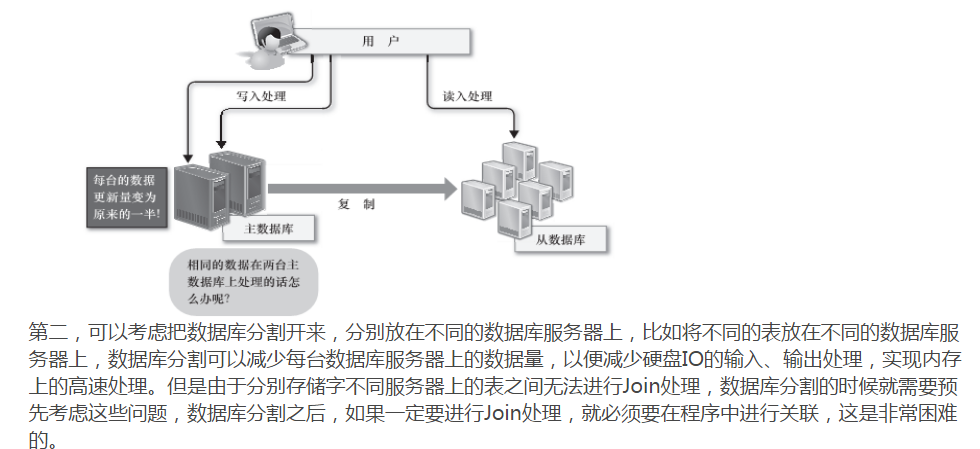
04.适用于海量数据的处理

05.持久化的性能低

06.因为没有事务，所以数据的安全性不高

07.出道时间短，功能没有关系型数据库完善





### 1.5：MySql数据库的特点

版本分类：

01.社区版：

自由下载免费！适用于普通用户！

02.企业版：

不能自有下载并且收费！官方提供了完整的技术支持！

适用于对数据库要求比较高的企业用户！

优势：

01.运行速度快，因为MySql体积小，命令执行速度快

02.使用成本低，因为MySql免费开源（相对于大多人而言）

03.易学易用，相对于其他数据库的设置和管理

04.可移植性强，能够运行在windows, linux ,unix中！

### 1.6：数据库管理系统和数据库系统

数据库管理系统（DataBase Management System,简称DBMS）:就是管理数据库的系统！

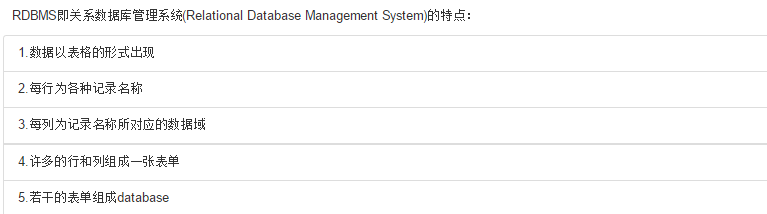
数据库系统（DataBase System,简称DBS）:包含数据库软件，数据库和数据库管理员！

数据库管理员（DataBase Administrator,简称DBA）:创建,监控和维护整个数据库的专业管理人员！

数据库( DataBase)

DBA通过DBMS操作DB！ 整体就是DBMS

关系型数据库管理系统（Relational DataBase Management System）



### 1.7：数据库中相关的名词

数据库：数据表的集合！

数据表：数据的集合！

行：一行数据（元组或者记录） 横向

列：一列数据（字段） 纵向

主键（Primary key）： 唯一标识数据的字段

外键(Foreign Key)：关联两个表之间关系的字段

标识列 是 自增列！

主键是 唯一！

复合主键：

  所谓的复合主键 就是指你表的主键含有一个以上的字段组成,不使用无业务含义的自增id作为主键。

比如

create table test

(

name varchar(19),

id number,

value varchar(10),

primary key (name,id)

)

上面的name和id字段组合起来就是你test表的复合主键 ,它的出现是因为你的name字段可能会出现重名，所以要加上ID字段这样就可以保证你记录的唯一性 ,一般情况下，主键的字段长度和字段数目要越少越好 。

这里就会有一个疑惑?  主键是唯一的索引，那么为何一个表可以创建多个主键呢？

其实“主键是唯一的索引”这话有点歧义的。举个例子，我们在表中创建了一个ID字段，自动增长，并设为主键，这个是没有问题的，因为“主键是唯一的索引”，ID自动增长保证了唯一性，所以可以。

此时，我们再创建一个字段name，类型为varchar，也设置为主键，你会发现，在表的多行中你是可以填写相同的name值的，这岂不是有违“主键是唯一的索引”这句话么？

所以我才说“主键是唯一的索引”是有歧义的。应该是“当表中只有一个主键时，它是唯一的索引；当表中有多个主键时，称为复合主键，复合主键联合保证唯一索引”。

为什么自增长ID已经可以作为唯一标识的主键，为啥还需要复合主键呢。因为，并不是所有的表都要有ID这个字段，比如，我们建一个学生表，没有唯一能标识学生的ID，怎么办呢，学生的名字、年龄、班级都可能重复，无法使用单个字段来唯一标识，这时，我们可以将多个字段设置为主键，形成复合主键，这多个字段联合标识唯一性，其中，某几个主键字段值出现重复是没有问题的，只要不是有多条记录的所有主键值完全一样，就不算重复。

### 1.8：数据完整性

数据完整性：

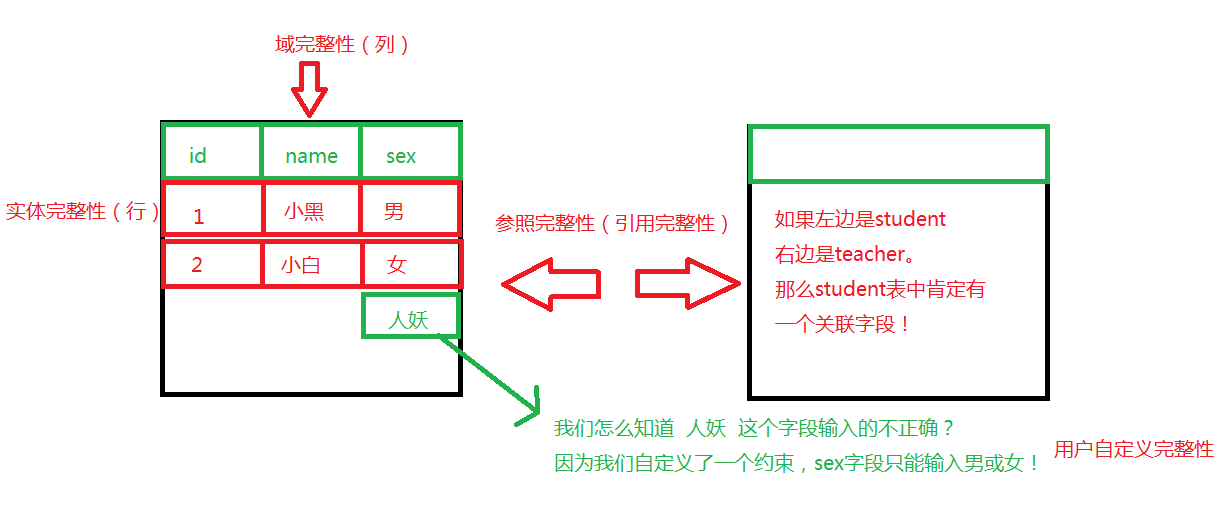
[存储](https://baike.baidu.com/item/%E5%AD%98%E5%82%A8" \t "https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%AE%8C%E6%95%B4%E6%80%A7/_blank)在数据库中的所有数据值均正确的状态。如果数据库中存储有不正确的数据值，则该数据库称为已丧失[数据](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE/5947370" \t "https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%AE%8C%E6%95%B4%E6%80%A7/_blank)完整性。

组成部分

1. 实体完整性（Entity Integrity）
2. 域完整性（Domain Integrity）

3.参照完整性（Referential Integrity）

4.用户自定义完整性（User Defined Integrity）



## 二：Mysql的配置

MySql默认的端口号是3306

### 2.1：默认字符集的设置

在mysql的安装目录，会看到my.ini文件！

my.ini文件介绍

01.default-character-set=utf8

设置客户端的字符集

02.character-set-server=utf8 设置服务器端的字符集

character-set-client=utf8 客户端发送的语句字符集

character-set-connection=utf8 设置服务器接收客户端语句的字符集

character-set-results=utf8 服务器将结果转换成对应字符集之后响应给客户端

同时设置字符集 client connection results

set names utf-8;

03.port

mysql的默认端口号是3306

04.basedir

数据库的安装位置

05.datadir

数据存放的位置

06.default-storage-engine

mysql的默认存储引擎

如果修改了配置文件，需要重启mysql服务才会生效！

Utf-8是一种针对unicode的可变长度字符编码，又称为万国码。它是一种用于解决国际上字符不统一的一种编码格式，对英文使用8位（1字节），中文24位（3字节）来编码。包含了世界上所有国家需要的字符，是国际编码，通用性强！

### 2.2：配置mysql环境变量

01.复制mysql安装位置的bin目录

02.右键我的电脑===》属性===》高级系统设置===》环境变量===》把刚才复制的bin路径放入path变量中即可！

## 三：命令行连接MySql

在windows操作系统中，命令不区分大小写！

### 3.1：启动mysql服务

01.win+R 输入 services.msc

02.找到mysql服务 右键点击开启

03.win+R 输入 cmd

04.进入命令窗口之后输入net start mysql 开启服务

05.net stop mysql 停止服务

### 3.2：连接mysql数据库

01.连接命令

Mysql -h 服务器主机地址 -u 用户名 -p密码

02.如果是本机操作，可以省略-h

03.-p之后也可以不写，密码！回车之后再输入密码

如果-p之后书写密码，则不需要有空格

Root用户连接mysql数据库

Mysql -u root -p密码

-p和密码之间不能有空格

### 3.3：查询mysql数据库的版本号以及当前登陆的用户

select version(),user();



### 3.4：MySql中数据库类型

1. 用户数据库

用户根据需求自己创建的数据库！是我们研究的重点！

2.系统数据库

安装完mysql之后，默认附带的数据库

01.information\_schema:存储了数据库对象的信息。如用户表信息，字段信息，权限信息，字符集信息和分区信息等。

02.performance\_schema:存储了数据库服务器性能参数信息。

03.mysql：存储了系统的用户权限信息

04.test：数据库管理系统自动创建的测试数据库，任何用户都可以使用

## 四：SQL简介

### 4.1：什么是SQL

SQL的全称是（Structred Query Language）结构化查询语句。

01.1974年被提出，1980年改名为SQL

02.关系型数据库的标准语言

03.数据库脚本文件的扩展名

### 4.2：SQL的组成

01.DDL（Data Definition Language）数据定义语言。

用来创建数据库中的各种对象。如表，视图，索引，同义词等。

create drop truncate alter

02.DQL（Data Query Language）数据查询语言。

select where group by having order by

03.DML（Data Manipulation Language）数据操作语言。

Insert update delete

04.DCL（Data Control Language）数据控制语言。

grant revoke rollback commit

### 4.3：SQL中的运算符

01.算术运算符

+ 加法

- 减法

\* 乘法

/ 除法

% 取余

02.赋值运算符

= 把=右边的值赋值给左边

03.比较运算符

> 大于

< 小于

<= 小于等于

>= 大于等于

<> 不等于

!= 不等于（不符合sql-92标准）

04.逻辑运算符

And 当两个表达式都为true,返回true

Or 当两个表达式有一个为true,返回true

Not 对表达式取反操作，优先级最高

## 五：使用DDL语句操作数据库

### 5.1：创建数据库

create database 数据库名称;

Query ok : 表示sql语句执行成功

1 row affected :表示影响的行数

0.00sec :表示执行时间

### 5.2：查询数据库列表

show databases;

### 5.3：选择数据库

use 数据库名称;

### 5.4：删除数据库

drop database 数据库名称;

### 5.6：创建用户

01.创建用户 只允许 本机 访问

CREATE USER 'xiaodoufu'@'localhost' IDENTIFIED BY 'xiaodoufu';

其他的机器也可以访问

CREATE USER 'xiaodoufu'@'%' IDENTIFIED BY 'xiaodoufu';

02.给用户授权

GRANT ALL PRIVILEGES ON \*.\* TO 'xiaodoufu'@'localhost';

03.让权限立即生效

FLUSH PRIVILEGES;

## 六：使用DDL语句操作数据表

### 6.1：数据类型

01. 数值类型

tinyint（M） 1字节

smallint（M） 2字节

mediumint（M） 3字节

int（M,D） 4字节

float （M,D） 4字节

double （M,D） 8 字节

Decimal （M,D） M+2字节

M:总位数，取值范围是1-65，默认值10.

