2014 年辛星 mysql 教程秋季版第一本

----------即 mysql 夯实基础 ----------

**********说明**********

1.本书的编写时间是 2014 年 8 月，如果您在一年后看到了本书，可以考虑搜索它的更新版本。

2.百度搜索“辛星 mysql”可以找到更多更全的 mysql 教程，打造史上最新最全最实用的 mysql 教程系列。

3.我的邮箱：xinguimeng@163.com，我的博客：blog.csdn.net/xinguimeng，在置顶博文中找到更多资源，绝对都是免费下载，我的 QQ：1808347923。

特色：更新更全更实用。

目标：传播编程知识，振兴中华软件。

动力：用心打造优秀教程。

前进的道路，辛星陪伴您。

只要星哥在，编程充满爱。

mysql 秋季版暂定内容(后续版本肯定会变更较大,敬请期待)：

第一本：mysql 夯实基础(适合零基础的读者朋友们)

第二本：mysql 进阶(适合有数据库基础的朋友)

第三本：mysql 精通(适合有较为扎实 mysql 基础的朋友)

第四本：mysql 优化(适合所有类型的读者朋友)

第五本：数据库理论及建模(适合本科在校学生或者工作人员)

第五本：写给 dba(适合致力于 dba 工作学习人群)

其他：暂未想好

**********说明**********

1.由于本系列书目是我第一次编写，可能问题较多，还望读者能够及时给我指正，邮件：xinguimeng@163.com 。

2.本套教程中的所有文档都是免费下载，同时也欢迎各位高手的批评指正，也欢迎新手的学习。

****

目录：

>>>>>>>>>>>何去何从<<<<<<<<<<<<

*********教程*********手册**********培训机构**********

1.这可能是让新手朋友们最迷茫的一件事了，那就是我该去抱一个培训机构还是去看教程，还是直接去看手册？

2.我可以给大家一个建议：如果您时间比较紧张，那我还是建议您抱一个培训机构，培训机构是快速入门、快速上手的不二法宝。

3.如果您时间比较宽裕，那我建议您还是自学比较好一点，毕竟，“有毅力，当自学”，自学到的知识不仅牢固，而且体会深刻，我们辛星系列就是专门为自学者打造的一系列书籍。

4.当您入门之后，就可以把教程丢在一边了，只保留一本手册即可了，因为手册里面的资料是最权威而且最全面的，它是任何一本教程都做不到的。

*****************辛星系列 ***************

1.传播编程知识，振兴中华软件，我心永恒，绝不动摇。

2.辛星系列的书籍全部都提供免费下载，而且目前统一为 pdf 格式。

3.它是“开源”的，即它的内容绝对无保留的展示给大家，读者可以拿去印刷和使用，还可以进行二次整理，还可以把它修改后的内容告诉我。

4.它是“免费”的，任何人都可以从我的置顶博文中找到它的下载地址：blog.csdn.net/xinguimeng。

5.如果您觉得这本书很不错，不妨帮忙推荐一下，毕竟“传播编程知识，振兴中华软件”这个伟大目标的实现，需要我们共同努力。

6.如果您感觉这本书有问题，不妨直接发邮件给我，因为这样会在以后的版本中彻底杜绝此种问题。

**********特色特点**********

1.如果您想问“辛星系列”和市面上其他书籍的区别，那我可以很明确的告诉您。

2.第一，它比那些只有十几页二十几页的教程更加系统和全面，因为十几页的教程固然看着方便，但是内容是在不够丰富。

3.第二，它比那些动辄五六百页的书更加灵活，辛星系列的书在原则上都是不超过三百页的，不会让读者一看到就头先大了一半，它会把一本七百页的内容拆成四个两百页的内容。

4.第三，它比那些陈旧的内容更新更实用，“更新更全更实用”是我们目前对自己的要求。

5.第四，它更加适合读者跟着学习，因为它的实践性很强，也一直保持着自己的独特风格。

**********拿破仑精神**********

1.拿破仑曾经说过一句话：“所有的士兵在经历过真正的洗礼之后，其归宿只有一个”。

2.我也想说一句话：“所有的知识在经历过真正优秀的讲师之后，都会很好理解。如果您 感觉某一部分知识暴难理解，那是讲师不够优秀。”

第一部分：开发环境与简单配置

第一节：mysql 快速简介

**********说明**********

1.我最烦的就是写这些形式上的东西，但是我对它的历史还是挺感兴趣的。

2.我以最快的速度介绍一下，当然这些东西百度百科上是找不到的，都是我自己收藏的。

**********艰苦的早期**********

1.mysql 最早可以追溯到 1979 年，在遥远的瑞典，创始人是 Monty Widenius，当时在一个叫做 Tcx 的小公司打工，用 BASIC 设计了一个报表工具。

2.1985 年瑞典的几位小伙子创建了一家公司，为瑞典的一些大型零售商提供数据仓库的服务。

3.1990 年，Tcx 公司的客户希望他们的数据库提供 SQL 支持。Monty 雄心大起，决定自己重写一个 SQL 支持。

4.六年时间一闪而过，1996 年，MySQL 的 1.0 版本发布，这是一个质的飞跃。

5.1996 年 10 月，3.11.1 版本发布，令我摸不着头脑的是，它没有 2.x 版本。

6.当时的 mysql 超简单，只能实现最简单的增删改查，没有其他功能。随后的两年里，mysql 被移植到各个平台。

**********辉煌的中期**********

1.1999-2000 年，MySQL AB 公司成立，开发出了著名的 BDB 引擎，MySQL 开始支持事务。

2.2000 年，MySQL 公布源代码，而且采用 GPL 协议，进入开源界。

3.2000 年 4 月，MySQL 整理了已有的引擎，命名为 MyISAM。

4.2001 年的 4.0 版本中，集成了 InnoDB 引擎。

5.2004 年 10 月，发布了经典了 4.1 版本。

6.2005 年 10 月，发布了一个里程碑式的版本，MySQL5.0，加入了游标、触发器、视图、事务等支持，从此迈向了高性能数据库的殿堂。

**********收购**********

1.2008 年 1 月 16 日，MySQL 被 SUN 公司收购。

2.2009 年 4 月 20 日，SUN 被 Oracle 公司收购。

3.2010 年发布了 MySQL5.5.

**********MySQL 的优点**********

1.MySQL 的优点主要就是体积小、速度快、性能卓越。

2.MySQL 源代码开放，即开源，可进行适当调整。

3.在 LAMP 与 LNMP 家族中的其他成员配合默契。

**********mysql 的竞争对手**********

1.当然很多对手向来是重量级的对手：Oracle、DB2、SQL Server 等等。

2.还有开源数据库也是竞争对手，比如 PostgreSQL。

3.还有其他类型的数据库也可能会瓜分 mysql 的市场：MongoDB 等等。

**********我的建议**********

1.我们学习一门数据库还是有必要的，至少我这么看。

2.如果您工作中和数据库八竿子打不着，那就算啦，毕竟意义不大。

第二节：下载与安装

**********说明**********

1.由于我个人电脑上一直在用 wamp，所以说就找了一台没有任何编程环境的给大家演示如何安装 mysql。

2.我后面的教程并不是刚安装的这个来给大家做的，不过影响不大，都差不多。

**********下载**********

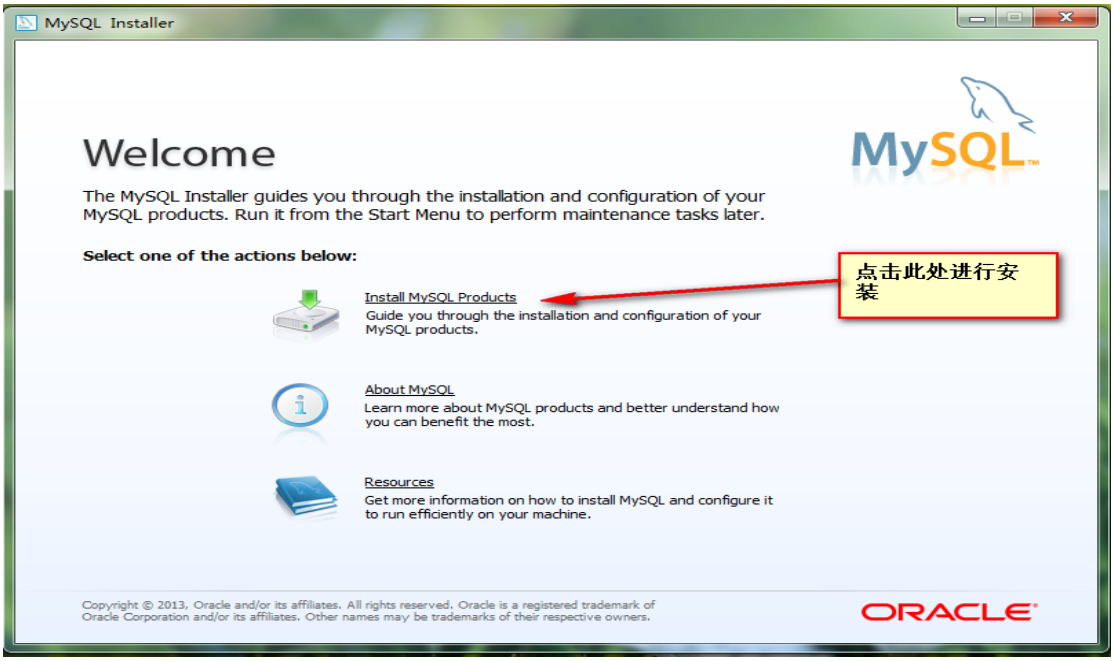
1.百度搜索“mysql 下载”就可以看到下面界面：

2.这里选了下了一个安装软件，然后它自动下载安装：

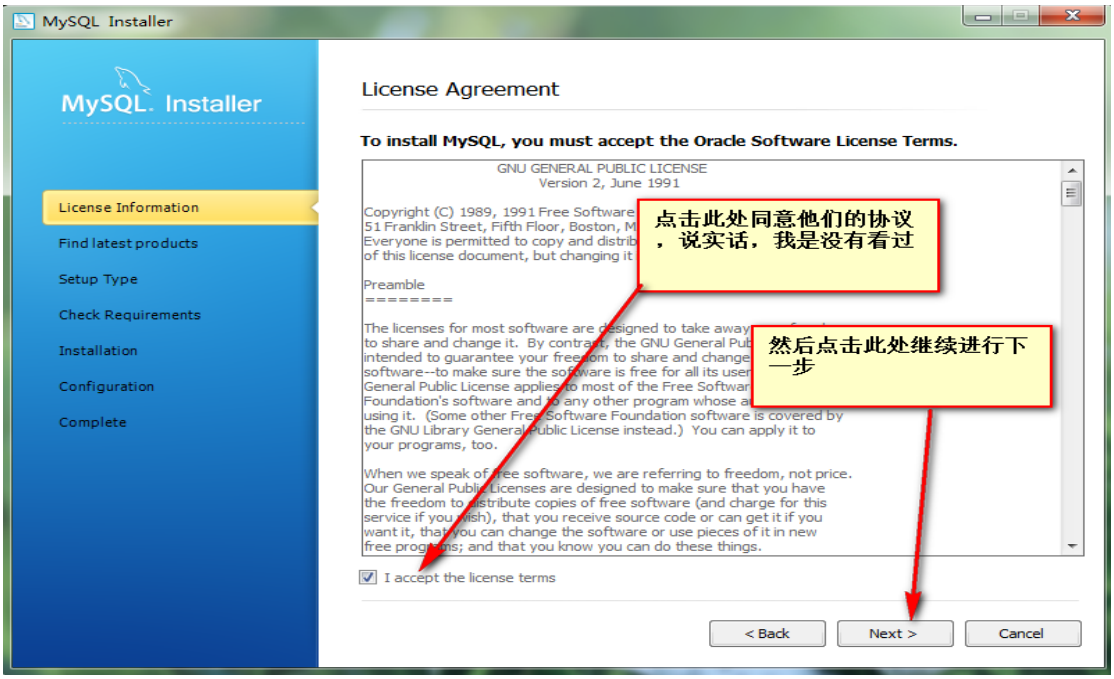
3.下载完毕之后它会自动运行安装程序，咱们也可以双击这个下载下来的 msi 文件来安装。