D:小数位，取值范围是0-30，不能超过M的值！

Tinyint,smallint,mediumint,int都是整数类型！

Float,double 都是浮点类型！

Decimal 是精度最高的数值类型！

薪水存储一般都是用decimal来存储，因为精确性高！

如果定义了一个字段的类型是decimal(3,1)

那么这个字段的取值返回是 -99.9 到 99.9

02. 字符串类型

Char(M) M字节 固定长度的字符串，M的取值返回是0-255

如果插入的数据字节数不够，空格补全！

如果插入的数据字节数超过了M，自动截取多余的字节！

Varchar(M) 可变长度 M的取值返回是0-65535

如果插入的数据字节数不够不需要补全！

如果插入的数据字节数超过了M，自动截取多余的字节！

Tinttext: 取值返回是0-255！ 微型文本串

Text ：取值返回是0-65535！ 文本串

03. 日期类型

Date YYYY-MM-DD current\_date这个函数可以获取当前系统的年月日

Datetime YY-MM-DD hh:mm:ss

Timestamp YY-MM-DD hh:mm:ss CURRENT\_TIMESTAMP这个函数也是设置系统默认时间

Time hh:mm:ss

Year YYYY 得到年份

注意点：

001.如果一个字段是日期类型，那么任何符号都可以作为日期的分隔符

比如

2017-02-01

2017+02+01

2017/02/01

2017.02.01

等 都是可以正确给日期类型的字段赋值成功！

002.CURRENT\_DATE 这个函数获取当前系统的年月日

003. CURRENT\_TIMESTAMP 这个函数获取系统 年月日 时分秒

004.CURRENT\_TIME 这个函数获取系统 时分秒

005.有需求是创建一个字段来记录某条信息的最近修改时间，

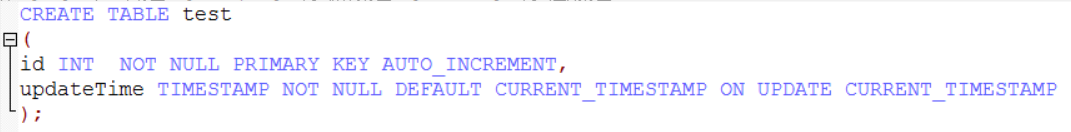
那么我们可以在创建表的时候，就定义为

字段名称 timestamp NOT NULL default CURRENT\_TIMESTAMP on update CURRENT\_TIMESTAMP

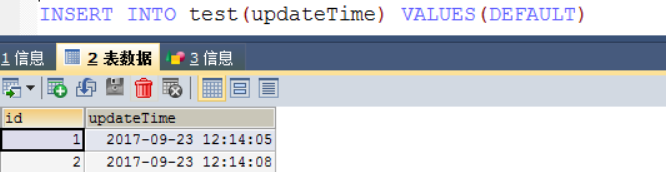
意思是说，只要有人修改这条信息的时候，这条信息的这个日期字段就会自动更新！

示例：

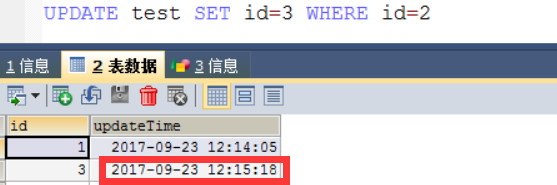
1. 创建表



2.新增两条数据



3.修改id为2的id为3



4.时间会自动更新

Enum类型

# 创建一个student表

CREATE TABLE IF NOT EXISTS student(

id INT(4) NOT NULL PRIMARY KEY AUTO\_INCREMENT COMMENT '学生编号',

`name` VARCHAR(10) NOT NULL UNIQUE KEY COMMENT '学生姓名',

sex ENUM('男','女') DEFAULT '男' COMMENT '学生性别'

)ENGINE='InnoDB' DEFAULT CHARSET='utf8' COMMENT '学生表';

### 6.2：创建表

1.语法

Create table [if not exists] 表名(

字段名称 数据类型 [字段属性|约束 ] [索引] [注释],

字段名称 数据类型 [字段属性|约束 ] [索引] [注释]

)[表类型][表字符集][注释];

2.注意点

01.如果创建的表名,字段等与数据库中的保留字冲突，建议使用 反引号(`)括起来！

Mysql自动生成的代码中,数据库，表名和字段都会使用 反引号(`)括起来！

02.DDL语句创建表的时候，多个字段之间使用逗号(,)隔开！最后一个字段无需逗号(,)

03. 单行注释 #

多行注释 /\*\*/

04.表中的字段我们也称之为 列！

3. 有符号类型 和 无符号类型

有符号类型 ：可以取负值

无符号类型：默认是0！ 0--类型的长度

ZEROFILL属性：如果位数不够，前面用零补齐！

若某数值字段指定了ZEROFILL属性，

将自动添加UNSIGNED属性！

# 创建一个student表  
CREATE  TABLE  IF NOT EXISTS student(  
 id INT(4) NOT NULL PRIMARY KEY AUTO\_INCREMENT COMMENT '学生编号',  
 `name` VARCHAR(10) NOT NULL UNIQUE KEY COMMENT '学生姓名',  
 sex  VARCHAR(2) NOT NULL DEFAULT '男' COMMENT '学生性别'  
)ENGINE='InnoDB' DEFAULT CHARSET='utf8' COMMENT '学生表';  
  
# 新增字段  
INSERT  INTO student(id,`name`,sex) VALUES(10,'小白',DEFAULT);  
INSERT  INTO student(`name`,sex) VALUES('小黑',DEFAULT);  
INSERT  INTO student(`name`,sex) VALUES('小黑2',DEFAULT);  
#  如果我们手动的更改了自动增长的列，那么下次自动增长的起点就变了  
  
#  01.同时增加多条数据  
  
INSERT  INTO student(`name`,sex) VALUES('小黑1',DEFAULT);  
INSERT  INTO student(`name`,sex) VALUES('小黑2',DEFAULT);  
INSERT  INTO student(`name`,sex) VALUES('小黑3',DEFAULT);  
INSERT  INTO student(`name`,sex) VALUES('小黑4',DEFAULT);  
  
#  02.同时增加多条数据  
INSERT  INTO student(`name`,sex) VALUES('小黑1',DEFAULT),('小黑2',DEFAULT),('小黑3',DEFAULT),('小黑4',DEFAULT);  
  
  
# 查询当前数据库中所有的表  
SHOW TABLES;  
  
# 查询表的定义  
 DESC student;  
 DESCRIBE student;

### 6.3：字段的约束

为了保证数据的完整性，mysql中提供了约束机制！

常用的属性约束：

1. 非空约束 not null

字段不允许为空！

2.默认约束 default

给字段设置默认值

3.唯一约束 unique key(UK)

设置字段的值是唯一。的允许有空值，但只能有一个！

1. 主键约束 primary key(PK)

设置表中记录的唯一标识！一个表中不允许重复！

1. 外键约束 foreign key(FK)

用于两个表之间建立关联关系！

注意点：

01.InnoDB引擎支持外键，MYISAM引擎不支持！

02.外键只能建立在从表中！

1. 自动增长 auto\_increment

01.一般用于主键，默认自增1

02.是整数类型

03.可设置初始值和步长

### 6.4：表和字段的注释

Create table 表名(

字段1 类型 comment 注释内容,

字段2 类型 comment 注释内容

)comment=’表的注释内容’

### 6.5：表的引擎

Create table 表名(

字段1 类型 comment 注释内容,

字段2 类型 comment 注释内容

)engine=’表的引擎’

### 6.6：表的编码

Create table 表名(

字段1 类型 comment 注释内容,

字段2 类型 comment 注释内容

)charset=’表的编码格式’

### 6.7：查询表

01.查询数据库的所有表

Show tables;

注意点： show tables之前必须先选择数据库！

02.查询表的定义

describe 表名; 或者 desc 表名;

### 6.8：修改表

01.修改表名

Alter table 旧表名 rename [to] 新表名;

02.给表中添加字段

Alter table 表名 add 字段名 数据类型[属性];

03.修改表中的字段

Alter table 表名 change 原字段名 新字段名 数据类型[属性];

04.删除字段

Alter table 表名 drop 字段名;

05.添加主键约束

Alter table 表名 add constraint 主键名 primary key 表名（字段名）;

复合主键

将多列设置成主键！

-- 给成绩表设置复合主键

ALTER TABLE result ADD PRIMARY KEY pk\_result (studentno,subjectno,examdate);

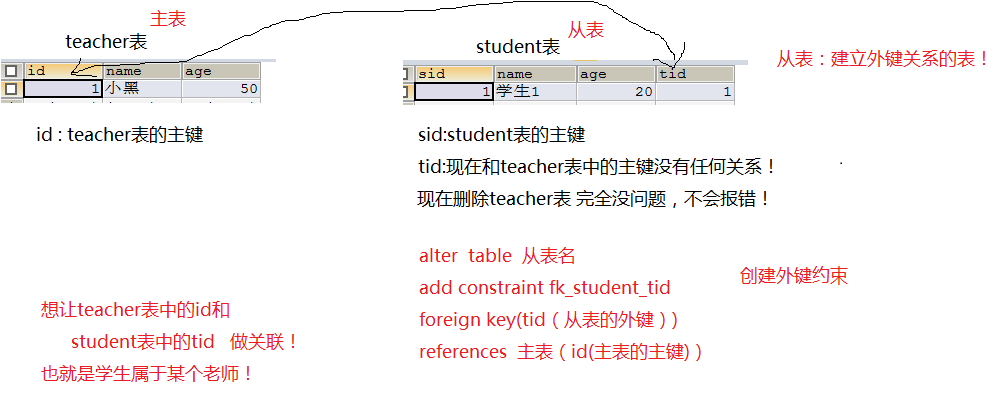
06.添加外键约束

Alter table 表名 add constraint 外键名 foreign key （外键字段名） references 关联表名(关联字段);

如果 现在有 两个表 ，比如说 一个是 teacher 一个是 Student

两表存在主外键关系！那么 不能直接删除teacher表！哪怕删除teacher表中的数据

也要考虑student表中是否有和老师对应的数据！



### 6.9：删除表

Drop table [if exists]表名;

## 七：MySql系统帮助

Help 查询内容;

# ：DML和DQL

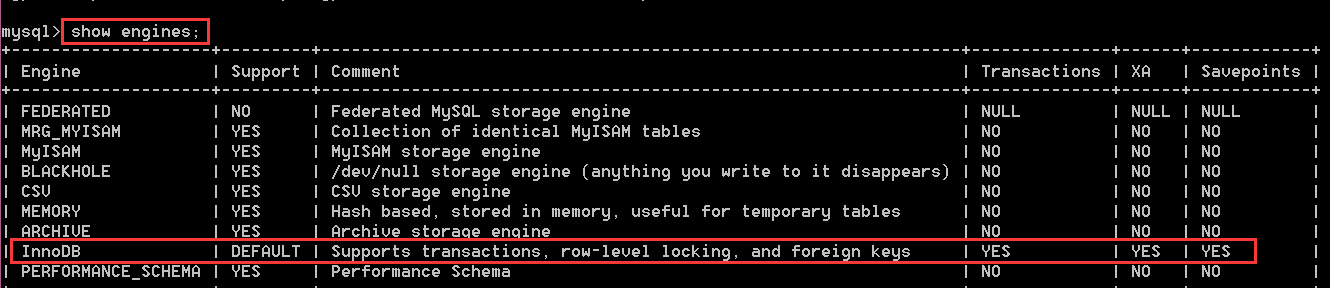
## 一：MySql的存储引擎

问题的引入：

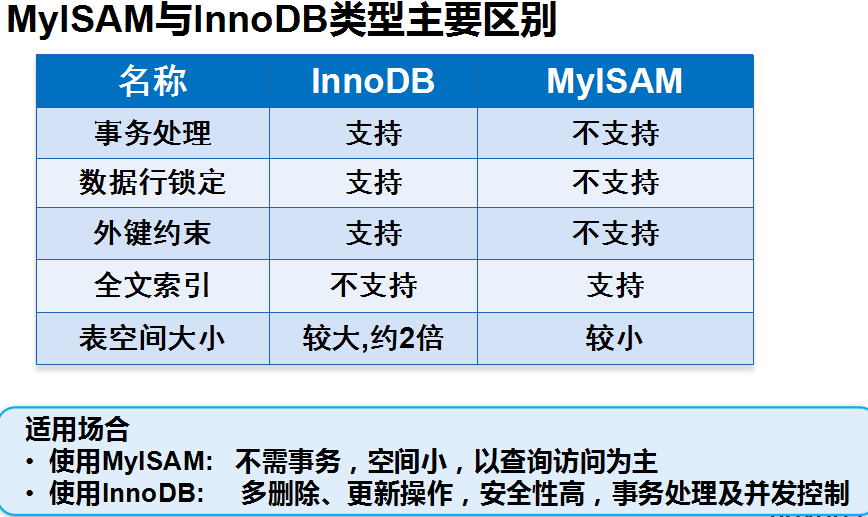
由于不同用户对数据的容量，访问速度，数据安全性有着不同的要求。

为了满足不同用户的需求，mysql数据库采用多种存储引擎来进行数据的存储！

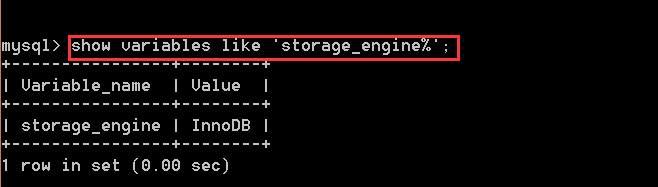
### 1.1：查询mysql数据库中所有的存储引擎



### 1.2：mysql数据库常用的存储引擎



### 1.3：查询默认的存储引擎



### 1.4：修改默认的存储引擎

找到安装目录的my.ini 文件 修改

default-storage-engine=引擎名称

之后重启mysql服务生效

### 1.5：mysql的数据文件

01.数据文件的存放位置

mysql安装目录下的data文件夹中

02.不同引擎的文件类型

InnoDB类型的文件

\*.frm ：表结构定义文件,存放表的元数据，与存储引擎无关！MyISAM也有！

\*.ibd : 数据文件，存放表中的数据！

MyISAM类型的文件

\*.frm ：表结构定义文件

\*.MYD ：数据文件

\*.MYI ：索引文件

## 二：使用DML插入数据

 新增数据  
INSERT  INTO student(id,`name`,sex) VALUES(10,'小白',DEFAULT);  
INSERT  INTO student(`name`,sex) VALUES('小黑',DEFAULT);  
INSERT  INTO student(`name`,sex) VALUES('小黑2',DEFAULT);

### 2.1：插入一条数据

INSERT  INTO student(`name`,sex) VALUES('小黑1',DEFAULT);  
INSERT  INTO student(`name`,sex) VALUES('小黑2',DEFAULT);  
INSERT  INTO student(`name`,sex) VALUES('小黑3',DEFAULT);  
INSERT  INTO student(`name`,sex) VALUES('小黑4',DEFAULT);

### 2.2：**同时增加多条数据**

INSERT  INTO student(`name`,sex) VALUES('小黑1',DEFAULT),('小黑2',DEFAULT),('小黑3',DEFAULT),('小黑4',DEFAULT);

注意点：

1.如果我们手动的更改了自动增长的列，那么下次自动增长的起点就变了  
 2.在使用insert语句的时候，如果没有书写表中的字段，那么values中的值必须和表中定义的字段顺序一致

3.建议在新增数据的时候 书写 字段名称

### 2.3：将查询结果插入到新表中

Create table 新表名(select 字段1,字段2.... from 旧表);

在执行查询语句的同时 会创建一张新表，字段就是查询出的字段！

前提： 新表不存在！ 若存在会报错！

## 三：使用DML更新数据

Update 表名 set 字段1=更新值,字段2=更新值 [where 条件];

注意点：

01.set之后可以跟多个字段，多个字段之间使用逗号隔开

02.where后面是更新那些数据的条件，如果不加条件，会对整个表的数据进行修改

## 四：使用DML删除数据

Delete from 表名 [where 条件];

Truncate table 表名;

delete 和 truncate的区别

01.delete

begin （开启事务）

select \* from grade;(查询年级表中所有的数据)

delete from grade; (删除年级表中所有的数据)

select \* from grade;(查询年级表中所有的数据，没有数据)

rollback; （事务回滚）

select \* from grade;(查询年级表中所有的数据,删除的数据恢复)

commit （提交事务)

02.truncate

begin （开启事务）

select \* from grade;(查询年级表中所有的数据)

truncate table grade; (删除年级表中所有的数据)

select \* from grade;(查询年级表中所有的数据，没有数据)

rollback; （事务回滚）

select \* from grade;(查询年级表中所有的数据，没有数据)

commit （提交事务)

区别：

01.delete后面可以拼接where条件，删除指定的行！

truncate只能删除表中所有的数据！不能有where！

02.delete可以回滚，数据库可以恢复！

truncate 不能事务混滚，数据不可以恢复！

03.truncate执行效率高！

04.都是删除数据，表的结构，索引，约束等属性不会删除！

## 五：DQL语句

 sql素材

语法：

Select <列名|表达式|函数|常量>

From 表名

[Where 条件]

[group by 分组依据]

[having 分组条件]

[order by 排序（desc|asc）]

[limit 分页数据]

Sql语句的执行顺序

01.from

02.where

03.group by

04.having

05.select

06.order by

07.limit

### 5.1：查询所有的数据行和列

Select \* from 表名; \*代表所有的列,性能低

### 5.2：查询部分列

Select 字段1,字段2... from 表名;

### 5.3：查询中使用别名

SELECT gradeID AS 年级编号,gradeName '年级 名称' FROM grade;

格式

01. 列名 AS 别名

02. 列名 别名

03. 如果别名中有特殊符号，必须把 别名用 单引号 引起来！

### 5.4：查询空值

使用is null 的时候 要确保 查询的列 可以为空！

null:

01.标识 空值

02.不是0，也不是空串""

03.只能出现在定义 允许为null的字段

04.只能使用is null 或者is not null 进行比较！

SELECT \* FROM student

WHERE loginPwd IS NULL

### 5.5：查询中使用常量列

如果需要在查询的时候，便于我们的统计和计算！可以使用常量列

Select name as ‘学生姓名’, ‘都是好学生’ as ‘备注信息’ from student;

模糊查询 % \_ 的区别

### 5.6：常用的聚合函数

count() 查询某个字段的行数

max()查询某个字段的最大值

min()查询某个字段的最小值

sum()查询某个字段的和

avg()查询某个字段的平均值

-- 查询成绩表的总成绩

SELECT SUM(studentResult) FROM result;

-- 查询成绩的平均值

SELECT AVG(studentResult) FROM result;

-- 查询成绩的最高分

SELECT MAX(studentResult) FROM result;

-- 查询成绩的最低分

SELECT MIN(studentResult) FROM result;

-- 查询有成绩的总数

SELECT COUNT(studentResult) FROM result;

### 5.7：常用的字符串函数

1. concat(...) 连接字符串

SELECT CONCAT('a','d','m','i','n') AS '拼接的结果';

2.insert(str,begin,length,newStr) 替换字符串

str:原始的字符串

begin：从原始字符串的哪个位置开始 在数据库中下标从1开始

length：需要替换的长度

newStr：替换的字符串

SELECT INSERT('admin',2,2,'ha');

3.substring(str,begin,length) 截取字符串

str:需要截取字符串

begin：开始截取的位置

length：截取的长度

SELECT SUBSTRING('admin',2,3);

4.lower(str) 转换成小写

SELECT LOWER('ADMIN');

5.upper(str) 转换成大写

SELECT UPPER('admin');

### 5.8：常用的日期时间函数

01.获取年月日

SELECT CURDATE();

02.获取时分秒

SELECT CURTIME();

03.获取年月日 时分秒

SELECT NOW();

04.获取年

SELECT YEAR(CURDATE());

SELECT YEAR(NOW());

05.获取小时

SELECT HOUR(NOW());

SELECT HOUR(CURTIME());

06.获取分钟

SELECT MINUTE(NOW());

SELECT MINUTE(CURTIME());

07.获取当前日期是本年的第几周

SELECT WEEK(NOW());

SELECT WEEK(CURDATE());

08.获取两个日期之间的天数 第一个参数 减去 第二个参数

SELECT DATEDIFF(NOW(),'2017-09-01');

SELECT DATEDIFF('2017-09-01',NOW());

09.获取给定日期之后的日期 第一个参数必须是日期格式

SELECT ADDDATE(NOW(),-34);

### 5.9：常用的数学函数

01.天花板函数

SELECT CEIL(3.0) FROM DUAL;

SELECT CEIL(3.1) FROM DUAL;

SELECT FLOOR(3.9) FROM DUAL;

02. 返回0-1之间的随机数

SELECT RAND() FROM DUAL;

03.四舍五入

SELECT ROUND(2.4) FROM DUAL;

### 5.10：dual伪表

dual 我们称之为 伪表！

在mysql中是一个摆设

select 9\*9;

select 9\*9 from dual;

select \* from dual; 报错

oracle中 必须使用 from dual;

select 9\*9 from dual; 正确的

select 9\*9 ; 错误

dual是一个只有一行一列的表！

只能查询！ 不能对 dual进行增删改！

### 5.11：和并列 分组

-- 面试题 和并列

-- 01.创建一个表 test 有字段（name,subject,sorce）

CREATE TABLE IF NOT EXISTS `exam`(

`name` VARCHAR(20) NOT NULL,

`subject` VARCHAR(20) NOT NULL,

sorce DOUBLE NOT NULL

)ENGINE=INNODB DEFAULT CHARSET=utf8;

-- 02.一次性插入5条测试数据

INSERT INTO exam(`name`,`subject`,sorce)

VALUES

('小黑1','java',80),

('小黑1','c#',88),

('小黑2','c#',90),

('小黑2','java',99),

('小黑3','java',100,

('小黑3','c#',100);

-- 03.根据name进行分组显示 成绩和科目名称（显示在一列）

SELECT `name` AS '姓名',GROUP\_CONCAT(`subject`,':',sorce) AS '合并列'

FROM exam

GROUP BY `name`;

### 5.12：order by , limit

# order by 和 limit

01.查询所有年级编号为1的学员信息

SELECT \* FROM student WHERE gradeid=1

02.查询所有年级编号为1的学员信息 并按照学号进行 降序排列

SELECT \* FROM student

WHERE gradeid=1

ORDER BY studentNo DESC # 默认是 asc　升序排列

03.查询所有年级编号为1的学员信息 并按照出生日期进行升序排列,学号进行降序排列

SELECT \* FROM student

WHERE gradeid=1

ORDER BY BornDate ASC ,studentNo DESC;

04.在03例子的基础上，查询年龄最大的5名同学

SELECT \* FROM student

WHERE gradeid=1

ORDER BY BornDate ASC ,studentNo DESC

LIMIT 0,5 # 0代表的是从第几条数据开始查询 5 显示几条数据

注意点：如果对多列进行排序时，每列之间使用逗号隔开，然后在列后面写上排序的依据！

## 六：子查询

-- 子查询

-- 把一个查询的结果 当成另一个查询的 字段，条件或者表！

SELECT studentName FROM student

-- 只能通过student 表 查询出 学生对应的 年级名称

-- 01. 先查询出 学生 武松 对应的 年级编号

SELECT GradeID FROM student WHERE studentName='武松'

-- 02.根据年级编号 取 年级名称

SELECT gradeName FROM grade WHERE GradeID=???

SELECT gradeName FROM grade WHERE GradeID

=(SELECT GradeID FROM student WHERE studentName='武松')

子查询注意事项

01.任何允许使用表达式的地方都可以使用子查询

02.嵌套在父查询SELECT语句的子查询可包括

SELECT子句

FROM子句

WHERE子句

GROUP BY子句

HAVING子句

select （子查询） from 表名

子查询的结果必须是单行单列！

03.只出现在子查询中而没有出现在父查询中的列不能包含在输出列中

04.出现在父查询中的where之后是作为一个条件存在的

05.出现在父查询中的from之后是作为临时表存在

select \* from (子查询)

子查询在作为临时表存在的时候，必须赋予别名

SELECT \* FROM (SELECT \* FROM student WHERE gradeid=1) AS temp

## 七：IN和NOT IN子查询

-- 查询年级编号是1或者2 的 所有学生列表

SELECT \* FROM student WHERE gradeId IN(1,2)

-- 查询 年级名称是 大一或者大二的所有学生信息

-- 学生表 中没有 年级名称 但是有年级编号

-- 01.根据 年级名称 查询出 编号

SELECT gradeID FROM grade WHERE gradeName IN('大一','大二');

-- 02.再根据id查询学生信息

SELECT \* FROM student WHERE

gradeID

IN (SELECT gradeID FROM grade WHERE gradeName IN('大一','大二'))

-- 查询参加 最近一次 高等数学-1 考试成绩的学生的最高分和最低分

-- 01. 发现成绩表中 没有 科目名称 只有编号！根据名称取编号

SELECT SubjectNo FROM `subject` WHERE

subjectName='高等数学-1'

-- 02.查询最近一次 高等数学-1 考试的时间

SELECT MAX(ExamDate) FROM result

WHERE

SubjectNo=(SELECT SubjectNo FROM `subject` WHERE

subjectName='高等数学-1')

-- 所有最近考试的成绩

SELECT \* FROM result

WHERE ExamDate='2013-11-11 16:00:00'

-- 03.开始获取最高分和 最低分

SELECT MAX(studentResult) AS 最高分,

MIN(studentResult) AS 最低分

FROM result

WHERE SubjectNo=(SELECT SubjectNo FROM `subject` WHERE

subjectName='高等数学-1')

AND ExamDate=

(SELECT MAX(ExamDate) FROM result

WHERE

SubjectNo=(SELECT SubjectNo FROM `subject` WHERE

subjectName='高等数学-1'))

SELECT

MAX(studentresult) '最高分',

(SELECT studentName FROM student s,result r

WHERE s.studentNo=r.studentNo

AND examdate='2017-11-11 16:00:00'

AND subjectNo=1

ORDER BY StudentResult DESC

LIMIT 0,1) AS '最高分姓名',

MIN(studentresult) '最低分',

(SELECT studentName FROM student s,result r

WHERE s.studentNo=r.studentNo

AND examdate='2017-11-11 16:00:00'

AND subjectNo=1

ORDER BY StudentResult

LIMIT 0,1) AS '最低分姓名'

FROM result r,student s WHERE subjectNo=1

AND examdate='2017-11-11 16:00:00'

AND s.studentNo=r.studentNo

-- 查询 高等数学-1 考试成绩是 60 分的 学生信息

-- 01.根据 科目名称 获取 科目编号

SELECT SubjectNo FROM `subject`

WHERE SubjectName='高等数学-1'

-- 02.根据编号 查询 所有的学生编号

SELECT studentNo FROM result

WHERE SubjectNo=(SELECT SubjectNo FROM `subject`

WHERE SubjectName='高等数学-1')

AND StudentResult=60; -- 成绩=60

-- 03.查询学生信息

SELECT \* FROM student

WHERE studentNo IN

(SELECT studentNo FROM result

WHERE SubjectNo=(SELECT SubjectNo FROM `subject`

WHERE SubjectName='高等数学-1')

AND StudentResult=60)