**********安装**********

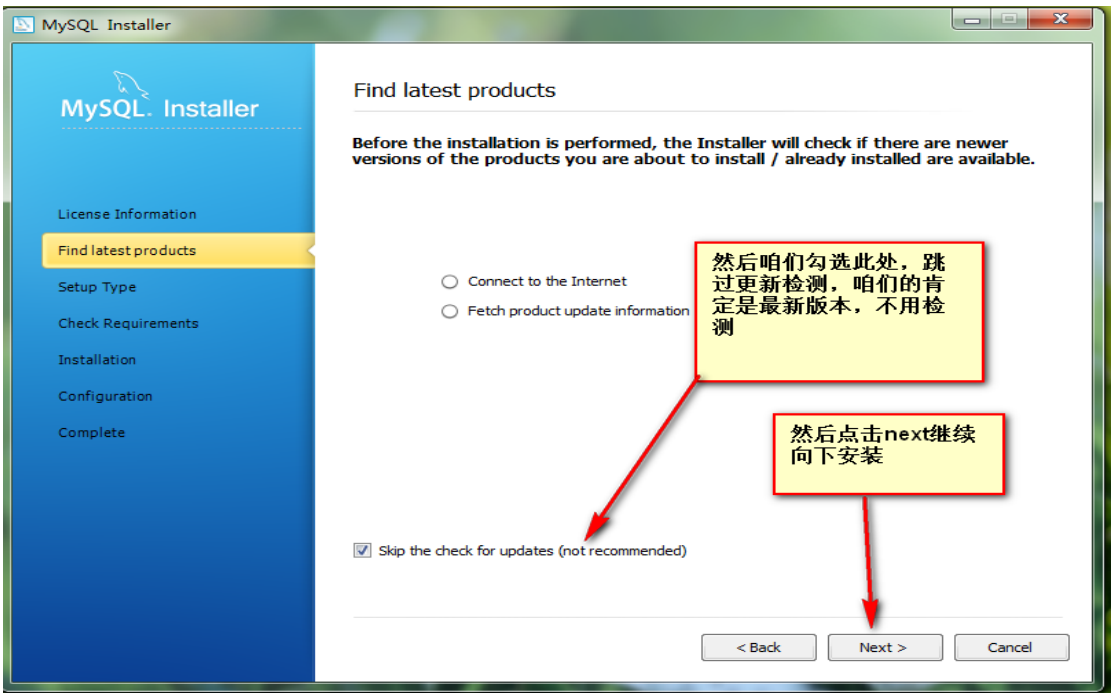
1.首先来到它的安装界面：



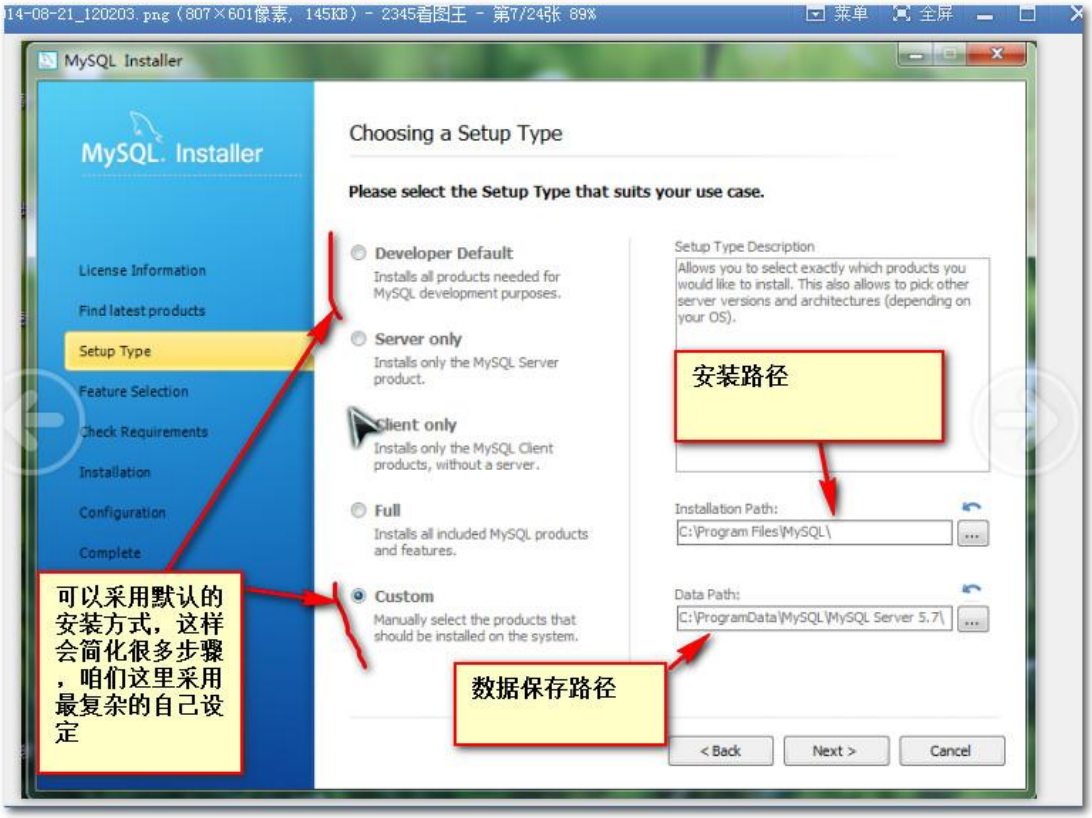
2.然后就需要同意他们的协议：



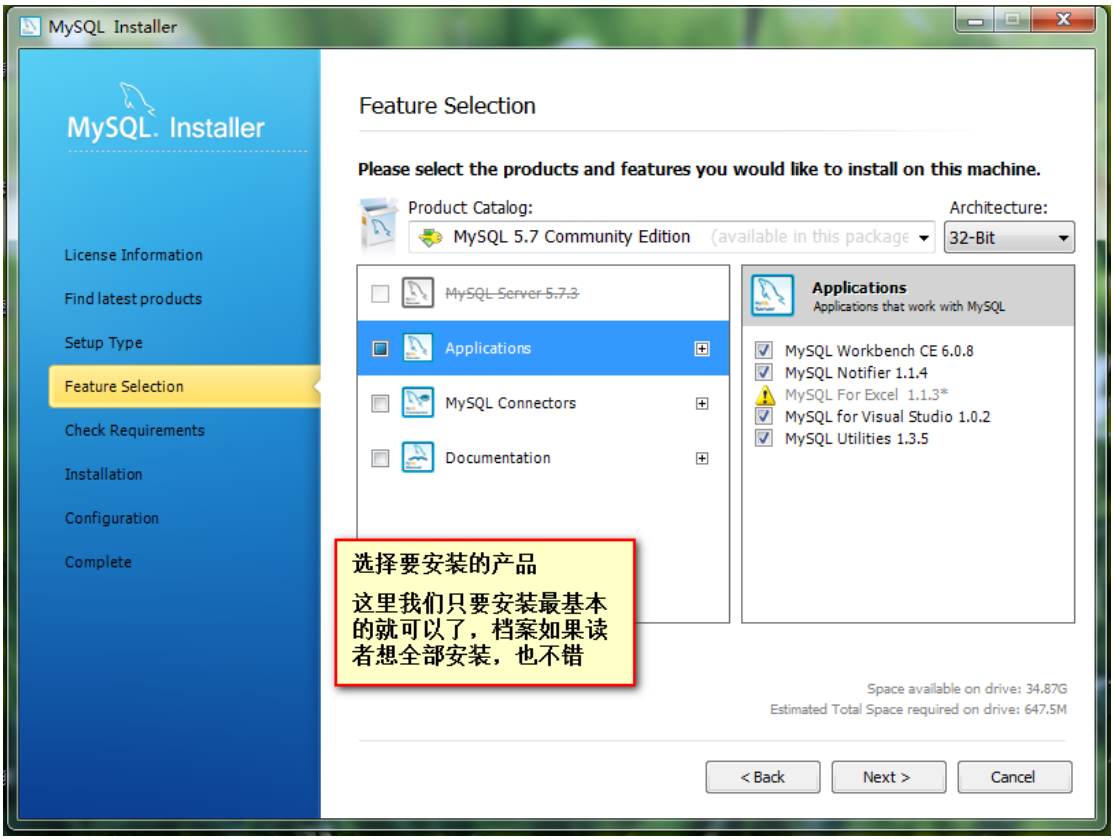
3.然后它会询问是否需要检测更新，咱们点击跳过检测，因为咱们的肯定是最新版：