-- 使用in替换 等于（=）的子查询语句！

-- in后面的子查询可以返回多条记录！

-- not in :不在某个范围之内

-- 查询未参加 “高等数学-1” 课程最近一次考试的在读学生名单

-- 01.根据 科目名称 获取 科目编号

SELECT SubjectNo FROM `subject`

WHERE SubjectName='高等数学-1'

-- 02.获取最近一次考试时间

SELECT MAX(ExamDate) FROM result

WHERE SubjectNo=

(SELECT SubjectNo FROM `subject`

WHERE SubjectName='高等数学-1')

-- 03.查询没参加的学生编号

SELECT studentNo,StudentName FROM student

WHERE studentNo NOT IN

(

SELECT StudentNo FROM result

WHERE SubjectNo=

(SELECT SubjectNo FROM `subject`

WHERE SubjectName='高等数学-1')

AND ExamDate=

(SELECT MAX(ExamDate) FROM result

WHERE SubjectNo=

(SELECT SubjectNo FROM `subject`

WHERE SubjectName='高等数学-1'))

)

# ：高级查询

## 一：Exists和Not Exists子查询

### 1.1：Exists子查询

Exists的特点

1.在执行create或drop语句前，可以使用exists语句来判断数据库对象是否存在，返回值是true或false

比如

drop table if exists student; 意思是如果存在表student则删除！否则不删除！

Create table if not exists student; 意思是如果不存在表student则创建，否则不创建！

2.exists还可以作为where条件的子查询

Select ..... from 表名 where exists （子查询）;

意思是：

如果子查询有结果，则返回值为true，继续执行外层的查询语句;

如果子查询没有结果，则返回值是false,外层的查询语句不会执行。

-- 检查“高等数学-1” 课程最近一次考试成绩

-- 如果有 80分以上的成绩，显示分数排在前5名的学员学号和分数

-- 不使用exists

-- 01.查询“高等数学-1” 课程 对应的编号

SELECT subjectNo FROM `subject`

WHERE SubjectName='高等数学-1'

-- 02.查询最近的考试成绩

SELECT MAX(ExamDate) FROM result

WHERE SubjectNo=(SELECT subjectNo FROM `subject`

WHERE SubjectName='高等数学-1')

-- 03. 在02的基础上 加条件 成绩大于80

SELECT \* FROM result

WHERE ExamDate=

(SELECT MAX(ExamDate) FROM result

WHERE SubjectNo=(SELECT subjectNo FROM `subject`

WHERE SubjectName='高等数学-1'))

AND StudentResult>80

-- 04.优化

SELECT studentNo,StudentResult FROM result

WHERE ExamDate=

(SELECT MAX(ExamDate) FROM result

WHERE SubjectNo=(SELECT subjectNo FROM `subject`

WHERE SubjectName='高等数学-1'))

AND StudentResult>80

ORDER BY StudentResult DESC

LIMIT 0,5

-- 使用exists

-- 检查“高等数学-1” 课程最近一次考试成绩

-- 如果有 80分以上的成绩，显示分数排在前5名的学员学号和分数

-- 01.查询“高等数学-1” 课程 对应的编号

SELECT subjectNo FROM `subject`

WHERE SubjectName='高等数学-1'

-- 02.查询最近的考试成绩

SELECT MAX(ExamDate) FROM result

WHERE SubjectNo=(SELECT subjectNo FROM `subject`

WHERE SubjectName='高等数学-1')

-- 03.查询学号和成绩

SELECT StudentNo,StudentResult FROM result

WHERE EXISTS

(

SELECT \* FROM result

WHERE subjectNo=(

SELECT subjectNo FROM `subject`

WHERE SubjectName='高等数学-1'

)

AND ExamDate=(

SELECT MAX(ExamDate) FROM result

WHERE SubjectNo=(SELECT subjectNo FROM `subject`

WHERE SubjectName='高等数学-1')

)

AND StudentResult>80

)

AND subjectNo=(

SELECT subjectNo FROM `subject`

WHERE SubjectName='高等数学-1'

)

AND ExamDate=(

SELECT MAX(ExamDate) FROM result

WHERE SubjectNo=(SELECT subjectNo FROM `subject`

WHERE SubjectName='高等数学-1')

)

ORDER BY StudentResult DESC

LIMIT 0,5

### 1.2：Not Exists子查询

-- 检查“高等数学-1”课程最近一次考试成绩

-- 如果全部未通过考试（60分及格），认为本次考试偏难，计算的该次考试平均分加5分

-- 01.查询“高等数学-1” 课程 对应的编号

SELECT subjectNo FROM `subject`

WHERE SubjectName='高等数学-1'

-- 02.查询最近的考试成绩

SELECT MAX(ExamDate) FROM result

WHERE SubjectNo=(SELECT subjectNo FROM `subject`

WHERE SubjectName='高等数学-1')

-- 03.查询成绩大于60的 反着来

SELECT StudentResult FROM result

WHERE StudentResult>60

AND SubjectNo=(

SELECT subjectNo FROM `subject`

WHERE SubjectName='高等数学-1'

)

AND ExamDate=(

SELECT MAX(ExamDate) FROM result

WHERE SubjectNo=(SELECT subjectNo FROM `subject`

WHERE SubjectName='高等数学-1')

)

-- 04. 如果全部未通过考试，考试平均分加5分

SELECT AVG(StudentResult)+5 FROM result

WHERE NOT EXISTS

(

SELECT StudentResult FROM result

WHERE StudentResult>60

AND SubjectNo=(

SELECT subjectNo FROM `subject`

WHERE SubjectName='高等数学-1'

)

AND ExamDate=(

SELECT MAX(ExamDate) FROM result

WHERE SubjectNo=(SELECT subjectNo FROM `subject`

WHERE SubjectName='高等数学-1')

)

)

AND SubjectNo=(

SELECT subjectNo FROM `subject`

WHERE SubjectName='高等数学-1'

)

AND ExamDate=(

SELECT MAX(ExamDate) FROM result

WHERE SubjectNo=(SELECT subjectNo FROM `subject`

WHERE SubjectName='高等数学-1')

)

-- 如果有 年级名称是大二 的学生，就 查询出 年级名称是大一的 所有学生信息

-- 01.先查询出 对应的年级编号

SELECT GradeId FROM grade WHERE GradeName='大一'

SELECT GradeId FROM grade WHERE GradeName='大二'

-- 02.在学生表中是否存在 年级名称是大二 的学生

SELECT \* FROM student WHERE gradeID=(

SELECT GradeId FROM grade WHERE GradeName='大二'

)

-- 03.如果有查询出 年级名称是大一的 所有学生信息

SELECT \* FROM student

WHERE EXISTS

(

SELECT \* FROM student WHERE gradeID=(

SELECT GradeId FROM grade WHERE GradeName='大二'

)

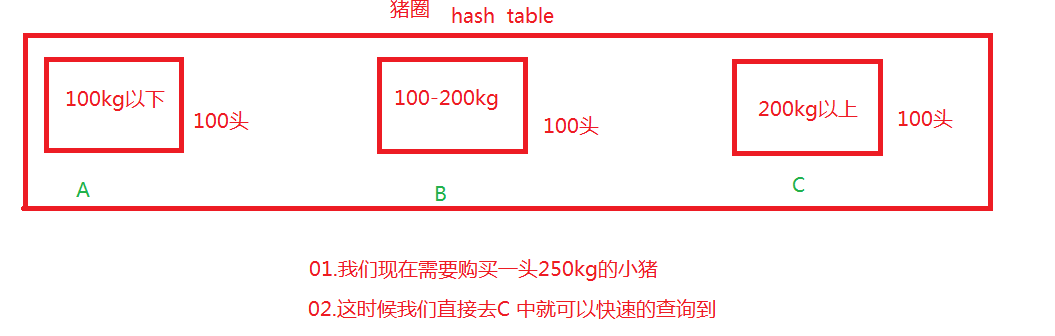
)

AND GradeId=(

SELECT GradeId FROM grade WHERE GradeName='大一'

)

### 1.3：Exists与IN , Not Exists与Not IN 的区别



01. IN和Not IN 做的是一个区间的判断，查询数据是否在区间内

Exists和Not Exists都是根据查询语句返回true或者false!

02. 例子：

select a.\* from A a

where a.id in(select id from B)

如果 A表中有1000条数据

B表中有1000条数据

分析步骤：

01. select id from B 会查询出B表中的所有id，然后缓存起来，共1000条（因为使用in 会先执行子查询）

02.然后分别拿A表中的每一个id和B表中的1000个id进行比较，也就是比较了 1000\*1000次

03.这样效率是非常慢的

04.如果B表中只有100或者10条数据（只是举例说明数据量小），那么就会比较1000\*10

这样相对来说效率会高点！

结论：

子查询中涉及的表（B）数据量小于 主查询中涉及的表（A）数据量 时，使用In来查询！可以提高效率

In 查询做的是 外表和内表的hash连接

hash连接就是 以 外层查询的表作为hash table ，内层查询的表在hash table中查询数据！

很显然，如果内层查询的数量大 ，查询效率就慢，查询数据量小，效率就高！

03. 例子：

select a.\* from A a

where exists (select id from B b where a.id=b.id)

如果A表中有1000条数据

B表中有1000条数据

分析步骤：

01. 使用了exists（会以外层查询为驱动）上面的sql语句只会执行1000次（因为A表中有多少条数据，就会执行几次）

02. exists查询不需要数据的结果集，只需要返回true或者false

结论：

子查询中涉及的表（B）数据量大于 主查询中涉及的表（A）数据量时，使用exists来查询！可以提高效率

Exists查询做的是loop循环

04. 如果子查询中涉及的表（B）数据量 和 主查询中涉及的表（A）数据量 差不多时，建议使用IN来查询！

因为In查询是在内存中的查询，exists需要查询数据库，所以内存中的查询肯定比查询数据库性能高！

05.not exists 在任何时候都比not in 效率高！

因为not in 那么内外表都进行[全表扫描](https://baike.baidu.com/item/%E5%85%A8%E8%A1%A8%E6%89%AB%E6%8F%8F" \t "https://baike.baidu.com/item/Exists/_blank)，没有用到索引！而not exists的子查询仍然可以用到索引！

## 二：any,some,all的使用

SELECT \* FROM student

WHERE studentname

IN(SELECT studentName FROM student)

-- 使用any（只要是在区间就行） 替换in

SELECT \* FROM student

WHERE studentname=ANY(SELECT studentName FROM student)

-- all 满足子查询中编号最大的

SELECT \* FROM student WHERE

studentNo>ALL

(SELECT studentNo FROM student WHERE studentNo IN(1014,1001,1002))

-- any 满足子查询中编号最小的

SELECT \* FROM student WHERE

studentNo>ANY

(SELECT studentNo FROM student WHERE studentNo IN(1014,1001,1002))

-- 和 any 效果一致

SELECT \* FROM student WHERE

studentNo>SOME

(SELECT studentNo FROM student WHERE studentNo IN(1014,1001,1002))

## 三：分组查询

-- 分组 group by

-- 01. 查询 每门课程的名称 以及平均分

-- 并且按照平均分降序排列

SELECT subjectName,AVG(StudentResult)

FROM `result` r,`subject` s

WHERE

r.`SubjectNo`=s.`SubjectNo`

GROUP BY subjectName

ORDER BY AVG(StudentResult) DESC

-- 02.在上述案例中增加 条件 having

-- 平均分>73的 显示

SELECT subjectName,AVG(StudentResult)

FROM `result` r,`subject` s

WHERE

r.`SubjectNo`=s.`SubjectNo`

GROUP BY subjectName

HAVING AVG(StudentResult)>73

ORDER BY AVG(StudentResult) DESC

-- 03.统计每个年级的男女人数 多列进行分组

SELECT gradeid '年级编号',COUNT(sex) '性别人数',sex '性别'