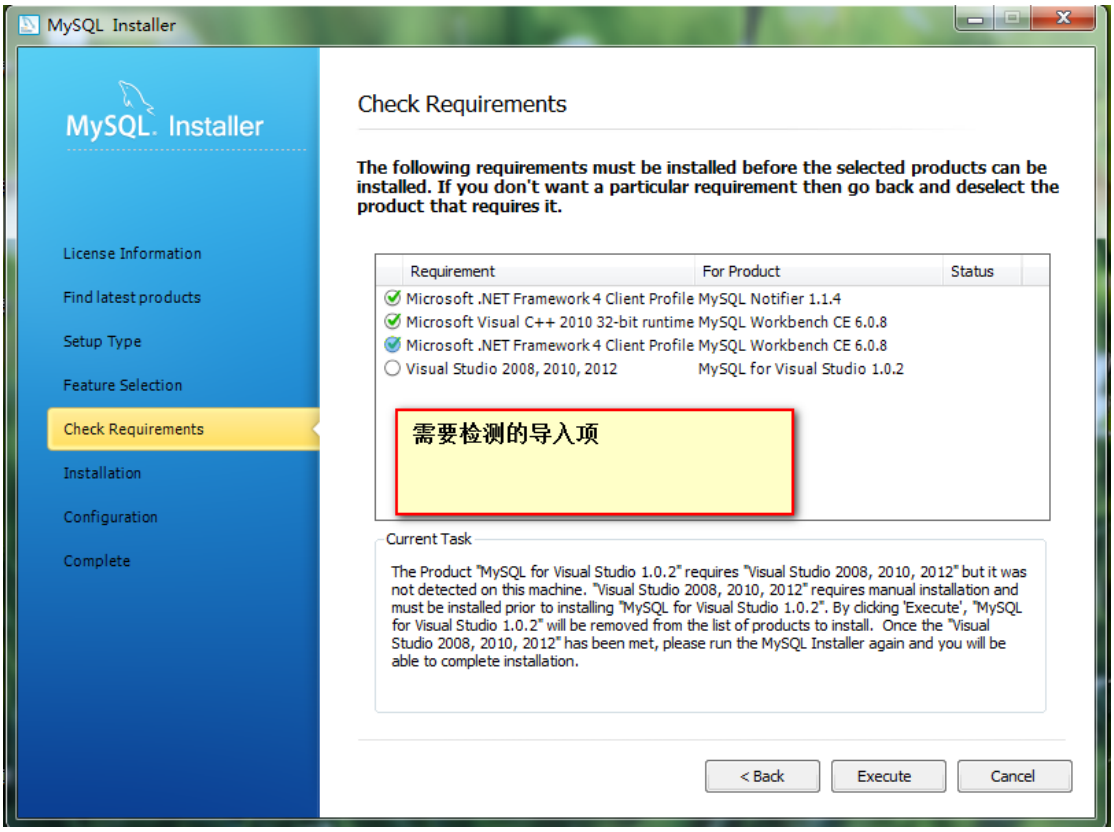
4.然后设置安装的类型：



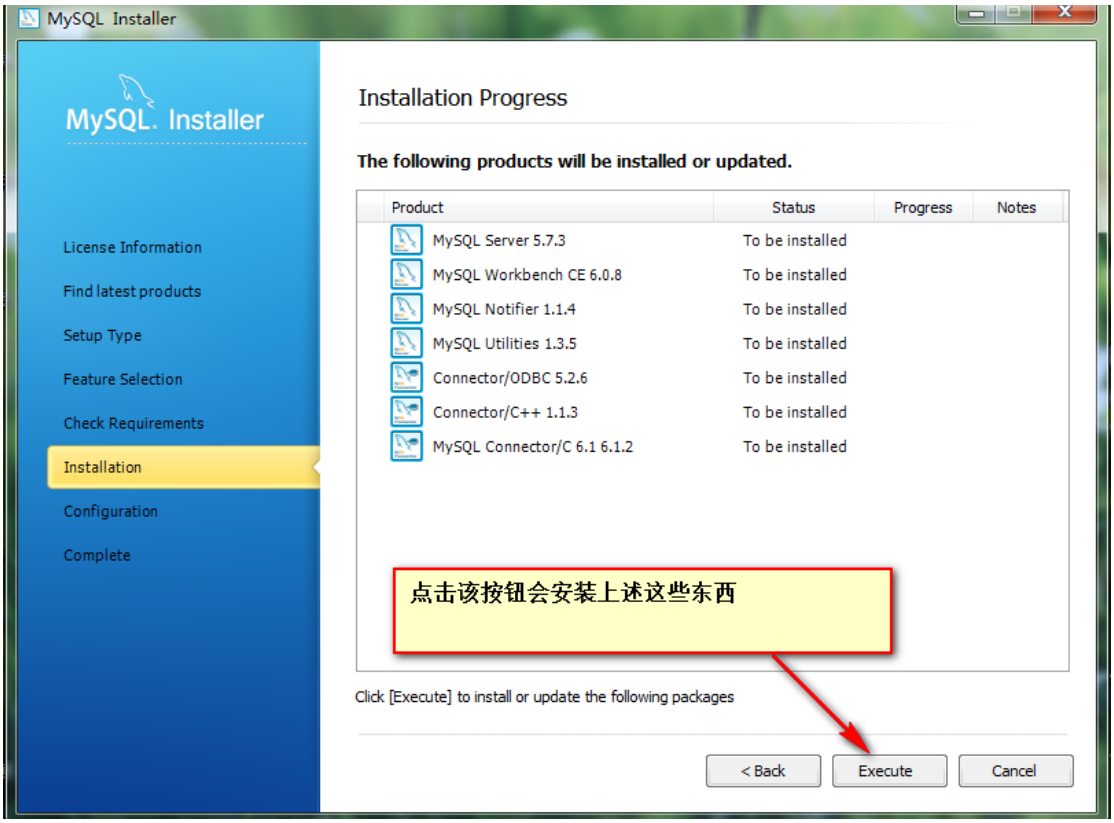
5.然后咱们选择要安装的产品：



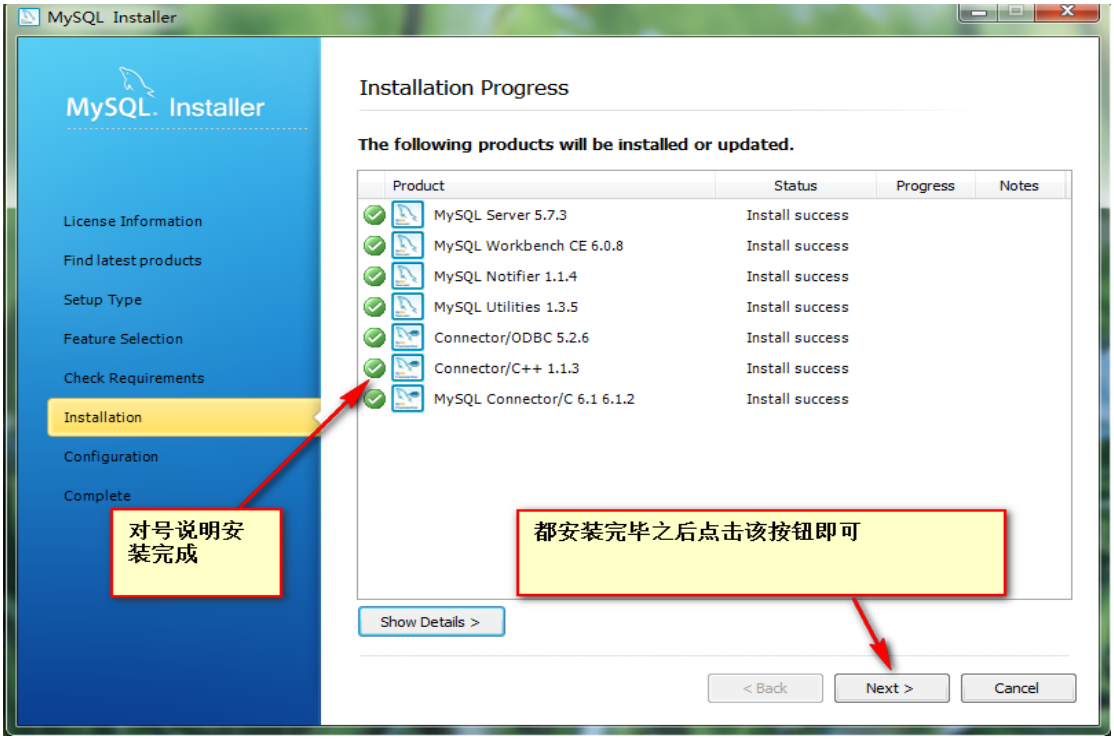
6.然后检测一下相关导入项：



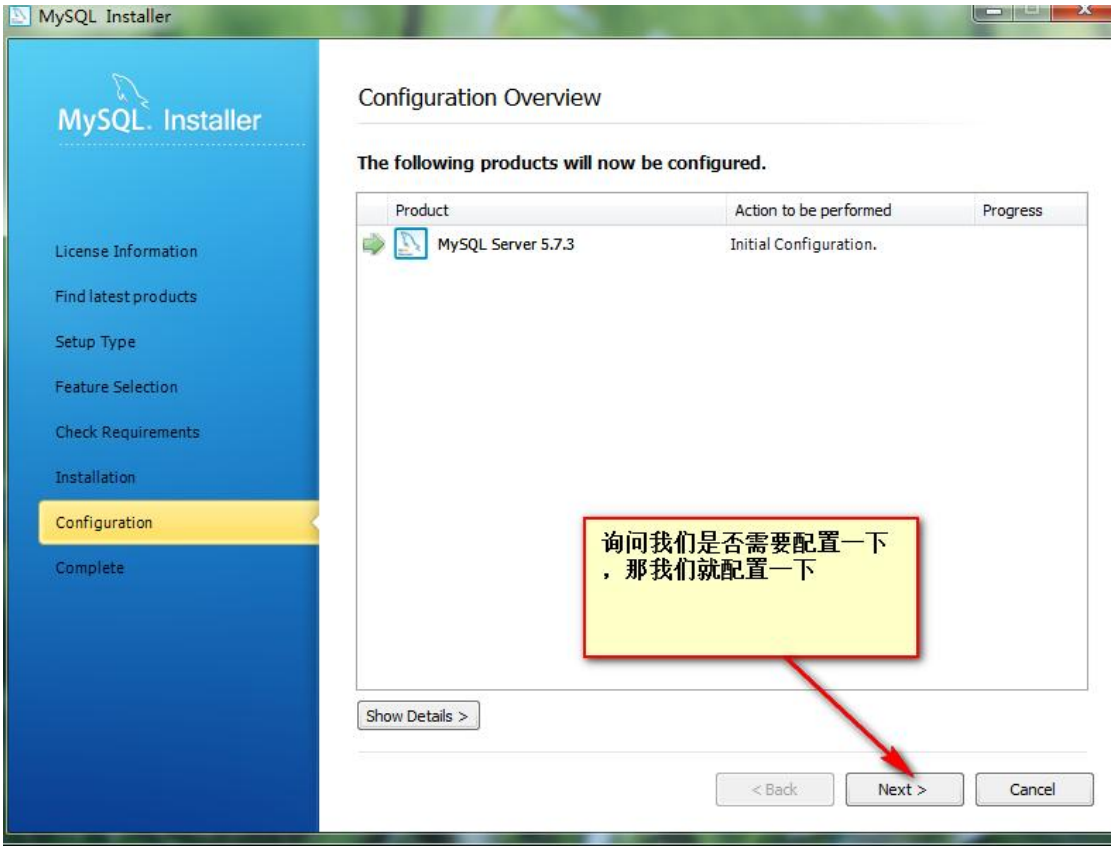
7.然后点击“execute”，然后点击"next"，过一小会儿就来到如下界面：



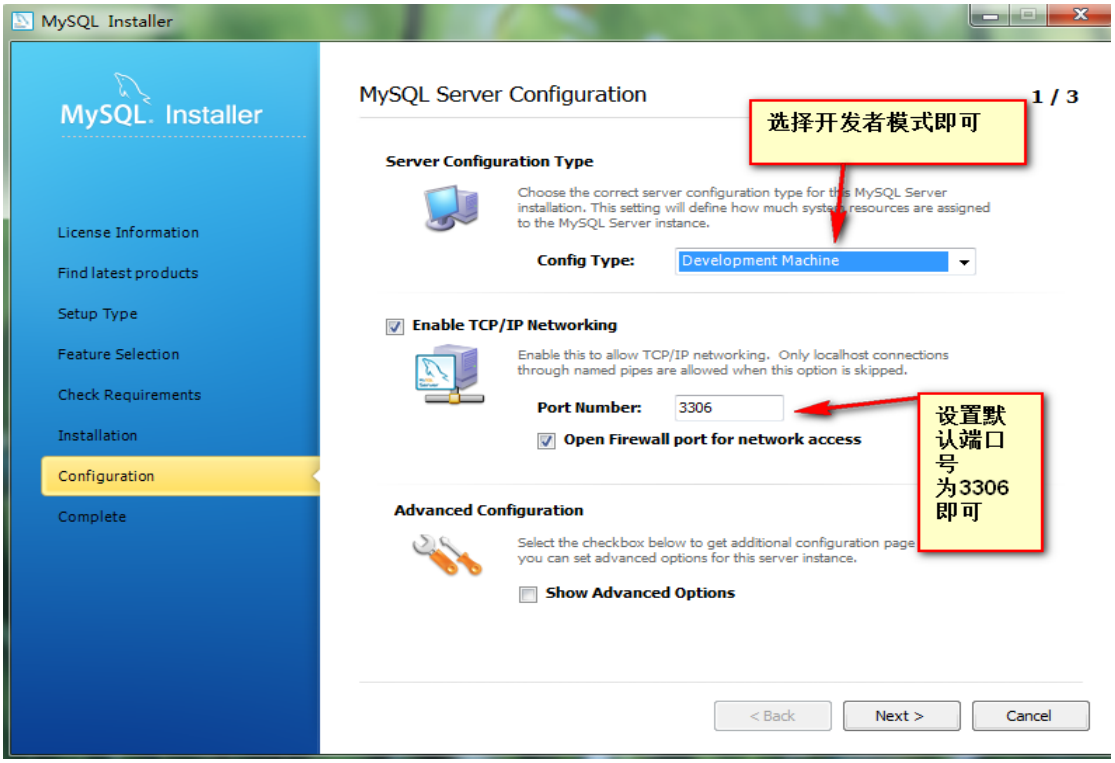
8.上面是我们需要安装的东西，点击“execute”进行安装：



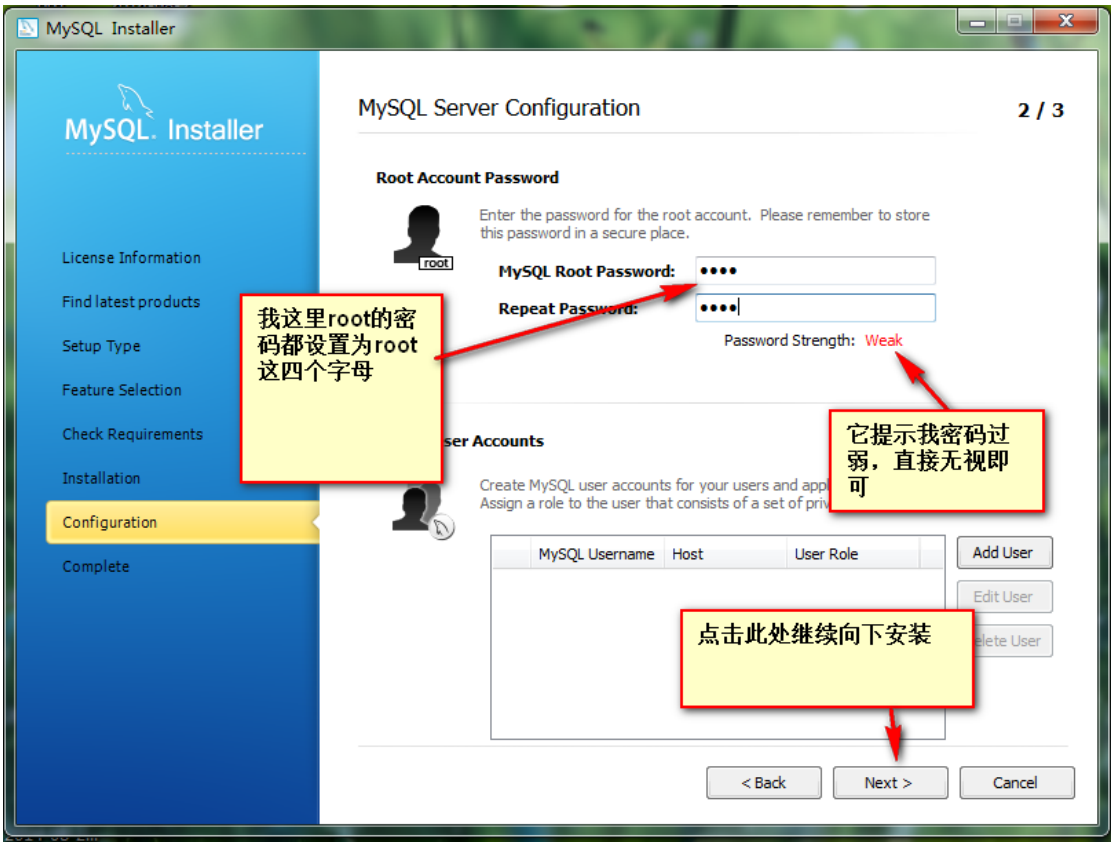
9.然后会询问我们是否需要配置一下，那我们就配置一下：



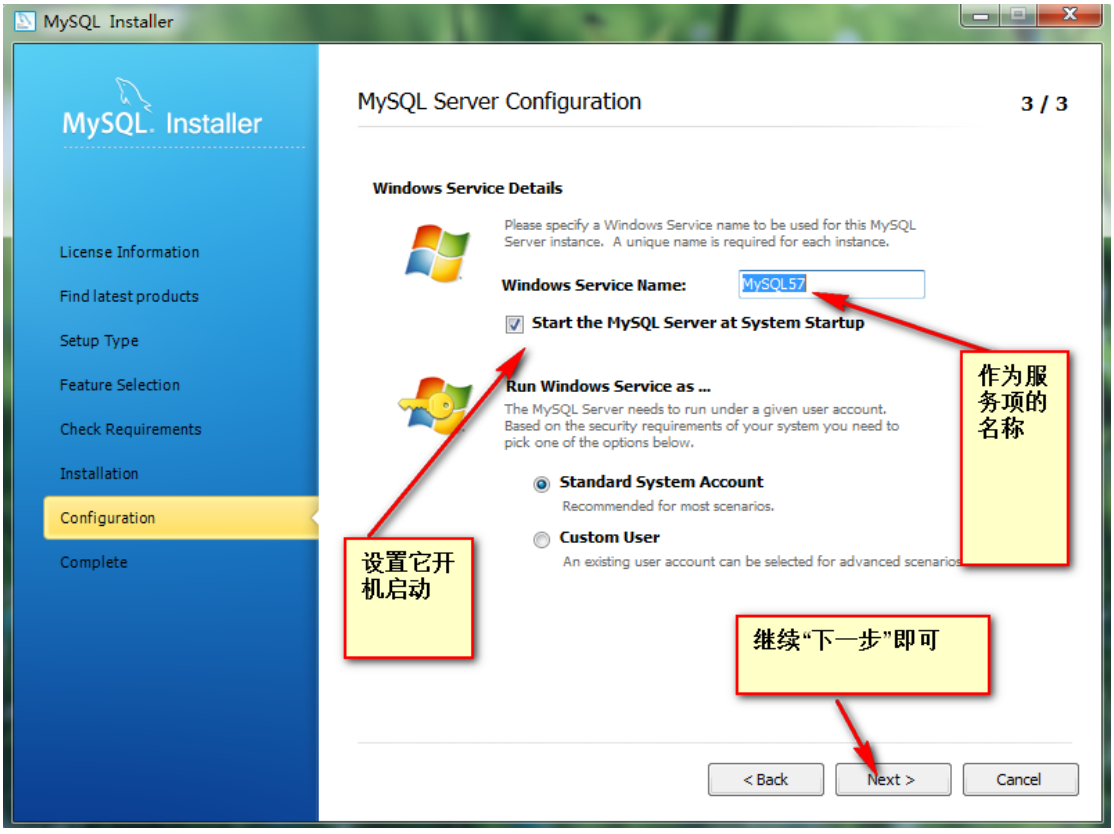
10.然后设置下端口号、机器类型什么的：



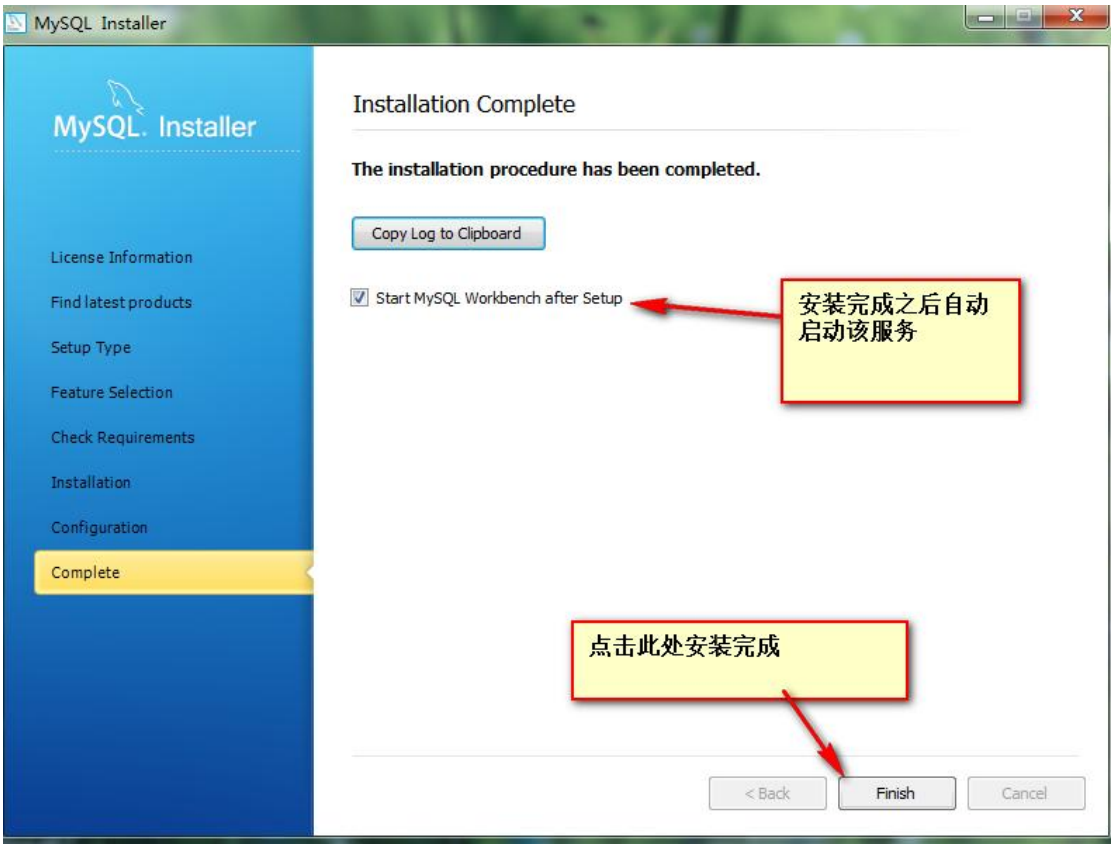
11.然后设置 root 的密码：



12.然后继续下一步即可，我们设置开机启动和它的服务名称：

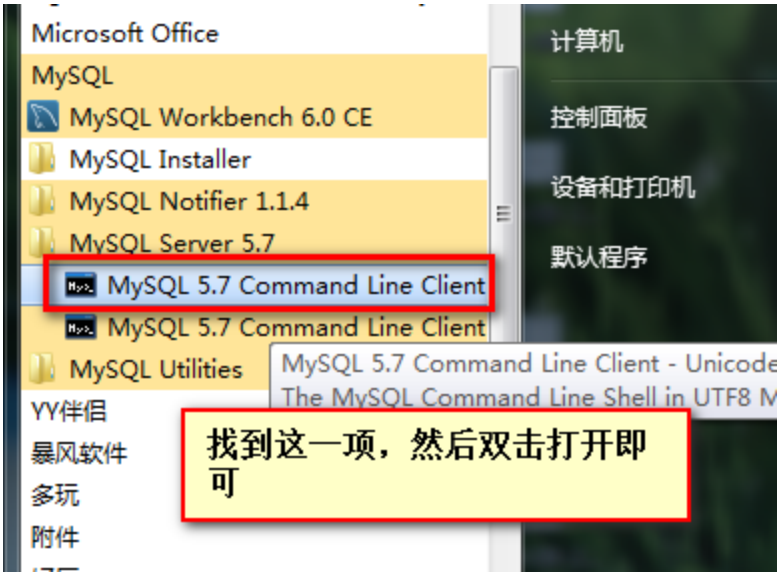


13.此时安装完成：



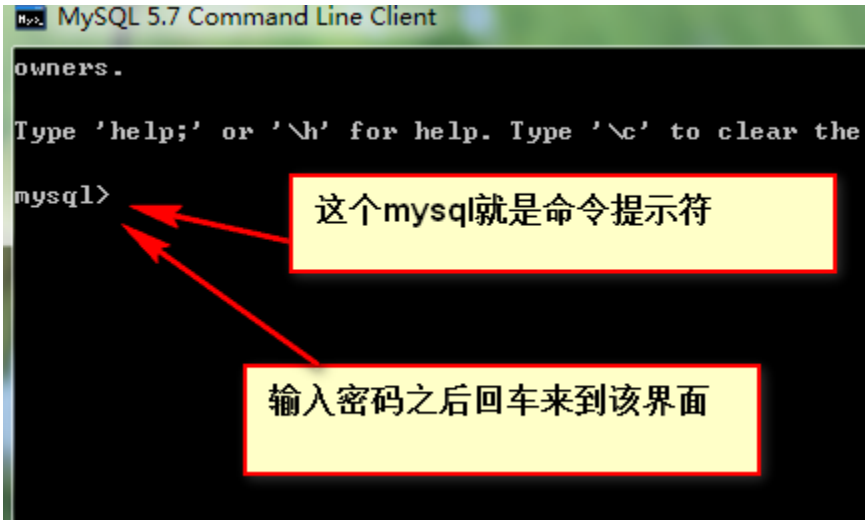
**********使用**********

1.首先打开它的控制台：

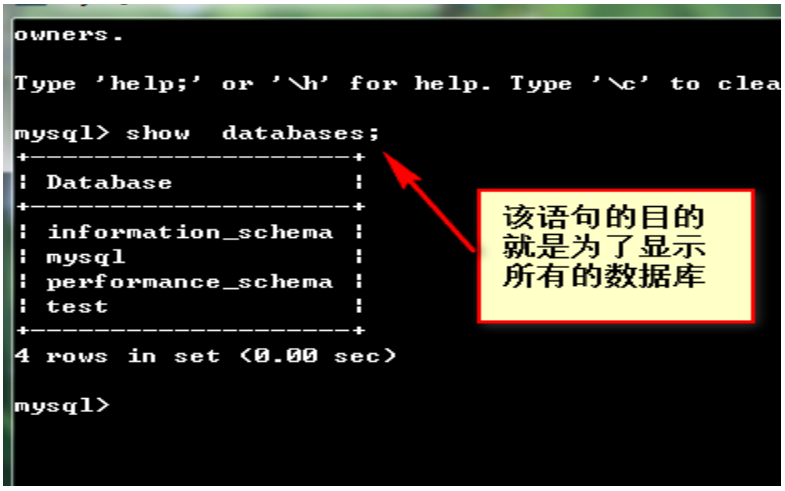


2.然后输入密码，如果读者是跟着我做的话，那么应该输入“root”四个字母即可，否则输入自己的密码：

3.输入密码回车之后来到该界面：



4.然后输入几个英文字母：show databases；然后回车，发现效果如下：



5.截止目前，我们的环境搭建，基本成功了。

**********说明**********

1.为了照顾我之前的 php 教程的读者，我这里使用的 是 wamp 的命令行来进行操作，不过它们的大体功能都是一样的。

2.这个我相信很多 mysql 的使用人群也都会明白。

3.我最早学习数据库的时候是学 Java 的时候，我感觉也有不少的 Java 学习者对 mysql 感兴趣，或者是 python 的学习者，因此，本教程在第一部分过后，不会涉及到任何的编程语言。

4.不过即使您不学习 php，我也建议您看一下第三节，因为它是写给读过我的 php 教程的人群的，但是绝不是仅限于此。

**********整理信心**********

1.前进的路上，您有信心吗？

2.只要星哥在，编程充满爱，一起加油吧。

第三节：写给我的 php 教程读者(wamp)