FROM student

WHERE sex IS NOT NULL AND gradeid IS NOT NULL

GROUP BY gradeid,sex

-- 04. 找出每个课程成绩的前三名

SELECT \* FROM result r1

WHERE

(

SELECT COUNT(1) FROM result r2

WHERE r1.subjectNo=r2.`SubjectNo`

AND r1.studentresult<r2.studentresult

)<3

ORDER BY subjectNo,studentResult DESC

## 四：多表连接查询

引入：

比如之前写的小例子，查询学生的成绩，我们获取的是学生编号和成绩！

但是如果获取了学生姓名和成绩岂不是更好？ 但是学生姓名和成绩不在一张表中！

这时候就需要我们的连接查询！

常用的连接查询方式：

01. 内连接

02. 外连接

### 4.1：内连接

内连接是典型的最常用的连接查询！ 特点就是两个表中存在主外健关系时，通常使用！

查询两张表中共同的数据！

内连接的实现方式有两种：

01. 在where条件中指定连接条件

比如 查询学生姓名以及对应的年级名称

Select studentName,gradeName

from student,grade

Where student.gradeId=grade.gradeId

02.在form 子句中增加 inner join 表 on 关系

比如查询学生姓名，科目名称以及考试成绩

SELECT studentName,subjectName,studentresult

FROM student s

INNER JOIN result r ON s.studentNo=r.studentNo

INNER JOIN `subject` su ON su.subjectNo=r.subjectNo

注意点：

001.inner可以省略

002.inner join 用来连接两个表

003.on用来设置条件

004. s r su是用的别名

### 4.2：外连接

外连接查询是至少返回一个表中的所有记录，根据匹配条件有选择地返回另一张表的数据！

外连接有主表和从表的概念！

以主表为准匹配从表的数据，符合连接条件的数据直接返回到结果集中，不符合的数据将被赋予null之后再返回到结果集中！

外连接查询又分为：

01.左外连接 Left outer join

以左表为主表，从表（右边的表）中如果没有匹配的数据返回null

例子1： 查询学生的姓名，考试科目以及成绩！

SELECT studentName,subjectNo,studentResult FROM student s

LEFT JOIN result r ON r.`studentNo`=s.`studentNo`

例子2：查询所有科目对应的学生成绩

SELECT subjectName,s.subjectNo,studentResult FROM

`subject` s LEFT JOIN result r

ON s.`SubjectNo`=r.`SubjectNo`

02.右外连接 right outer join

以右表为主表，从表（左边的表）中如果没有匹配的数据返回null

例子：查询年级名称和学生名称 两个结果是否一致？

SELECT gradeName,studentName FROM grade

RIGHT JOIN student ON grade.`GradeID`=student.`GradeId`

SELECT gradeName,studentName FROM grade

INNER JOIN student ON grade.`GradeID`=student.`GradeId`

# ：事务,视图,索引,备份和恢复

## 一：事务

### 1.1：事务的定义

事务（transaction）：是将一系列数据操作捆绑成一个整体进行统一管理。

如果某个事务执行成功，那么事务中进行的所有数据更改都会提交，永久的保存在数据库中！

如果某个事务中途取消或者执行失败，那么事务中进行的所有数据更改都不会提交，会恢复到事务之前的状态！

简单来说：事务就是一条或者多条sql语句的集合！

### 1.2：事务的ACID特性

原子性 (Atomicity)：在事务中的操作，要么都执行，要么都不执行！

一致性（Consistency）：事务必须保证数据库从一个一致性的状态变成另一个一致性的状态！

隔离性（Isolation）:每个事务之间互不干扰！哪怕是并发执行也不干扰！

持久性（Durability）:事务一旦被改变，那么对数据库中数据的影响是永久性的！

### 1.3：事务的小例子

模拟 银行转账！

-- 创建数据库

CREATE DATABASE IF NOT EXISTS myBank;

-- 切换到指定的数据库

USE myBank;

-- 创建表

CREATE TABLE IF NOT EXISTS bank(

`name` VARCHAR(10) NOT NULL COMMENT'用户名',

`money` DECIMAL(10,2) NOT NULL COMMENT'账户余额'

);

-- 同时插入多条数据

INSERT INTO bank(`name`,money)

VALUES('小黑',10000),('小白',20000);

-- 模拟小黑给小白转账

-- 小黑-100

-- 小白+100

-- 上诉两条数据 都执行成功 才算转账成功

UPDATE bank SET money=money-100 WHERE `name`='小黑';

UPDATE bank SET money=money+100 WHERE `name`='小白1';

-- 转账的用户名不存在，但是小黑的钱没有了

-- 把上述两条sql捆绑在一起成为一个事务

-- 开启事务 START TRANSACTION 或 BEGIN

BEGIN;

UPDATE bank SET money=money-100 WHERE `name`='小黑';

UPDATE bank SET money=money+100 WHERE `name`='小白1';

-- 如果没有取消自动提交事务 那么小黑的钱还是 减了

COMMIT; -- 提交事务

-- 取消自动提交事务 SET autocommit=0(关闭自动提交)autocommit=1（开启自动提交）

SET autocommit=0;

BEGIN;

UPDATE bank SET money=money-100 WHERE `name`='小黑';

UPDATE bank SET money=money+100 WHERE `name`='小白1';

ROLLBACK; -- 事务回滚 小黑的钱就不减少了

COMMIT;

-- 取消自动提交事务

SET autocommit=0;

BEGIN;

UPDATE bank SET money=money-100 WHERE `name`='小黑';

UPDATE bank SET money=money+100 WHERE `name`='小白1';

COMMIT;

SET autocommit=1; -- 开启自动提交事务

UPDATE bank SET money=0 WHERE `name`='小黑';

mysql中是自动提交事务的

如果想取消自动提交事务

需要设置 set autocommit=0;

-- 取消自动提交事务

SET autocommit=0;

BEGIN; -- 开启事务

UPDATE teacher SET sal=sal-1000 WHERE `NAME`='小白1';

SAVEPOINT ponit1; -- 设置回滚点

UPDATE teacher SET sal=0 WHERE `NAME`='小白1';

ROLLBACK TO ponit1; -- 返回到指定的回滚点 ROLLBACK; -- 回滚 直接到达最初的位置

COMMIT; -- 提交事务

### 1.4：事务的书写原则

1.事务尽可能简短

因为事务启动后，数据库管理系统会保留大量资源来满足事务的ACID特性，如果多用户系统中，运行较大的事务会严重影响系统性能！

1. 事务中访问的数据量尽量要小

当并发的情况下执行事务时，事务操作的数据量越小，事务之间对数据的争夺就越小！

1. 查询数据时尽量不使用事务

因为查询不会对数据库中的数据进行更改，所以应避免使用查询事务！

1. 在事务处理过程中，尽量不要出现等待用户输入的操作

因为如果等待时间长，占用资源，有可能造成系统阻塞！

### 1.5：事务的隔离级别

1.事务隔离级别的引入

在数据库操作中，为了有效保证并发读取数据的正确性，提出的事务隔离级别。

更新丢失

两个事务都同时更新一行数据，一个事务对数据的更新把另一个事务对数据的更新覆盖了。这是因为系统没有执行任何的锁操作，因此并发事务并没有被隔离开来。

脏读

一个事务读取到了另一个事务未提交的数据操作结果。这是相当危险的，因为很可能所有的操作都被回滚。

不可重复读

不可重复读（Non-repeatable Reads）：一个事务对同一行数据重复读取两次，但是却得到了不同的结果。

包括以下情况：

（**1**）虚读：事务T1读取某一数据后，事务T2对其做了修改，当事务T1再次读该数据时得到与前一次不同的值。

（**2**） 幻读（Phantom Reads）：事务在操作过程中进行两次查询，第二次查询的结果包含了第一次查询中未出现的数据或者缺少了第一次查询中出现的数据（这里并不要求两次查询的SQL语句相同）。这是因为在两次查询过程中有另外一个事务插入数据造成的。

2.解决方案

未授权读取

也称为读未提交（Read Uncommitted）：允许脏读取，但不允许更新丢失。如果一个事务已经开始写数据，则另外一个事务则不允许同时进行写操作，但允许其他事务读此行数据。该隔离级别可以通过“排他写锁”实现。

授权读取

也称为读提交（Read Committed）：允许不可重复读取，但不允许脏读取。这可以通过“瞬间共享读锁”和“排他写锁”实现。读取数据的事务允许其他事务继续访问该行数据，但是未提交的写事务将会禁止其他事务访问该行。

可重复读取（Repeatable Read）

可重复读取（Repeatable Read）：禁止不可重复读取和脏读取，但是有时可能出现幻读数据。这可以通过“共享读锁”和“排他写锁”实现。读取数据的事务将会禁止写事务（但允许读事务），写事务则禁止任何其他事务。

序列化（Serializable）

序列化（Serializable）：提供严格的事务隔离。它要求事务序列化执行，事务只能一个接着一个地执行，不能并发执行。仅仅通过“行级锁”是无法实现事务序列化的，必须通过其他机制保证新插入的数据不会被刚执行查询操作的事务访问到。

隔离级别越高，越能保证数据的完整性和一致性，但是对并发性能的影响也越大。对于多数应用程序，可以优先考虑把数据库系统的隔离级别设为Read Committed。它能够避免脏读取，而且具有较好的并发性能。尽管它会导致不可重复读、幻读和第二类丢失更新这些并发问题，在可能出现这类问题的个别场合，可以由应用程序采用悲观锁或乐观锁来控制。

博文连接 ：http://lvwenwen.iteye.com/blog/**2045951**

3.使用事务隔离级别

SET tx\_isolation='read-uncommitted';

SET tx\_isolation='read-committed'

SET tx\_isolation='repeatable-read';

-- 案例1：验证可以脏读取

-- 查询mysql默认的事务隔离级别

SELECT @@tx\_isolation;

-- 01.在SqlYog中设置mysql的事务隔离级别

SET tx\_isolation='Read-uncommitted'

-- 02.使用doc命令创建连接mysql数据库

mysql -uroot -p

-- 03.设置mysql的事务隔离级别

SET tx\_isolation='Read-uncommitted'

-- 04.在doc窗口中 执行

BEGIN;

UPDATE bank SET moeny=50000 WHERE NAME='小黑';

-- 05.在SqlYog中就可以看到这个脏数据

-- 06.在doc窗口中 执行

ROLLBACK;

-- 07.在SqlYog中就不可以看到这个脏数据 因为doc中回滚了！

-- 案例2：验证不可以更新丢失

-- 查询mysql默认的事务隔离级别

SELECT @@tx\_isolation;

-- 01.在SqlYog中设置mysql的事务隔离级别

SET tx\_isolation='Read-uncommitted'

-- 02.使用doc命令创建连接mysql数据库

mysql -uroot -p

-- 03.设置mysql的事务隔离级别

SET tx\_isolation='Read-uncommitted'

-- 04.在doc窗口中 执行

BEGIN;

UPDATE bank SET moeny=money+50000 WHERE NAME='小黑';

-- 05.在SqlYog中执行

UPDATE bank SET moeny=money+20 WHERE NAME='小黑';

-- 可以看到这个update 一直在等待doc窗口中的commit;

-- 06.在doc窗口中 执行

COMMIT;

-- 07.在SqlYog中不可以看到正确的结果！

## 二：视图

### 2.1：什么是视图

01.视图就是一条SELECT语句执行后返回的结果集。

比如说：我们的student表中有10个字段，我们现在只需要拿到studentName和address！

Sql语句就是： select studentName,address from student;  
 这条语句的查询结果我们就可以理解成一个视图！只有学生姓名和住址的视图！

02.视图是一张虚拟的表！

### 2.2：为什么是需要视图

01.方便查询操作，减少负责的sql语句，增强可读性。

比如说：老师经常查询学生的某门课程的成绩！

如果总是执行三表联查或者使用子查询来得到结果，显示是非常麻烦的！

这时候我们只需要写一次复杂的sql，然后把结果保存在视图中，以后访问视图，

即可以得到想要的结果！

02.提高数据库表的安全性。

比如说：在公司中，不同的职位查询的信息，不一样！

老板可以看到所有信息，

人事可以看到所有员工的薪水,

而普通员工只能看到自己的信息！

这种情况，就可以使用视图，不同的权限查询不同的结果！

还有一点就是，数据库的grant授权命令不能精确的限定到指定的行或者列！

而视图就可以！

比如说 薪水这一列，不能让其他人看到，我们就可以在视图中不加入此列即可！

使数据库中的表更安全！

### 2.3：视图的使用

视图的命名规范：

view\_xxx或者v\_xxx

1. 创建视图

Create view 视图名称

As

Select语句;

1. 查询视图

Select \* from 视图名称; 查询所有

Select 字段1,字段2 from 视图名称; 查询指定

1. 删除视图

Drop view [if exists] 视图名称;

4.操作视图的小例子

-- 创建一个视图 只获取 学生姓名 编号 以及考试成绩

CREATE VIEW view\_student\_result

AS

SELECT s.studentNo 学生编号,studentName 学生姓名,

studentResult 考试成绩

FROM student s,result r

WHERE s.`studentNo`=r.`studentNo`

-- 查询视图中的内容

SELECT \* FROM view\_student\_result

-- 查询mysql数据库中所有的视图

SELECT \* FROM information\_schema.views;

-- 删除视图

DROP VIEW view\_student\_result;

-- 创建一个表的视图 学生姓名，地址，手机号

CREATE VIEW view\_student

AS

SELECT studentName 学生姓名,address 地址,phone 手机号

FROM student

-- 查询视图

SELECT \* FROM view\_student

### 2.4：视图使用注意事项

**01**.表示一张表的部分数据或者是多张表的综合数据！

**02**.结构和数据都是建立在对真表的查询基础之上的！

**03**.视图中存放的数据其实就是对真实表的引用！

对视图中的数据进行添加，更新删除都会影响到真实的表！

**04**.一个真实的表可以创建N个视图！

**05**.如果视图关联了多表，不允许增 删！单表可以增删改

**06**.视图一般都是使用查询！

## 三：索引

### 3.1：什么是索引

索引是一个特殊的文件，文件中保存了对数据库中所有表中记录的引用！

### 3.2：为什么需要索引

索引类似于我们书本的目录！目的就是便于我们查询指定的内容！

通过使用索引，可以大大的提高数据库检索速度，从而提高数据库的性能！

### 3.3：索引的分类

mysql索引按存储类型分2种:

**01**.B-树索引（BTREE）:INNODB,MyISAM 支持

**02**.哈希索引

**01**.普通索引：允许在定义为索引的列中 出现 重复值和空值！

**02**.唯一索引：索引列不允许出现重复值，但是可以出现一个空值！

**03**.主键索引：创建主键的时候，系统会默认创建一个主键索引！唯一！不能为空！

**04**.复合索引：将我们的多个列组合起来！（name,sid）姓名和身份证号！ name被查询时才能触发索引的使用！

需要注意的是，只有查询中使用了复合索引最左边的字段时，索引才会被使用！

**05**.全文索引：允许值的全文查询！允许空值和重复值！适合在一个内容比较多的列！text！

**06**.空间索引：对空间数据类型的列建立的索引！

### 3.4：索引的使用

-- 创建索引

CREATE [UNIQUE|FULLTEXT|SPATIAL] INDEX index\_name

ON

table\_name(column\_name[LENGTH])