**********说明**********

1.因为这些教程都是我写的，所以我的教程还是很成体系的，笔记之间不会独立的那么厉害，因此，我这里还是使用 wamp 为例给大家继续讲解。

2.其实呢，我们使用集成开发环境和单个安装是很相似的，这里就不废话了。

**********写给你们**********

1.凡是看过我的 php 教程夏季版的，我都默认你们安装了 wamp，而且我在夏季版中也简要的介绍了下 mysql 的基础知识。

2.但是那里的介绍比较笼统，而这本书的介绍则系统全面的多了。

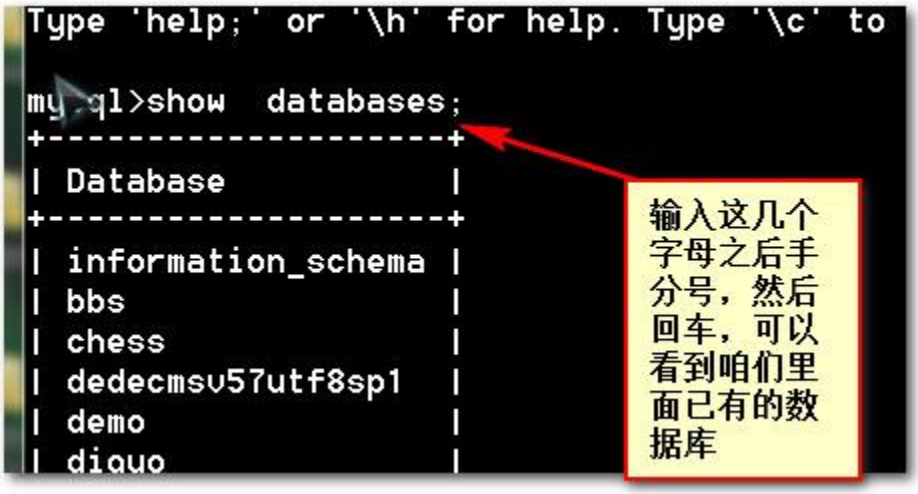
**********如何使用**********

1.首先对其点击左键，在弹出菜单中选择“MySQL”，然后就单击 MySQL 控制台即可：

2.其实 mysql 默认是没有密码的，咱们只需要回车即可：

3.然后进入之后是这样子的界面：

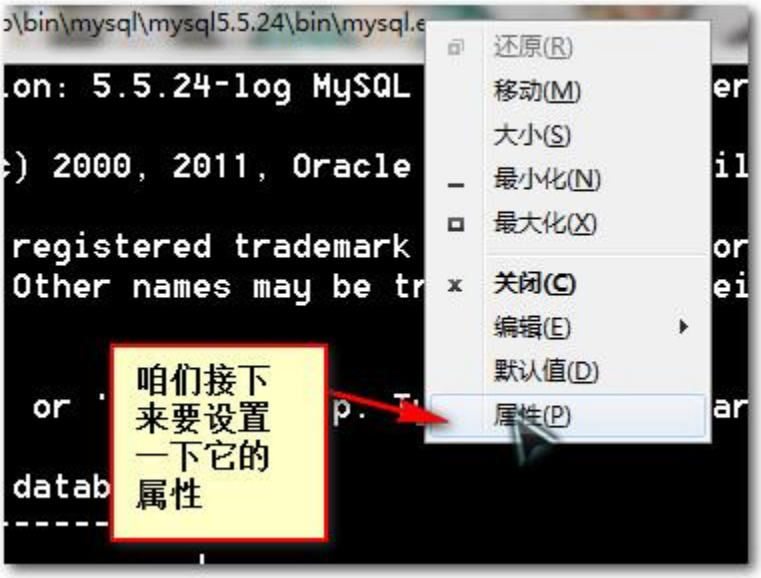
4.然后输入一条语句：



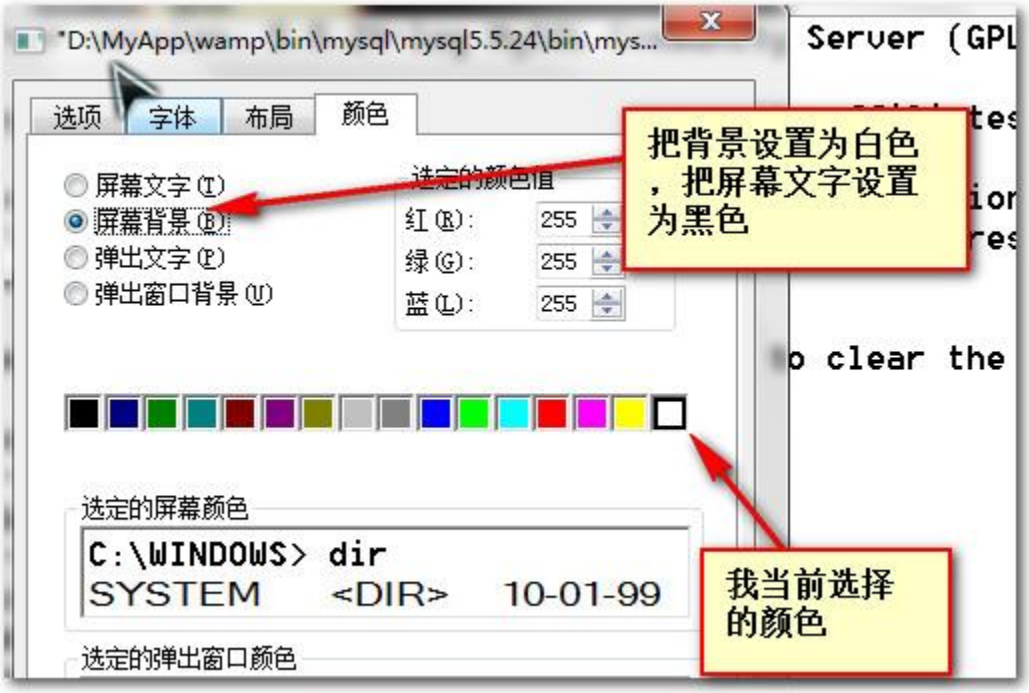
5.我们以后写 sql 语句就在这个里面写了。

**********一些个性化设置**********

1.我不喜欢用它的默认的设置，我们对着标题栏点右键，点击“属性“这一项：

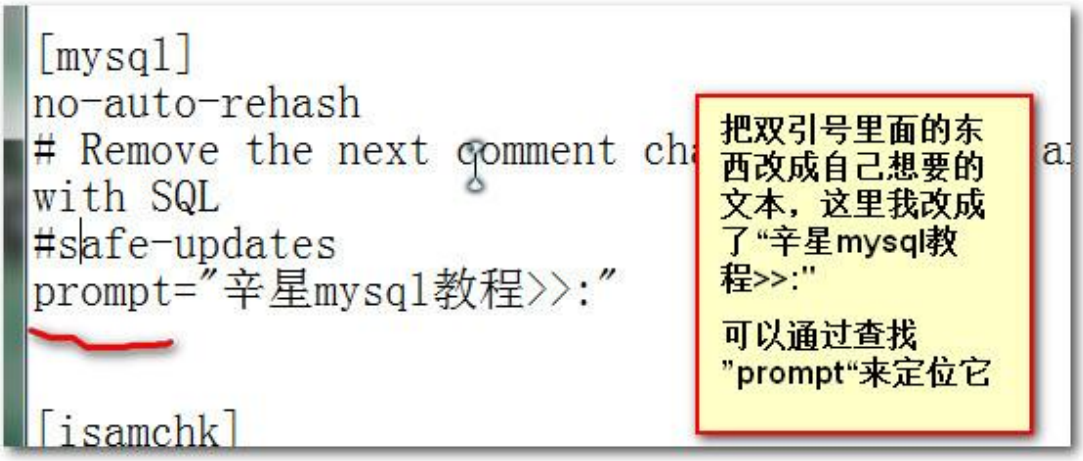


2.然后设置一下颜色：



3.然后在 wamp 中的左键，然后 MySQL，然后 my.ini，我们要修改下其配置文件：

4.使用“替换和查找”，找到修改其 prompt 这一项的内容：



5.这里咱们通过修改这个 prompt，其实它支持显示很多功能的，比如现在登录的用户，比如现在的时间，比如当前所在的数据库等等，大家可以百度下“mysql prompt”来阅读更多内容。

6.然后关闭，保存它，然后我们重新打开 mysql 的控制台，发现当我输入那个空密码后，效果如下：

**********小结**********

1.我是希望你们看了我的 php 教程，能够看这一本教程的，因为它们本身配合的比较紧密，衔接起来很连贯。

2.好啦，不过它是一本数据库教程，不是 php 教程，关于 php，咱们就扯这么多。

第四节：SQL 简介

**********什么是 sql**********

1.sql 是 structed query language 的缩写，即“结构化查询语言”。

2.sql 原名是 sequel，后来由于法律原因，改名。

3.sql 最早可以追溯到 1974 年，源于 IBM 实验室。

4.在 1986 年通过了 ANSI 的数据库委员会成为了标准。

5.1987 年通过 ISO 成为了国际标准。

6.92 年进了一次标准化，99 年进行了一次标准化，03 年进行了一次标准化。

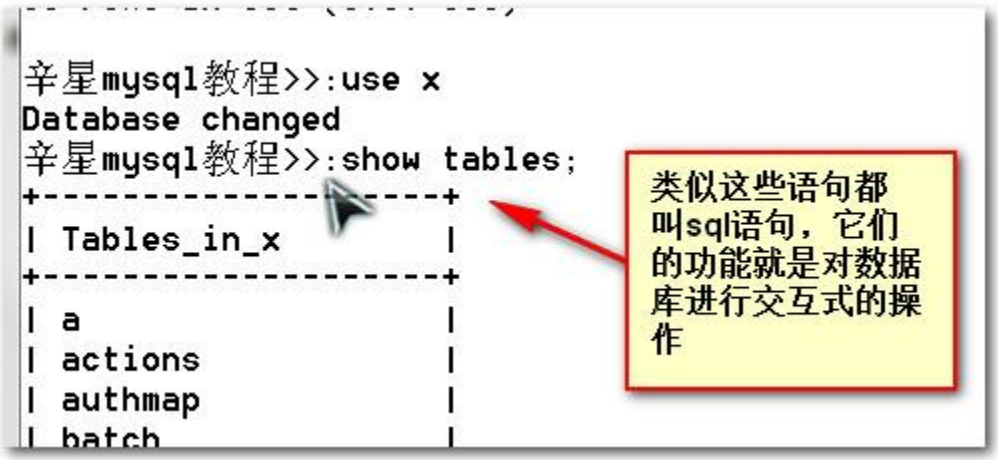
**********mysql**********

1.mysql 对 SQL 标准的支持还是比较完善的，而且进行了一定的扩充。

2.不过 mysql 也有违背 SQL 标准的地方。

**********sql 语句**********

1.本书我不是很想讲那些理论，我们的 sql 语句就是在命令行里面敲的语句：



2.学过 python 或者 shell 的，对这种交互式的模式并不会特别陌生。

3.不过只学习过 c、c++、Java、php 等语言的，可能会这种交互式的编程不是很习惯，不过慢慢习惯就好。

**********学习思路及风格**********

1.我尽量在避免废话，虽然我废话也较多。

2.因为我的教程的风格就是一栏一栏的星号线，然后下面是一二三四等等这样的数字。

3.因此，而且本教程的操作性也绝对足够强，我也希望读者可以跟着我的教程多敲敲代码，这样的进步是最快的。

**********学习方式**********

1.不懂的地方多去看看博客什么的，绝对是一个比较不错的学习方式。

2.最后衷心的祝愿您学习愉快。

第二部分：mysql 增删改查

第一节：数据库和表

**********概念一**********

1.我们的数据库系统和我们的文件系统很相似，因为前者就是根据后者发展起来的。

2.“数据库”就类似“文件夹”，“数据表”就类似“文件”。

3.在文件系统中，我们把信息写入到文件中，然后我们把文件放入到文件夹里。

4.在数据库系统中，我们把数据写入到数据表里面，然后我们把数据表放入到数据库里。

5.它们大致有这样的一个层级关系：

数据库系统：数据→数据表→数据库

文件系统：信息→文件→文件夹

**********概念二**********

1.我们的数据表是如何组织的呢？其实它很类似与我们的 Excel 表格，咱们的表也分为行和列。

2.就像 Excel 中我们通常用列信息表示信息的类型，比如第一列表示公司名，第二列表示创始人姓名，第三列是创始年份等等，我们通常用行信息来表示相同类型信息的不同个体，比如第一行表示微软，第二行表示思科，第三行表示华为，第四行表示腾讯，第五行表示阿里巴巴等等。

3.我们的数据表中也是借鉴了这种组织数据的方式，数据表中的列称为“字段”，每一列表示一类信息，比如第一列表示 id，第二列表示用户名，第三列表示密码。

4.数据表中的行称为“记录”，每一行表示一条记录，比如第一行表示辛星的信息，第二行表示小倩的信息，第三行表示小楠的信息。

**********实例演示**********

1.下面我通过这个截图来给大家展示一下我们的信息是如何在数据库中存储的：

2.对于少量的数据，我们可以这么直观的展示出来，它就是一个表格，数据表嘛。

3.但是对于数据表，我们通常用“一个字段”来代替“一列”这种说法，我们通常用“一条记录”代替“一行”这种说法。

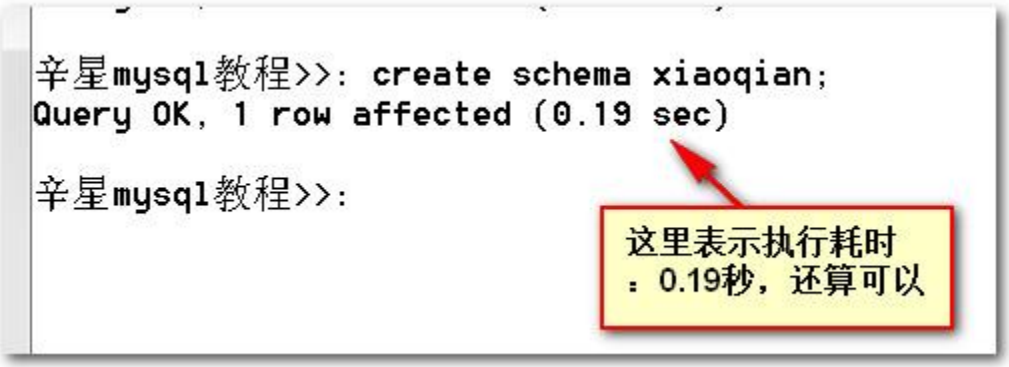
**********数据库的创建**********

1.那么下面我们先来创建一个数据把，我们必须为数据库指定一个名字用来彼此区分，我建议大家取名不要太奇葩，尽量用字母开头，然后跟数字和下划线就可以了。

2.我们来到命令行，敲入如下命令【create database xinguimeng；】然后敲回车，效果如下：

3.此时咱们就创建了一个数据库，名字叫做 xinguimeng，虽然它里面什么都没有，只有一个名字。

4.咱们也可以把上述的 database 换成 schema，也表示创建一个数据库，咱们可以用【create schema xiaoqian；】再来创建一个数据库：

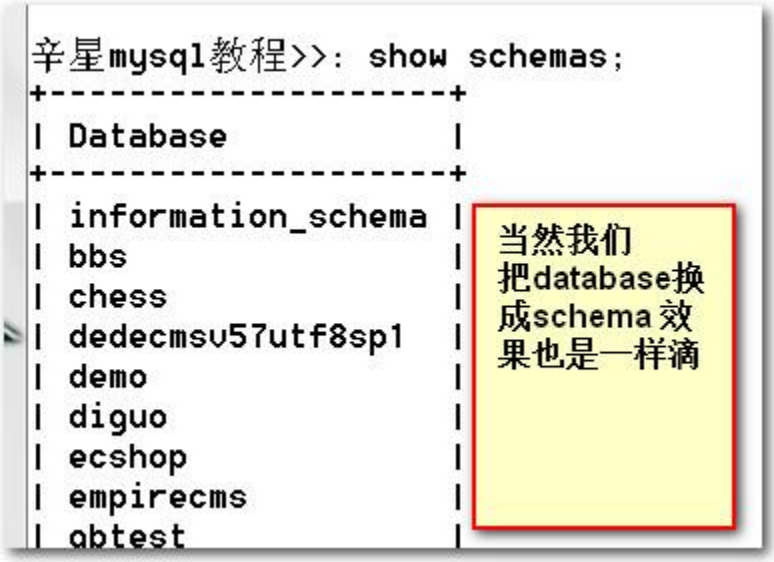


5.那么两者有什么区别吗？没什么区别，就是同名操作。就像汉语中的“头”和“脑袋”是一回事，工程师多用 database，学术派多用 schema。

**********显示和删除数据库**********

1.我们可以用【show databases;】来显示所有的数据库，看我实例：

2.然后我们把 database 换成 schema 也是一样子滴：



3.如果我们想删除一个数据库，可以使用【drop database 数据库名】这种格式，比如我们想干掉 xiaoqian 这个数据库，那么可以使用如下 sql 语句【drop database xiaoqian】：

4.至此，我们已经学会了如何创建、删除、查看数据库，接下来我们研究一下数据表的操作。

**********数据表和数据类型**********

1.创建数据表比创建数据库复杂的多，就像我们使用文件也比使用文件夹复杂的多，文件夹都是一样的，但是文件就五花八门了，有视频和音频文件，还有 word 和 pdf 等 N 多格式。