UNIQUE：唯一索引

FULLTEXT：全文索引

SPATIAL:空间索引

index\_name:索引名称

table\_name:表名

column\_name:字段名称

LENGTH:索引长度,只能是字符串类型才能指定索引的长度

-- 给姓名增加 普通索引

CREATE INDEX index\_studentName

ON student(studentName);

-- 给学生姓名和身份证号 增加 组合索引

CREATE INDEX index\_name\_sid

ON student(studentName,identityCard)

-- 查询指定表的索引

SHOW INDEX FROM student;

TABLE:索引所在的表

Non\_unique:索引是否唯一 **0**:唯一 **1**：不唯一

key\_name：索引名称

seq\_in\_index:该列在索引中的位置，如果索引为单列值为1，是复合索引，则是索引中定义的顺序

column\_name:定义所用的列名称null：该列是否为空

Collation :列以什么方式存储在索引中。A:升序 null:无分类

Cardinality:

1. 列值代表的是此列中存储的唯一值的个数（如果此列为primary key 则值为记录的行数）  
   2. 列值只是个估计值，并不准确。  
   3. 列值不会自动更新，需要通过Analyze table来更新一张表或者mysqlcheck -Aa来进行更新整个数据库。  
   4. 列值的大小影响Join时是否选用这个Index的判断。  
   5. 创建Index时，MyISAM的表Cardinality的值为null，InnoDB的表Cardinality的值大概为行数。  
   6. MyISAM与InnoDB对于Cardinality的计算方式不同。

Packed 指示关键字如何被压缩。如果没有被压缩，则为NULL。

index\_type:索引类型

-- 删除索引

DROP INDEX index\_name\_sid ON student;

DROP INDEX index\_studentName ON student;

-- 创建索引的原则

**01**.经常被查询的列

**02**.经常用作选择的列

**03**.经常排序，分组的列

**04**.经常用作连接的列（主键/外键）

使用索引时的注意事项：

**01**.查询时减少使用\*返回全部的列，不要返回不需要的列！

**02**.索引尽量要少，在字节数小的列上建立索引！

**03**.where字句中有多个条件表达式的时候，包含索引的列要放在其他表达式之前！

**04**.在order by的字句中避免使用表达式！

## 四：数据库的备份和恢复

为了防止数据的丢失，将损失降到最低，定期对数据库进行备份是非常必要的！以便于发生意外情况后，可以对数据库及时的恢复！

### 4.1：使用mysqldump命令备份数据库

备份数据库的语法：

mysqldump -u userName -h HOST -p PASSWORD

Dbname [tbname1 tbname2...] [参数] > filename.sql

userName:用户名

HOST：登陆用户的主机名称，如果是本机可以省略

PASSWORD：登陆密码

Dbname：需要备份的数据库

tbname1：需要备份的数据库表

filename.sql：备份之后的文件位置

小例子：

备份数据库中的两张表的数据

mysqldump -u root -p 数据库名称 表1 表2 > 目标位置

常用的参数配置：

01.-t或者-no-create-info :只导出数据，不添加create table语句

02.-c或者--complete-insert：每个insert语句前都加列名，在数据导入另一个数据库时有用！

03.-d或者--no-data:不存储表的任何行信息，只转储表结构

还有很多其他的配置参数

通过 mysqldump --help;查询

### 4.2：使用mysql命令恢复数据库

第一种方式：无须登陆mysql

恢复数据库的语法

Mysql -u userName -p dbname < filename.sql

注意点

Dbname数据库必须已经存在，否则会报错！

第二种方式：登陆mysql

Source e:\a.sql;

### 4.3：使用复制文件实现备份和恢复数据库

注意点：

01.执行flush tables，确保将所有数据写入到数据文件中

02.停止mysql服务

03.找到mysql安装目录中的data目录！复制文件即可！

04.存在版本不兼容的问题！

### 4.4：将数据导入到文本文件中

语法：

Select \*或者字段 from 表名

[ Where 条件 ]

Into outfile ‘文件位置’

例子：

把学生姓名和年级名称 放入到文件中

SELECT studentName,gradeName FROM student

INNER JOIN grade

ON student.gradeid=grade.gradeid

INTO OUTFILE 'e:/haha.txt'

FIELDS

TERMINATED BY '\，' --字段之间的分隔符

ENCLOSED BY '\"' --包裹字段的符号

LINES

STARTING BY '\>' --每行数据开头符号

TERMINATED BY '\r\n' --每行数据结束符号

为了使导出的数据有可读性，我们可以设置导出文件的格式。

常用的参数配置：

01.Fields terminated by ‘字符’ ：设置字段之前的分隔符，默认是\t空格

02.Fields enclosed by ‘字符’ ：设置包裹字段的符号。默认没有任何符号

如果设置了optionally，则只有char和varchar等字符串字段被包裹！

03.Fields escaped by ‘字符’ ：设置转义字符的转义符号，默认是 ’ \ ’

04.lines starting by ‘字符’ ：设置每行开头的字符。默认没有任何符号

05.lines terminated by ‘字符’ ：设置每行结束的字符。默认是’ \ ’换行

### 4.5：将文本文件导入到数据库

注意点：

如果导出文本时，设置了参数，那么导入的时候务必写上对应的参数！

语法：

Load data infile 文件名称 into table 表名称 [option];

LOAD DATA INFILE

'e:/haha.txt' INTO TABLE 表名称

FIELDS

TERMINATED BY '\，'

ENCLOSED BY '\"'

LINES

STARTING BY '\>'

TERMINATED BY '\r\n'

注意点：

01.表事先已经存在了

02.表中字段的赋值，会根据文本文件中一行数据的顺序添入

# ：数据库设计

问题1：

大家小时候都玩过积木？

我们再落积木的时候，有考虑过怎么落既漂亮有牢固吗？？

问题2：

房地产开发商在开发一个楼盘的时候，会找专业的人士来设计吗？

答案是肯定的！

那么我们的数据库在创建的时候是否应该规划一下？设计一下结构呢！！！

### 5.1：什么是数据库设计

对于一个给定的应用环境，构造最优的数据库模式，建立[数据库及其应用](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93%E5%8F%8A%E5%85%B6%E5%BA%94%E7%94%A8" \t "https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93%E8%AE%BE%E8%AE%A1/_blank)系统，使之能够有效地[存储数据](https://baike.baidu.com/item/%E5%AD%98%E5%82%A8%E6%95%B0%E6%8D%AE" \t "https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93%E8%AE%BE%E8%AE%A1/_blank)，满足各种用户的应用需求。这种设计我们称之为[数据库设计](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93%E8%AE%BE%E8%AE%A1" \t "https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93%E8%AE%BE%E8%AE%A1/_blank)(Database Design)！

在数据库领域内，常常把使用数据库的各类系统统称为[数据库应用系统](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93%E5%BA%94%E7%94%A8%E7%B3%BB%E7%BB%9F" \t "https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93%E8%AE%BE%E8%AE%A1/_blank)（DBAS DataBase Application System）。

如果我们对数据库的设计不合理？

01.我们更新和检索数据的时候会出现各种问题；

02.我们在使用数据库的时候，效率会比较低！

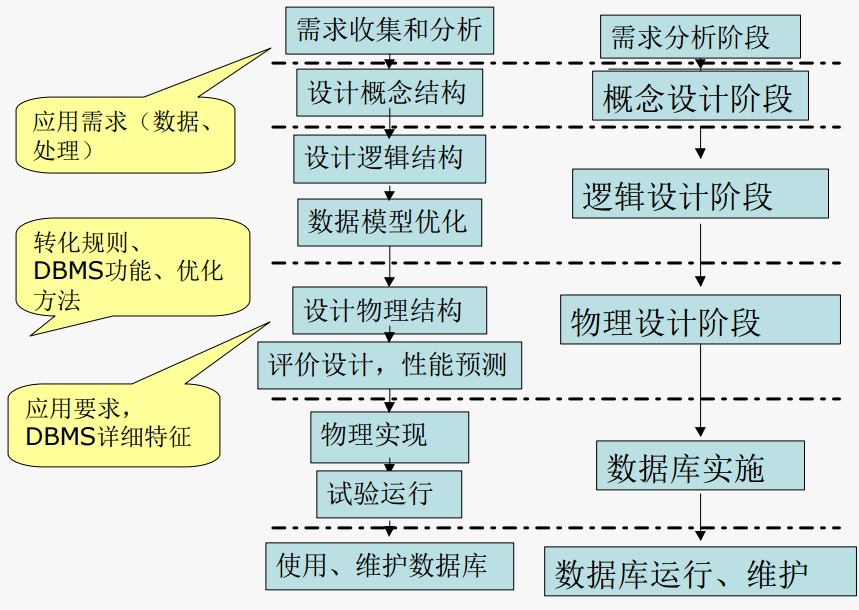
那么良好的数据库设计会给我们带来的好处？

01.我们使用数据库时效率高；

02.便于我们后期的维护和扩展；（大家都知道我们java中也有开闭原则，数据库中也不例外）;

03.可以使我们的应用程序的开发变得更容易!

### 5.2：数据库设计的步骤



1.需求分析阶段：分析客户的业务和数据处理需求

2.概要设计阶段：设计数据库的E-R模型图，设计人员和客户之间进行沟通，确认需求信息的正确和完整

3.逻辑设计阶段：将E-R模型图转换成多张表，进行逻辑设计！应用三大范式审核数据库结构

4.物理设计阶段：将逻辑视图转换成物理视图！

5.代码实施阶段：根据物理视图，书写对应的代码，创建数据库对象

6.安装部署阶段：数据库运行和维护

### 5.3：PowerDesigner的使用之创建概念模型

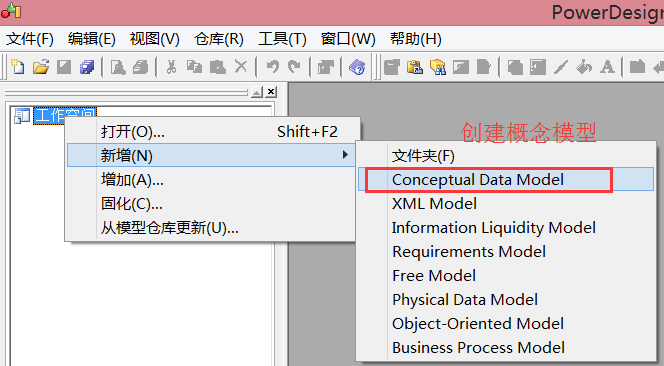
****一、概念数据模型概述****数据模型是现实世界中数据特征的抽象。数据模型应该满足三个方面的要求：  
1）能够比较真实地模拟现实世界  
2）容易为人所理解  
3）便于计算机实现

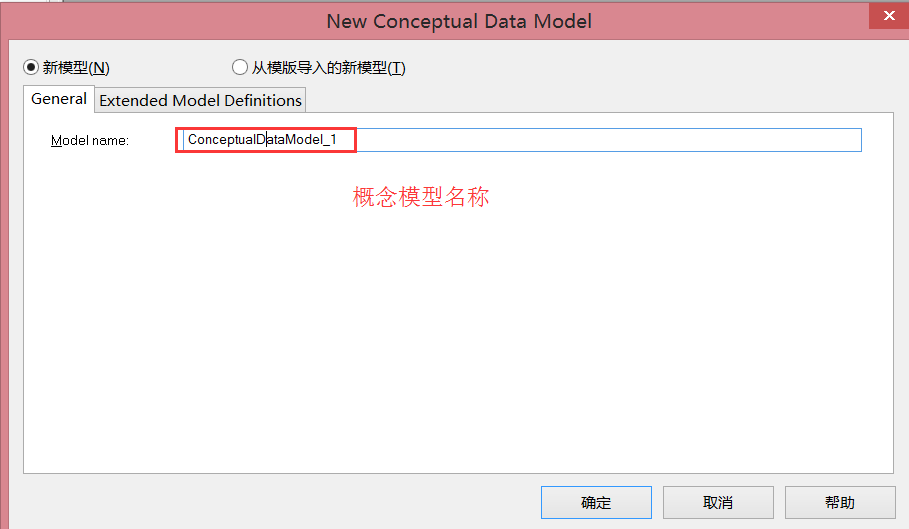
概念数据模型也称信息模型，它以实体－联系(Entity-RelationShip,简称E-R)理论为基础，并对这一理论进行了扩充。它从用户的观点出发对信息进行建模，主要用于数据库的概念级设计。

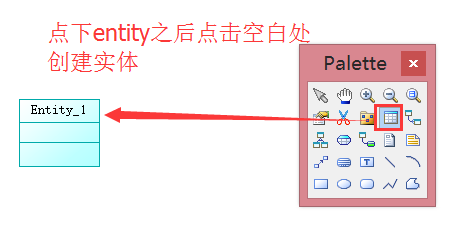
先将现实世界中的客观对象抽象为实体(Entity)和联系 (Relationship),它并不依赖于具体的计算机系统或某个DBMS系统，这种模型就是我们所说的CDM;然后再将CDM转换为计算机上某个 DBMS所支持的数据模型，这样的模型就是物理数据模型,即PDM。