2.创建数据表的时候我们必须指定这个表的字段，也就是指定这个表的列都是什么，对于行里面存什么信息，在创建数据表的时候不用关心。

3.而要说到表的字段，就必须牵扯数据类型，咱们还没有学数据类型，因此咱们统一使用 int 这个数据类型，下面咱们开始建表。

**********开始创建表**********

1.在建表的时候，咱们的数据表可以有多个字段，每个字段之间用逗号分开，对于一个字段来说，首先跟的是这个字段的字段名，然后是这个字段的类型，比如 id int 就表示这个字段的名字是 id，这个字段的类型是 int。

2.类似数据库的创建，数据表的创建也使用 create 语句，但是创建表的时候就必须使用 table 了，然后我们使用小括号括起来各个字段，比如下面的 sql 语句【create table xinxing (id int,age

int);】就创建了一个数据表，表名为 xinxing，第一个字段为 id，第二个字段为 int。

3.我们开始执行一下该语句：

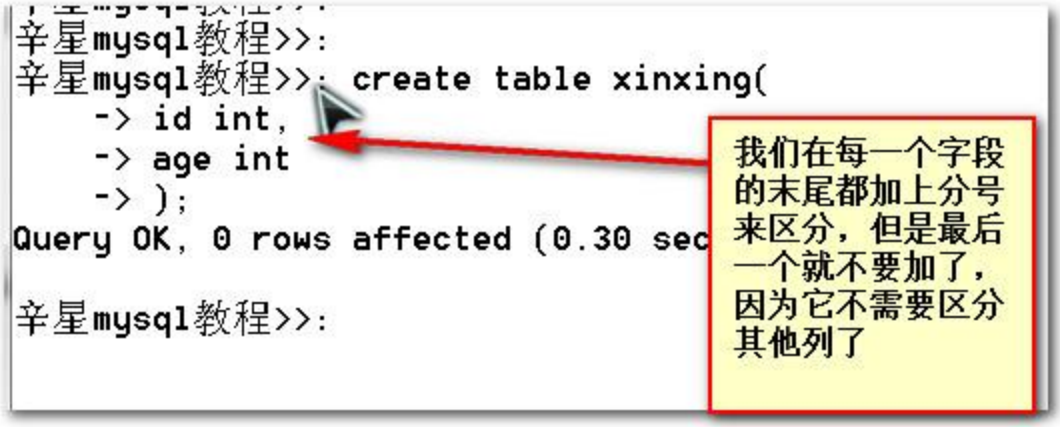
4.然后我们的 mysql 报了一个错误，说：No database selected，也就是还没选定数据库。

5.也就是说，我们不能在不指定一个数据库的时候创建一个数据表，就像我们不能有一个文件是不属于任何文件夹的，那么这个文件的位置该怎么确定呢？

6.我们使用【use 数据库名】这种格式来选择一个数据库，我们选择我们刚才创建的 xinguimeng 数据库，代码：

7.然后我们会看到“Database changed”这样的字样，表示我们当前使用的数据库已经发生了变化。我们的“use xinguimeng”就相当于我们进入了该数据库，我们接下来的操作就是在这个数据库中的。

8.然后我们再次执行这个数据表的创建语句：

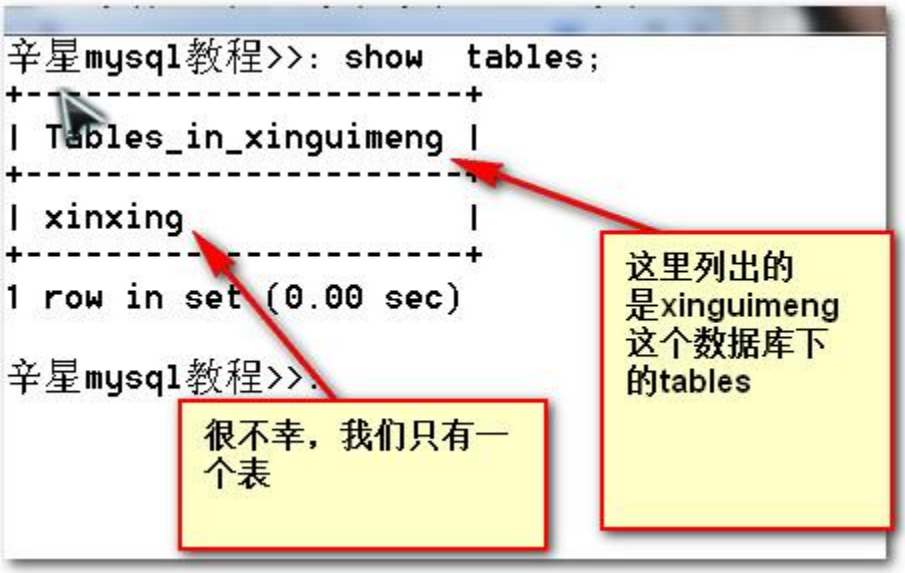


9.这里我们创建了一个表叫做 xinxing，虽然它什么信息都没写，但是它是一个完整的表。

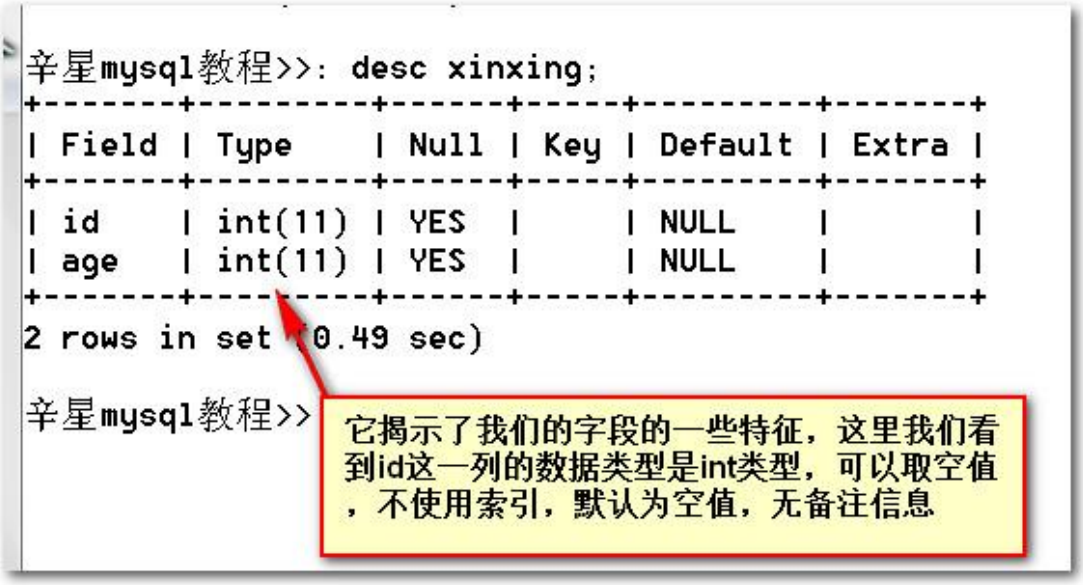
**********表的查看**********

1.表的查看就比较复杂了，因为我们可以从各个角度去看这个表。

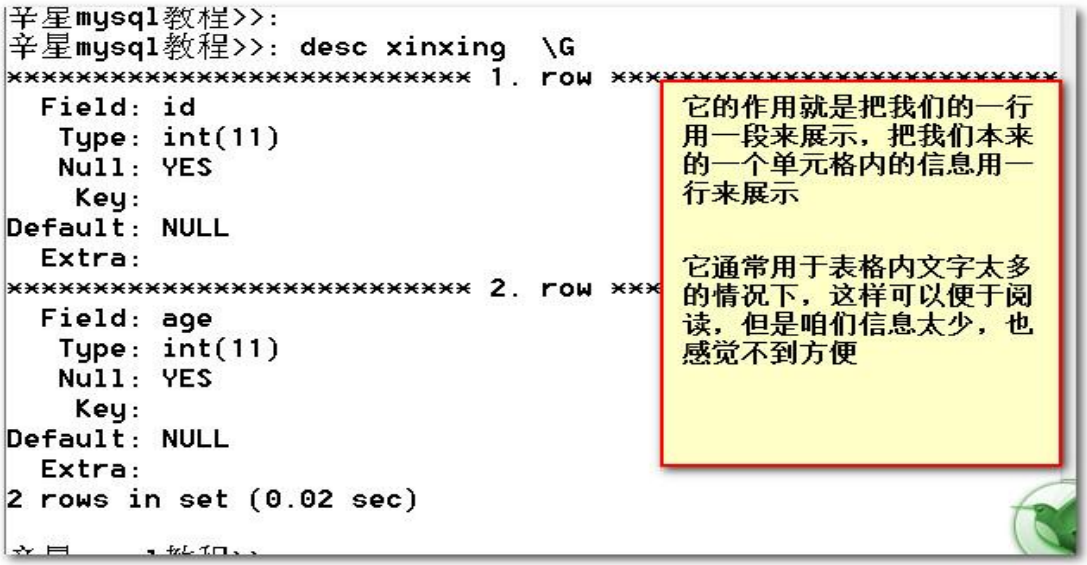
2.我们可以使用【show tables】来查看该数据库下有多少个表，注意这里的 tables 是加了复数的：



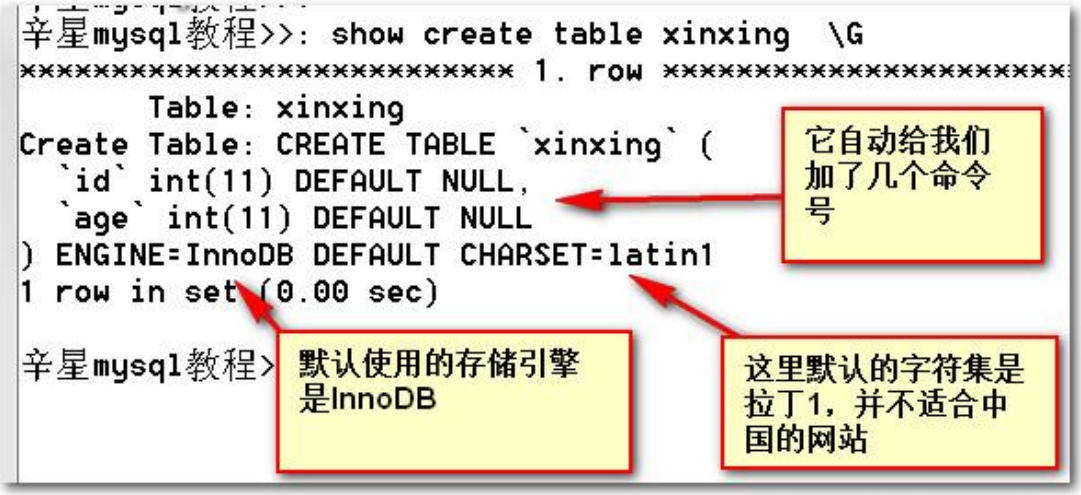
3.我们可以还可以查看这个表的结构，我们使用【desc 表名】或者【describe 表名】来查看一个表的内部结构：



4.以前我们总是用；结尾，这次我们用\G 结尾，大家看一下效果：



5.我们还可以使用 show 语句来查看我们的建表语句，我们使用格式如下【show create table 表名】，别忘记了我们刚才的\G，可以调整格式奥，下面我们看个例子：



6.我们可以看到，这里分为两列，第一列是 Table，就是表名，当然是 xinxing 了，第二列就是 Create Table，就是咱们的建表语句，但是貌似跟咱们写的建表语句不大一样，咱们没有指定一些东西，mysql 会使用默认方式给咱们补全。

7.engine =InnoDB 是说使用的存储引擎，我们在数据库优化的时候再讲，字符集这一本书会讲到。

8.对于那个 DEFAULT NULL 就是“默认为空”的意思，它的学名叫做“列级完整性约束”，可以理解为对这个字段的修饰符。

**********小结**********

1.这一节我们学习了数据库的创建和删除以及查看。

2.然后我们学习了数据表的创建和查看表的信息。

3.大家关键是理解这些概念性的东西，可以参照文件系统去理解。

**********备注**********

1.文件系统中，文件夹内可以有文件夹，但是数据库系统中，数据库中是不允许再次包含数据库的。

2.对于\G 还有各种 sql 语句，还是需要大家的练习的，毕竟，我只能给大家讲清楚，不能陪大家练熟。

第二节；插入数据

**********插入记录**********

1.我记得我们上一节说过，表分为行(记录)和列(字段)，字段是在建表的时候被指定的。

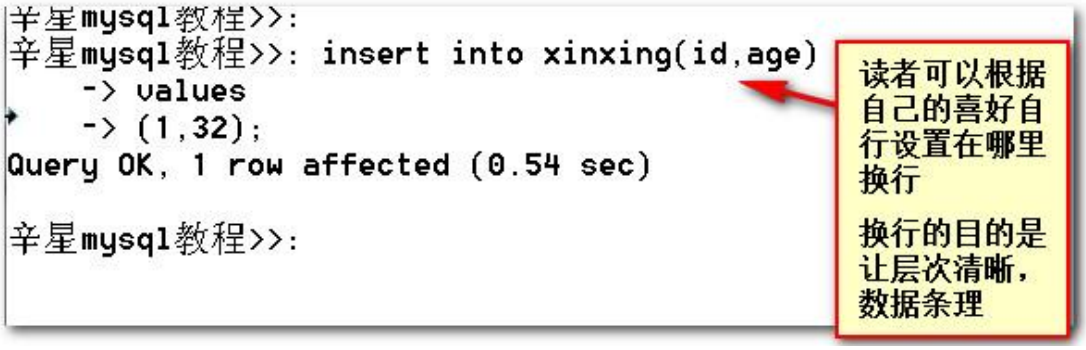
2.我们向里面插入数据的时候就是插入的记录，即我们以“行”为单位进行插入。

3.我们一次性插入一条记录，它通常包含了该条记录对应的各个字段上的取值。

**********insert 语句**********

1.语法：【insert into 表名 (列名 1，列名 2.....列名 n) values (值 1，值 2.....值 n) ;】

2.比如我们上一节建立的一个 xinxing 表，我们向里面添加一些数据，我们根据上面的语法格式来进行如下操作：



**********查询信息**********

1.我们怎么知道我们的信息是否插入了呢？我们必须得亲自看一下才行。

2.我们第三部分再学习详细的查询信息的 select 用法，大家这里先记住一个语法即可【select * from 表名】表示查询一个表的全部信息。

3.比如我们对我们的表使用该 select 语句，看看效果：

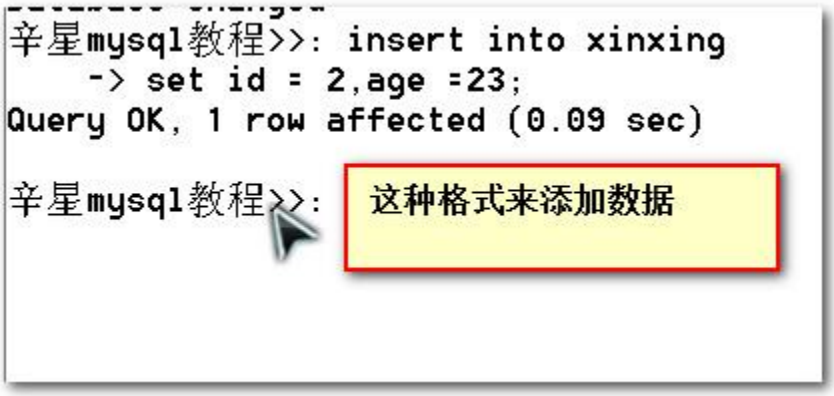
**********另一种插入方式**********

1.下面我们再介绍一种插入方式，使用 set 的方式。

2.这种写法并不多见，甚至有些人本来是学习了它的用法，结果由于长时间不用，又忘记了。

3.语法：【insert into 表名 set 列名 1＝值 1，列名 2=值 2……】；

4.实例：



5.现在我们看一下效果：

**********插入多条数据**********

1.有时候我们想一次性插入多条数据，当然是可以的了。

2.格式：【insert into 表名 (列名 1，列名 2……列名 n) values (值 11，值 12……值 1n),(值 21，值 22,……值 2n).....(值 n1，值 n2,……值 nn) ;】

3.直接看下面示例：

4.我就不用 select 语句给大家展示了，我们的信息肯定是插入了。

**********简写形式**********

1.很多人为了省事，也推出了简写形式，不过使用倒不是很多，但是我们还是有必要了解下，打扫干净盲区。

2.咱们经常写的【insert into xxx】这个 into 是可以省略的，省略这个的还是有一些人的，比如咱们操作如下：

3.有时候咱们也可以不指定表的字段，但是此时我们的(值 1，值 2....值 n)这个顺序必须和原表的字段顺序一样才行，实例代码：

4.如果多了或者少了一个字段，或者数据类型不匹配(咱们后面学习数据类型),那么就会报错：

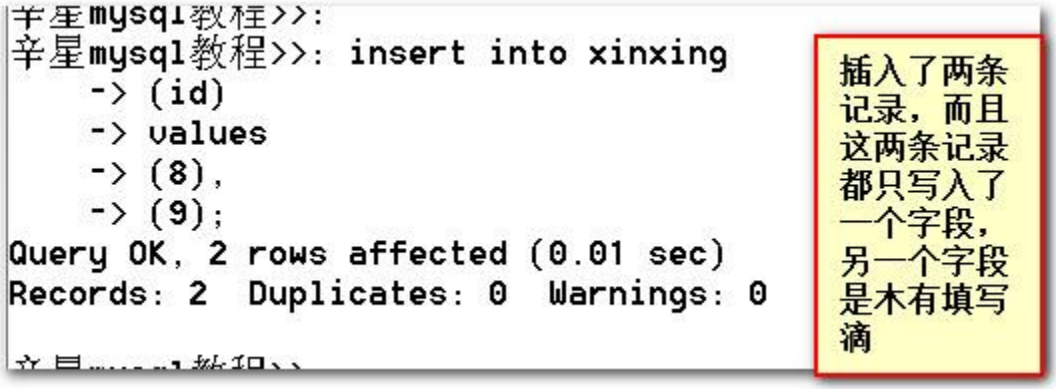
5.比如上面就报了一个错误，说列匹配出错。

**********插入特定字段**********

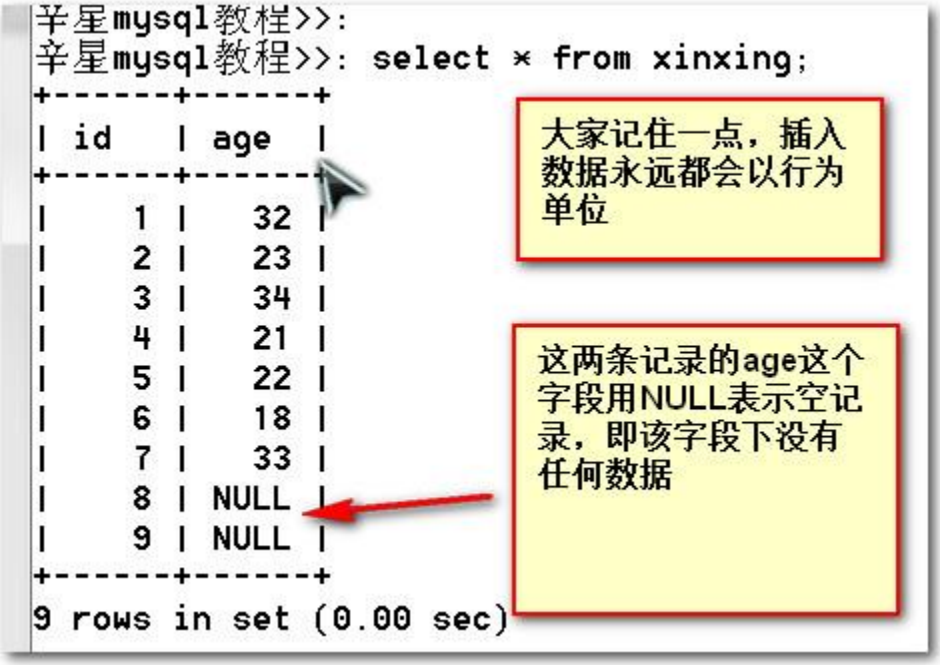
1.有时候我们并不需要插入所有字段，比如我们也无法插入所有字段，比如有些信息我们也不知道。

2.我们可以插入一个表中的部分字段，比如我们的表 xinxing 有两个字段，我们可以只向其插入 id 这个字段(比如我们部门听说来了几个同事，并且员工编号都确定了),但是 age 无法登记(比如他们还没来报道，我们也没有他们的身份信息).

3.那么我们就可以先向数据库录入部分信息：



4.可能有人会说，那此时数据库中的信息是什么样呢？我们来看一下：



**********一点说明**********

1.其实我们插入数据的时候，我们小括号中的类名顺序是可以颠倒的。

2.即【insert into xinxing(id,age) values(10,23);】和【insert into xinxing(age,id) values(23,10);】插入的信息是一样的。

3.我们看一下效果：

4.然后我们看一下表中的数据：

5.此时我们发现表中有两条记录是一样一样的，它并不违反数据库的原则，但是可能新手看它有点别扭，它也确实不好，它违反了第二范式，我们第四本讲数据库理论的时候会给大家讲解的非常清楚。

6.就是说我们需要插入什么数据，就对表中相应的字段进行填充就 ok 了。

第三节：修改数据

**********为什么要修改数据**********

1.有时候我们把数据写入到数据库并不是最终目的，有时候我们必须修改数据。

2.有时候我们根据数据库的使用来分类，有些数据库一旦插入数据，几乎是永不再变(比如存储邮政编码的数据库，比如存储身份证号的数据库)。

3.有些数据库存储了数据就是为了不断的变更的，当然这里的变更往往包括插入数据、修改数据，(甚至是删除数据),它的数据修改也非常频繁(比如大型商城的订单信息的数据库，比如银行记录客户存款的数据库)。

**********我们去设计**********

1.现在想象一下我们是大科学家或者总工程师(不是小兵而是大将),我们自己设计一套协议来规范修改数据库中的数据的操作，或者让我们设计修改数据库中的数据的语法。

2.那么我们应该去怎么设计，此时我们应该考虑到哪些是修改数据库中的数据的要点，大家不妨暂停下来想想，自己尝试着给出一套方案。

3.可能我们一辈子都没有权利做影响这么深远的决定，但是很多小项目我们还是有权去决策的。

**********设计的要点**********

1.要点一就是修改哪个数据库，要点二就是修改哪些字段，要点三就是修改之后的数据是什么。

2.那么修改之前的数据呢？我们不关注，直接当成垃圾扔掉。

3.怎么修改呢？我们可以通过一个条件来控制它的执行。

**********官方给出的方案**********

1.官方讨论这个问题讨论了多久我不知道，他们给出了一套自己的语法格式。

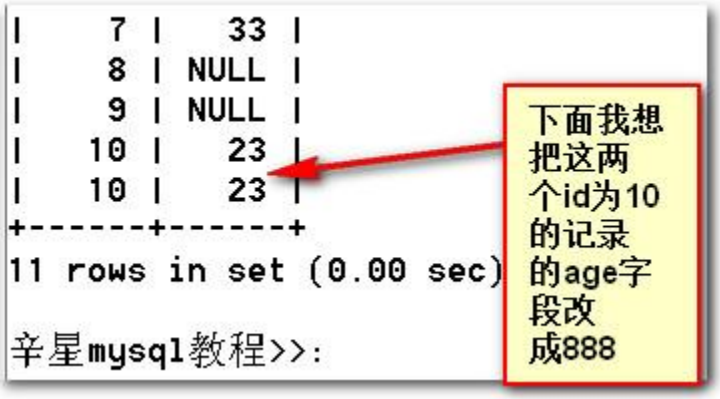
2.语法：【update 表名 set 列1=值1，列2=值2.....列n=值n where 条件】；

3.这里的where条件用来控制我们修改哪些数据，我们后面再详细讲解其语法规范，这里我们只要知道它干什么的即可。

**********实例演示**********

1.还是以咱们的xinxing表为例，还记得那两个id为10的记录吗？咱们把它们的age这个字段修改为888。

2.修改数据之前用select语句看了一下：



3.然后执行修改：

4.然后咱们看一修改后的数据把：

**********where 条件**********

1.where 后面跟一个逻辑表达式，这个逻辑表达式跟我们的 php，Java，python 语法都很像，注意，是像，不是完全一样。

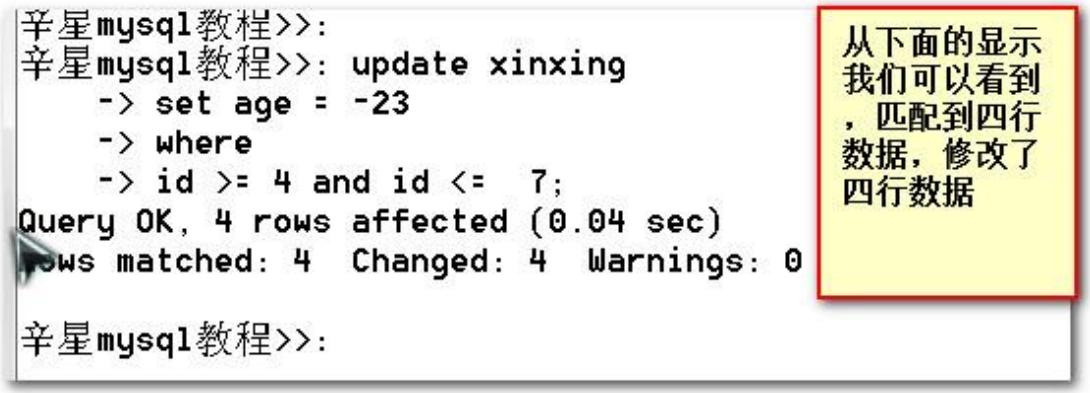
2.逻辑表达式内部可以使用比较符号，比如=表示等于，<表示小于,>表示大于,<=表示大于等于,<>表示不等于等等一些比较运算符。