CDM：（concept data model）概念数据模型。可以由其生成PDM（物理数据模型），进而生成数据库的表结构。

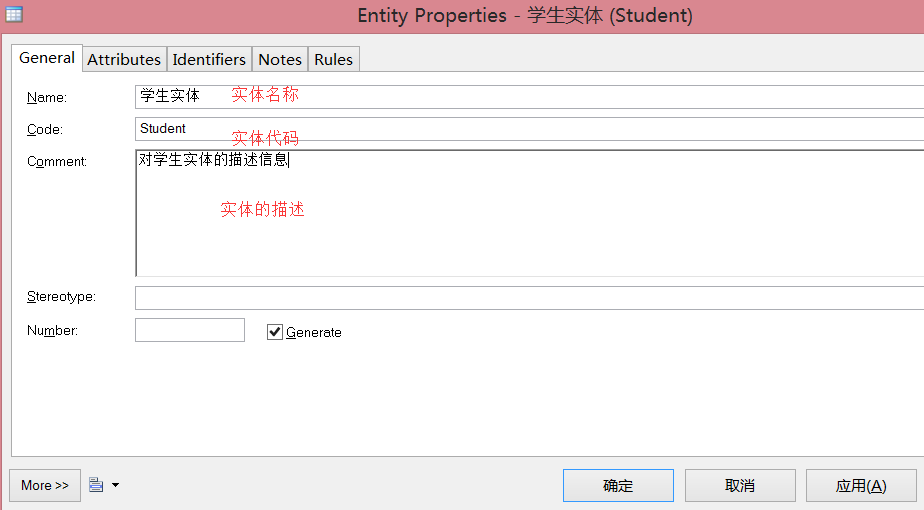
PDM（Physical Data Model）物理数据模型，提供了系统初始设计所需要的基础元素，以及相关元素之间的关系。

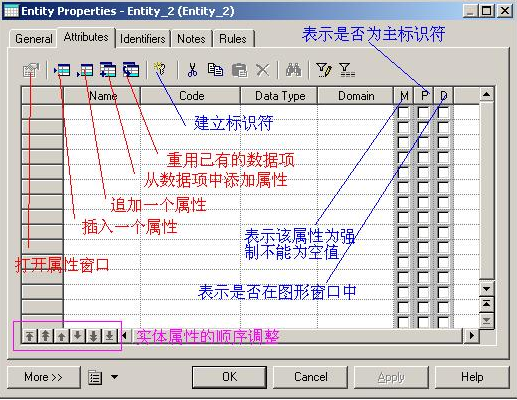


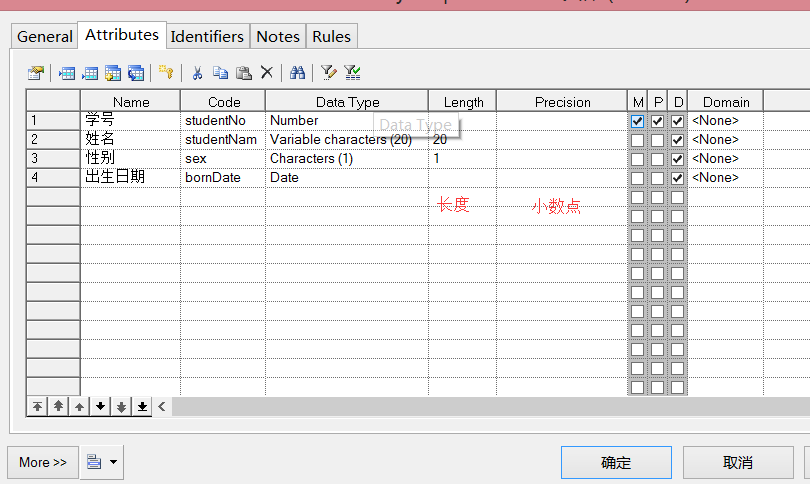


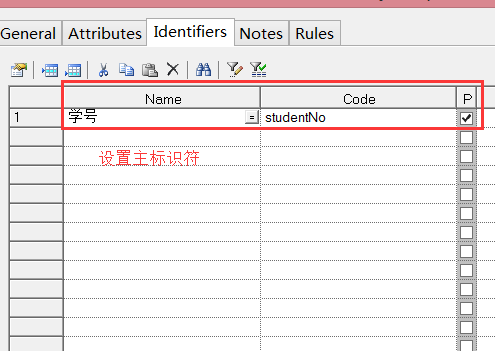


之后双击Entity\_1 图标会显示如下页面



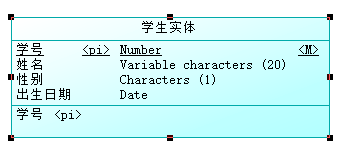




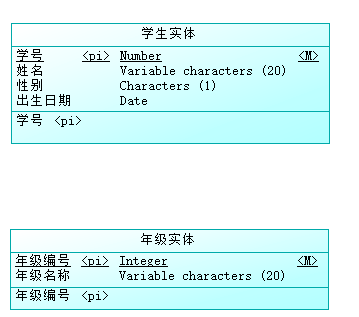


一、标识符  
标识符是实体中一个或多个属性的集合，可用来唯一标识实体中的一个实例。要强调的是，CDM中的标识符等价于PDM中的主键或候选键。   
每个实体都必须至少有一个标识符。如果实体只有一个标识符，则它为实体的主标识符。如果实体有多个标识符，则其中一个被指定为主标识符，其余的标识符就是次标识符了。

应用之后！效果图如下



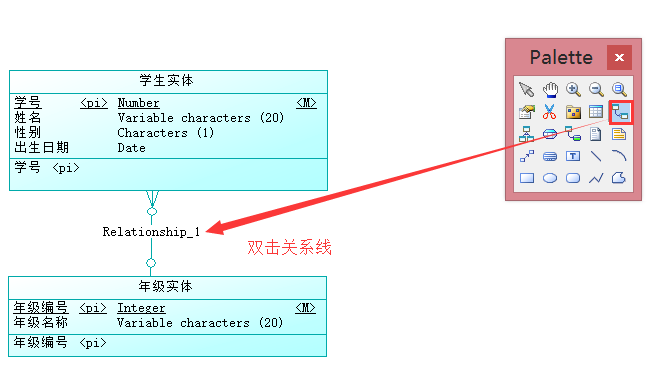
按照上诉操作 创建出年级表

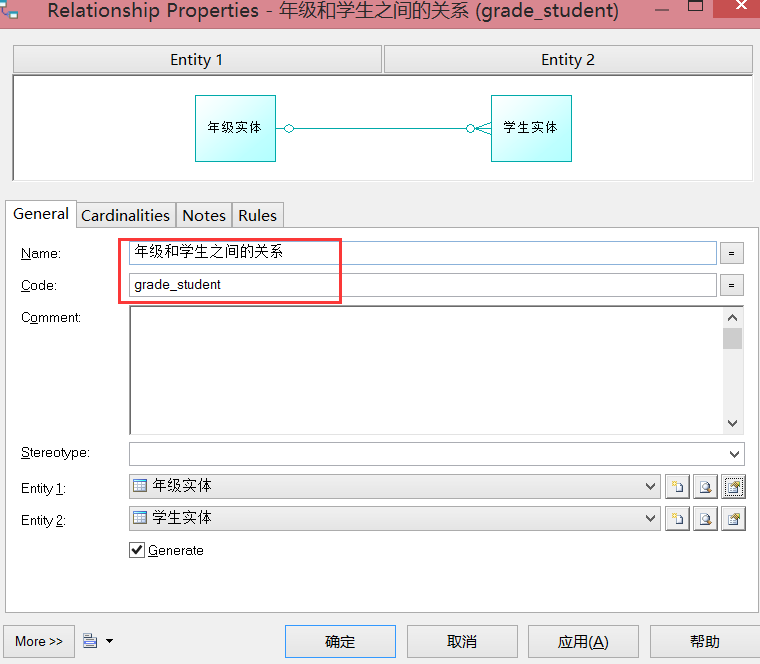


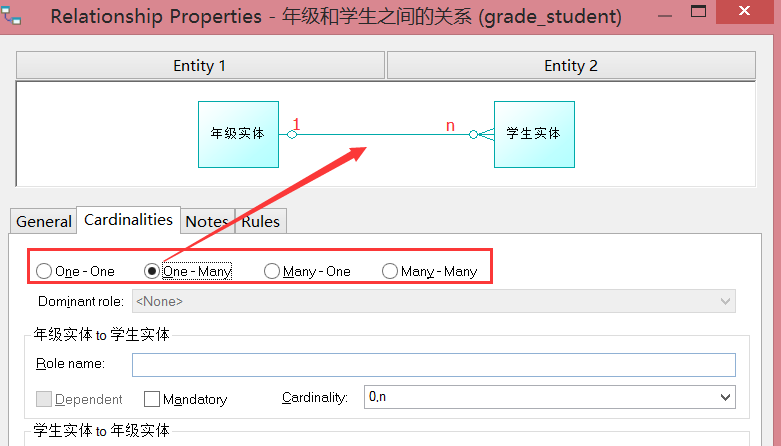
建立年级与学生之间的关联关系

一个年级对应多个学生，

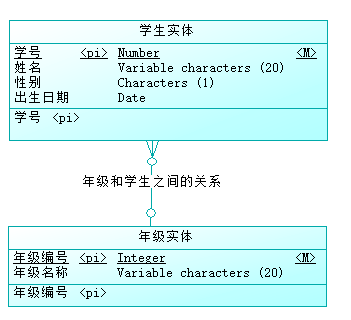
多个学生对应一个年级。



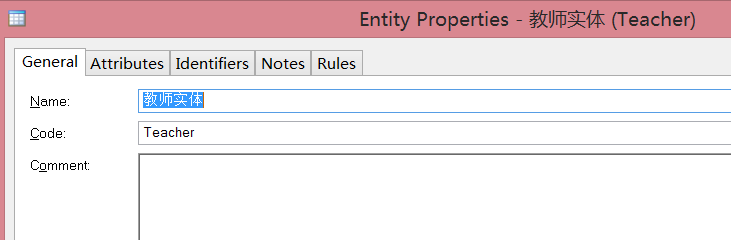


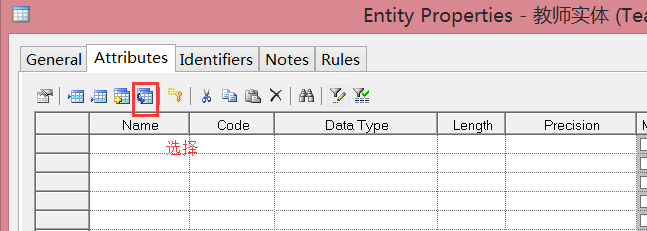


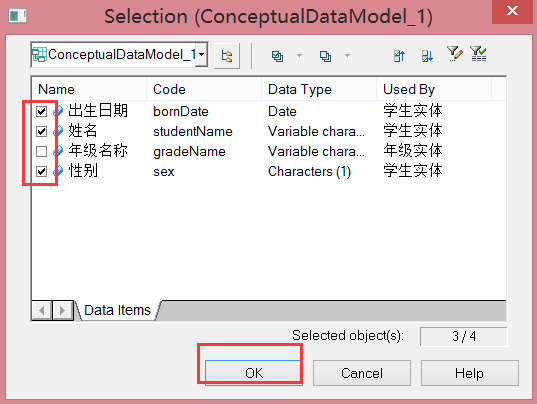
效果图如下

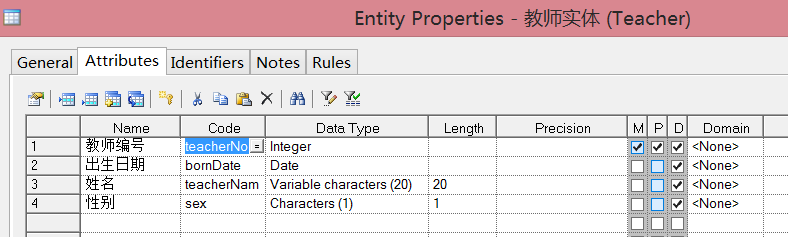


新建教师表 可以复用学生的实体属性







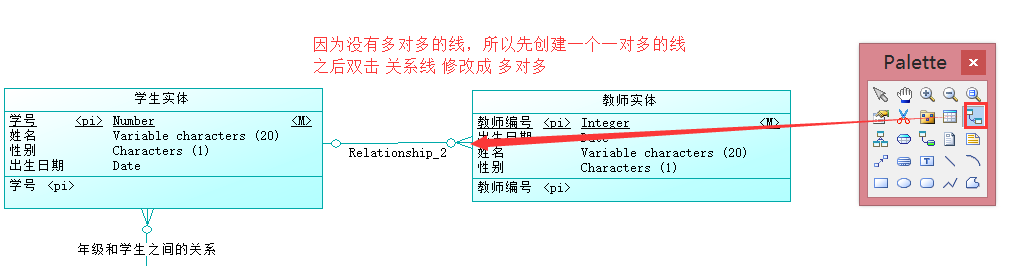


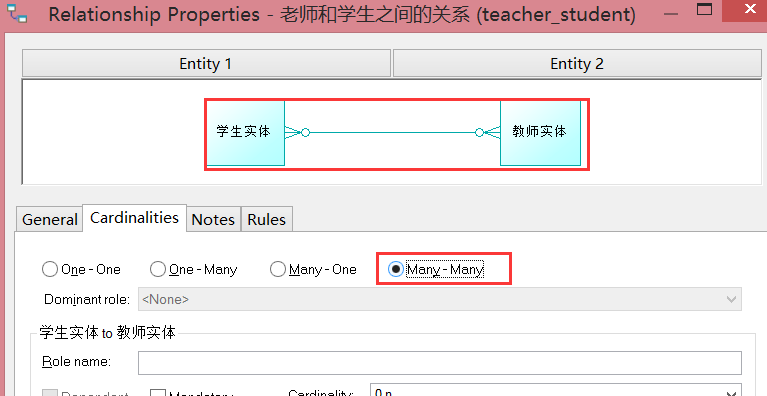
建立教师和学生之前的关系

一个学生可以有多个老师

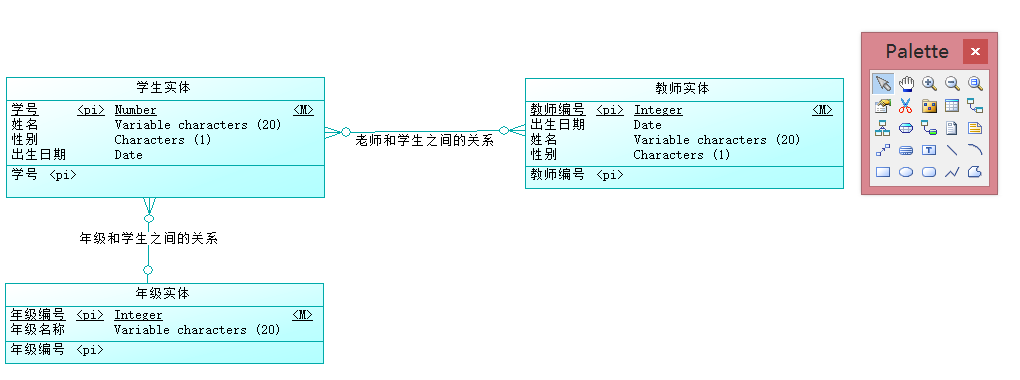
一个老师可以有多个学生

两个一对多的关联关系 就组成了 多对多的关系

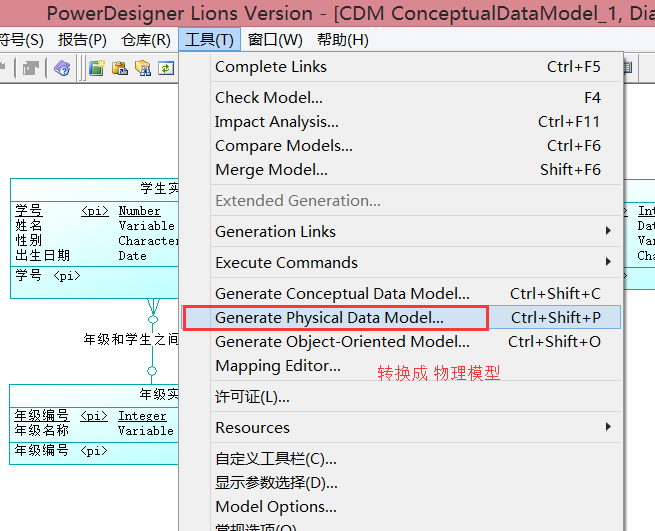


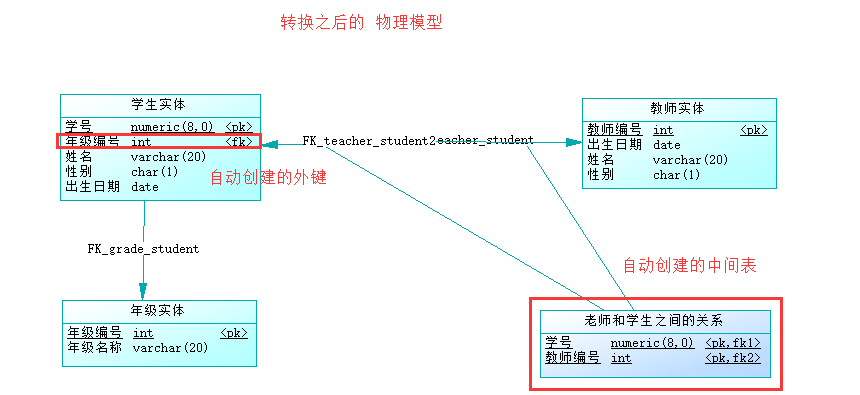


整体效果图如下：

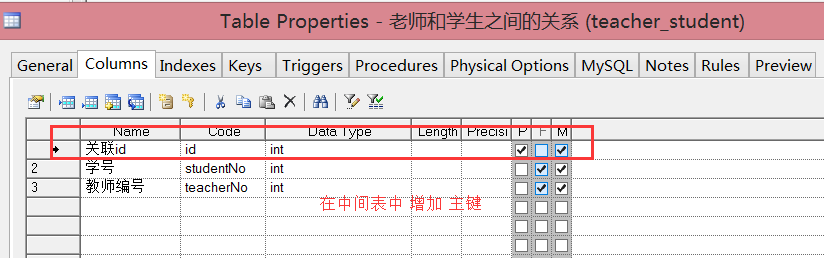


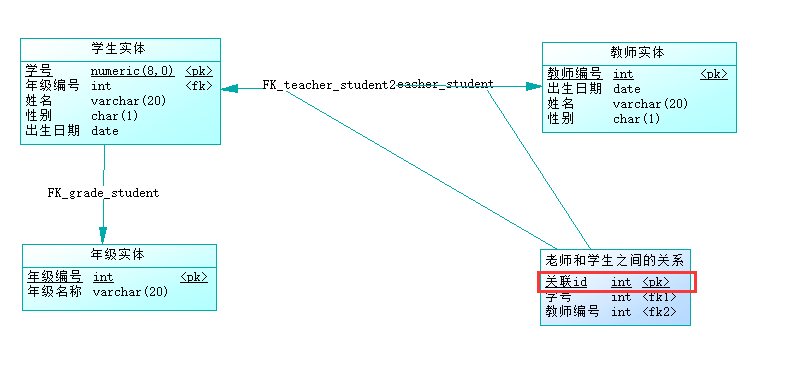
### 5.4：概念模型转换成物理模型



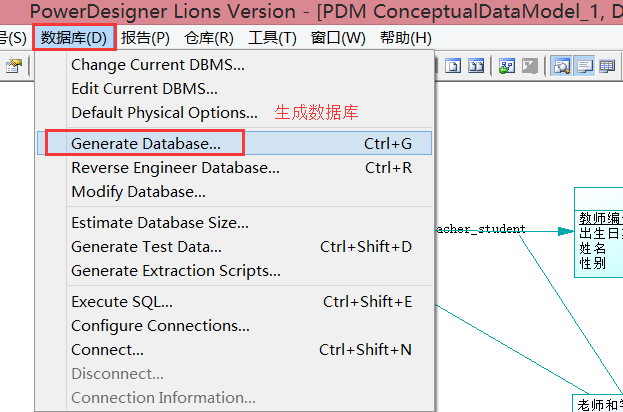


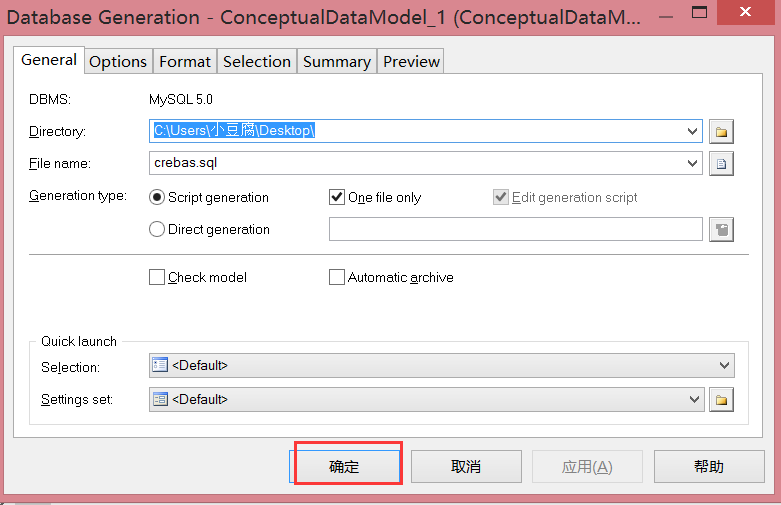
双击中间表 创建中间表的主键id





### 5.5：物理模型转换成数据库中的表





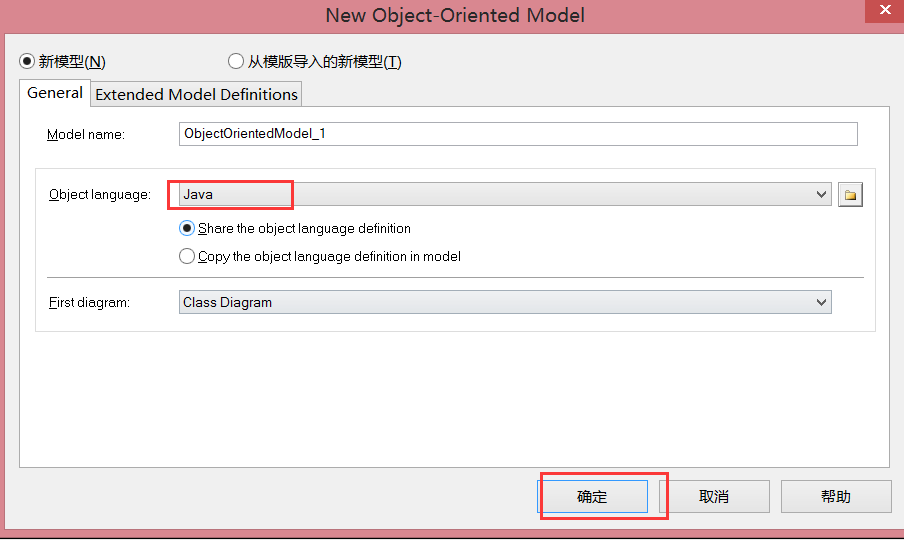
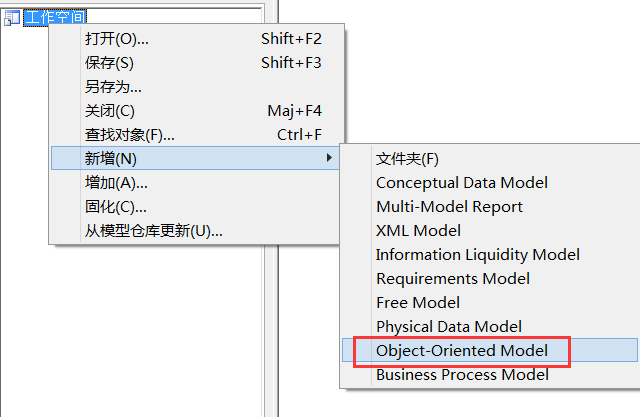
在mysql中运行生成的sql语句 即可

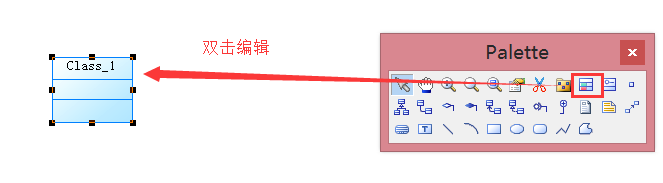
### 5.6：OOM面向对象模型

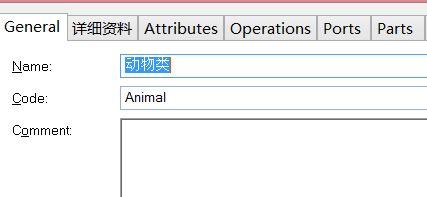
UML (unified Modeing Language) 标准建模语言

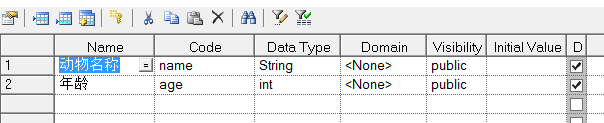
* 用例图(User Case Diagram):通常用来定义系统的高层次草图，它从用户角度描述了应用的系统功能，指出了各个功能的外部操作者。
* 序列图(Sequence Diagram):描述系统如何实际完成在User Case图中定义的功能。可以画出对象(类的实例)之间交互时产生的时序关系。
* 类图(Class Diagram):描述类与类之间的静态关系。其中包括一系列的包，类，接口和它们之间的关系。类图是定义其他图的基础。

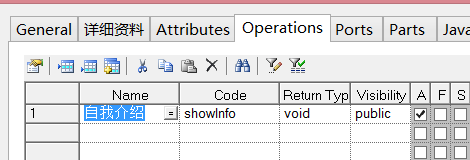
创建类图实例



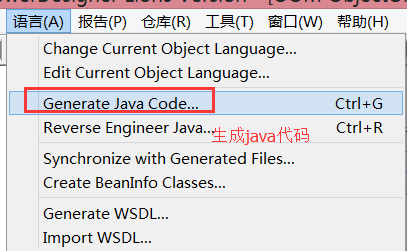








安装上述操作创建出Bird类 和 fly接口



类和类之间的关系有如下几种关系：

泛化（Generalization） extends

实现（Realization）implements

关联（Association) 类中的一个属性类型是另一个类

聚合（Aggregation）类中的一个属性类型是一个集合

组合(Composition)类中的一个属性类型是另一个类或者集合，

真正使用的时候是在方法中

依赖(Dependency)关系对象出现在局部变量或者方法的参数里，或者关系类的静态方法被调用 

各种关系的强弱顺序：

泛化 >实现>组合> 聚合> 关联> 依赖