3.逻辑表达式之间还可以用 and 表示逻辑与，or 表示逻辑或操作。

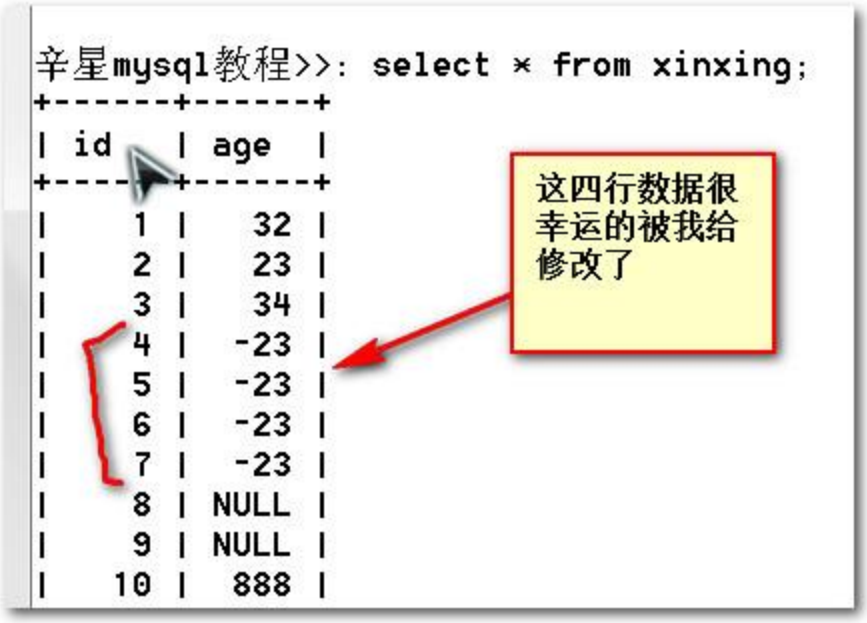
4.比如我们把 id 大于等于 4 小于等于 7 的数据的 age 全部修改为-23，我们先写这个 where 语句里面的内容

【id >= 4 and id <= 7】

那么我们可以使用如下语句来修改数据啦：



5.然后我们使用 select 语句看看效果把：

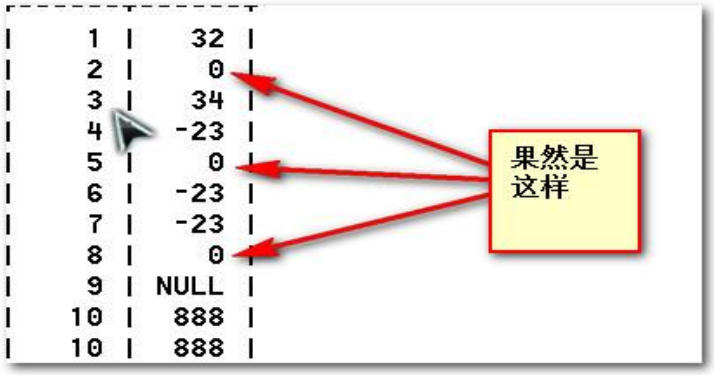


**********in 运算符**********

1.有些人说：那如果我想修改第 2 行，第 5 行和第 8 行的数据呢？是否可以在一个 update 语句中完成呢？

2.答案是可以的，但是，它需要使用到 in 运算符，比如我们把第 2 行，第 5 行和第 8 行的 age 改成 0，看下面操作：

3.然后我们在数据库中查看相应数据，会发现：



**********说明**********

1.可能又会有人说：那如果我想按照 xxx 的方式来修改该怎么写 where 子句呢？

2.我只想说：强劲的 mysql 在 where 子句这里支持超过 25 种运算符，我后面在 select 那一节会给大家讲解比较重要的内容。

3.如果您是新手，跟着我的课程走是最靠谱的，如果您是高手，倒是可以把您对本课程体系的想法告诉我。

**********一点想法**********

1.有没有人想过，如果我们不写 where 子句将会是神马情况呢？

2.其实它会把整个表的数据都给修改了，因为此时默认是全局有效。

3.因此，修改数据的时候必须谨慎再谨慎，毕竟，权力越大，后果越大。

**********小结**********

1.本节我们主要讲解了使用 update 来修改数据。

2.我们用【set 列名 = 值】这种格式来指定修改后的数据。

3.对于 where 表达式，大家现阶段熟练掌握常见的比较运算符和 in 运算符就可以了。

4.辛星，期待您的关注。

第四节：删除数据

**********删除数据**********

1.如果说修改数据就够让人感到心跳了，那么删除数据就完全是战战兢兢，汗如雨下了。

2.可能新手体会不到，当一个黑客篡改或者删除我们的数据之后，给我们带来的损失有可能是致命的。

3.如果我们没有备份，或者数据恢复不及时，那后果就无法想象，只能用两个字形容：“刺激”。

**********假想敌**********

1.试想一下，如果你研究数据库若干时间，变成了超级黑客(菜鸟级黑客就算了)，入侵了腾讯公司的数据库系统。

2.你只需执行一些 delete 或者 drop 命令，就把它的数据库干的一干二净。

3.第二天所有的 qq 号都无法登陆，所有的网页都打不开，这将是多么惨烈的事情.

4.当然腾讯公司肯定有备份，想干掉备份是很难的，因为它们静静的躺在硬盘里，这些硬盘和网络没有任何的物理接触。

**********删除数据的等级**********

1.我喜欢把删除数据分为三个层次，第一种是比较精细的删除数据，第二种是只干掉数据不干掉表的结构，第三种直接把这个表干掉。

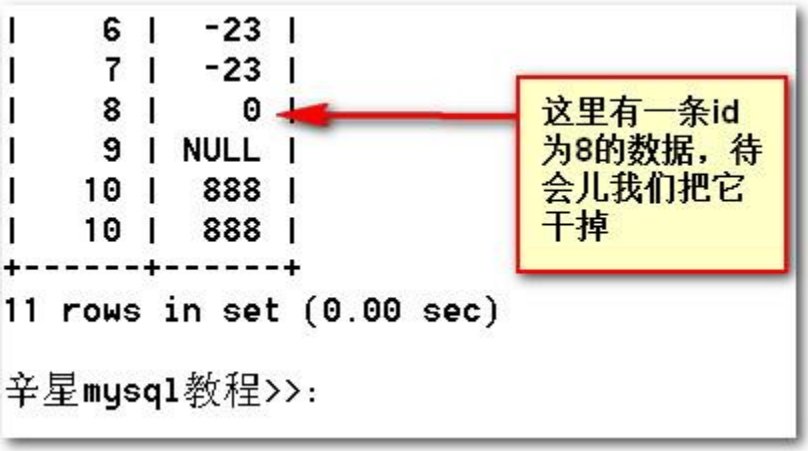
2.可能读者有点晕了，没有关系，我给大家进行实例操作之后，大家会发现：原来如此。

**********精雕细琢 ****delete**********

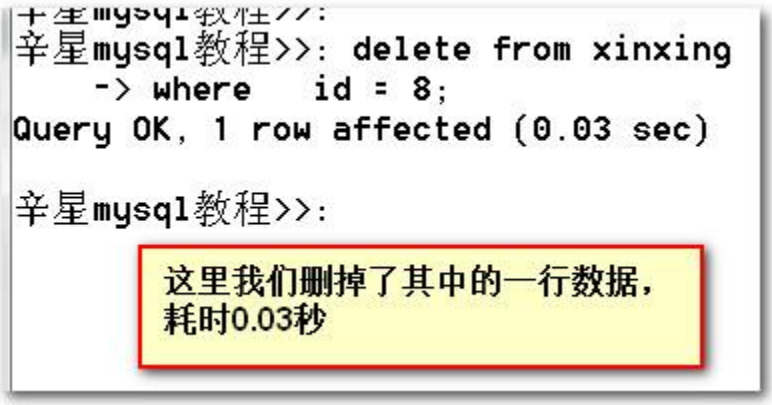
1.这种删除我们使用 detele 这个动词，表示“删除，干掉”。

2.语法格式：【delete from 表名 where 表达式】

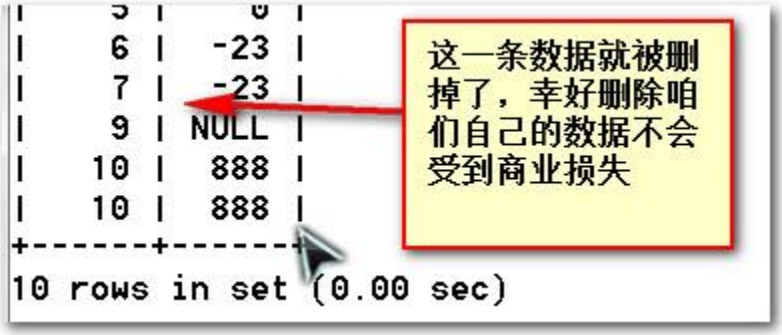
3.我们把 id 为 8 的那一条记录给干掉，干掉之前先截图留念：



4.敲入如下命令：



5.然后我们发现：



6.之所以说它很"精细",是因为它能精确到某一行，这就是比较精细的操作了。

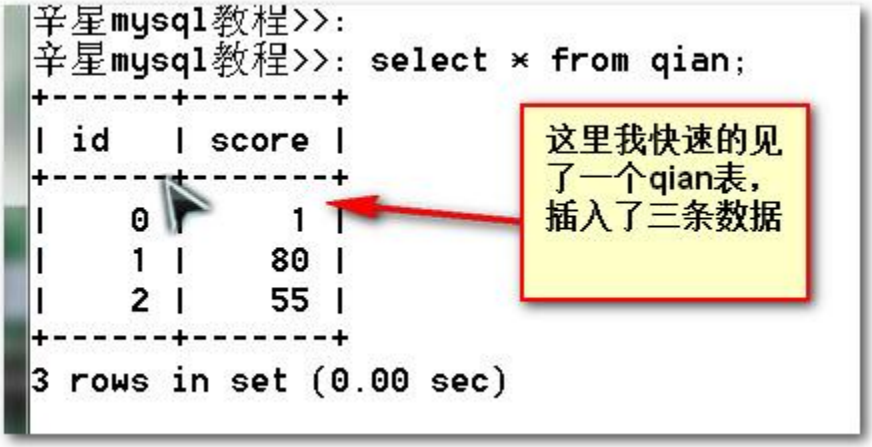
7.当然，它也需要用到 where 子句，这 where 子句，咱们后面会专门介绍。

**********彻底扫荡*********truncate***************

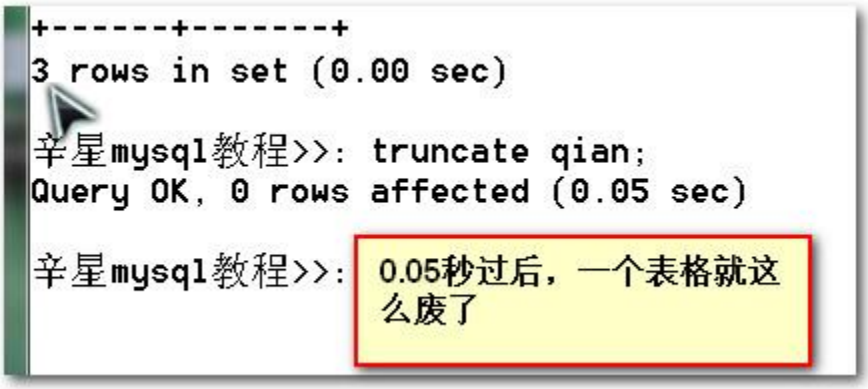
1.我们使用 delete 语句来删除语句的最小单位就是行，是“精雕细琢”的艺术家风格。

2.而我们 truncate 语句则是直接干掉整个数据表的黑客风格。

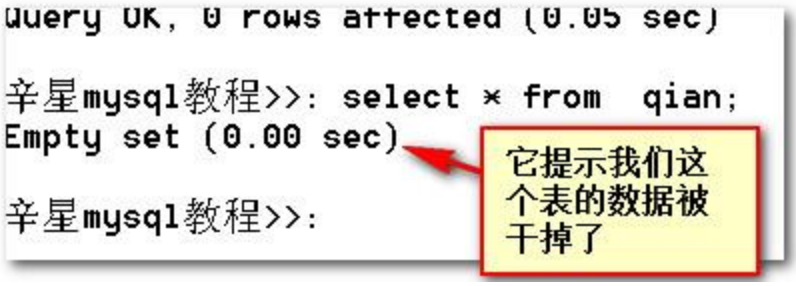
3.我们这里还是再建立一个 qian 表，然后插入几条数据：



4.然后我们 truncate 一下，看下面操作：



5.然后我们再次用【select * from qian】看一下运行效果：



6.我们可以用【desc 表名】这样子来看一下表的结构：

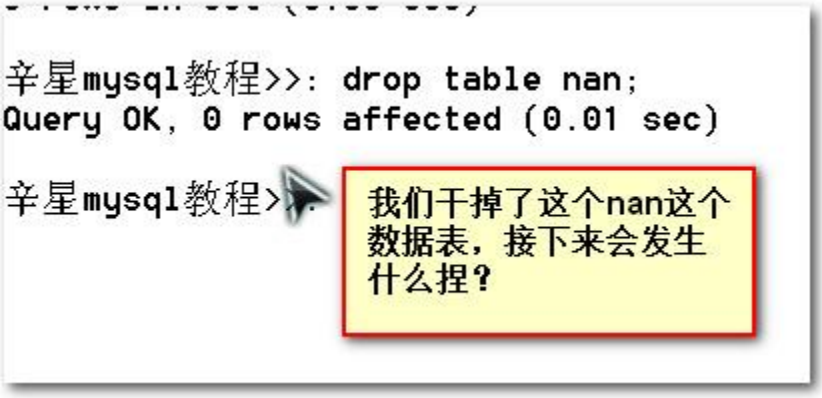
7.也就是说，结构没有任何的改动，我们向里面插入数据依然是可以的。

**********毁尸灭迹******drop**********

1.如果说 truncate 还不够彰显黑客的心狠手辣与无情冷血，那么 drop 足够满足您的要求。

2.比如我建立了一个 nan 表，它的信息如下：

3.那么我们在命令行敲入如下命令：



4.我们执行一下查询：

5.通过 mysql 给我们报的这个错误，我们发现 xinguimeng 数据库下的 nan 这个表已经彻底被干掉了，当然，我们也就无法用【desc nan】来查看它的结构了。

**********表格**********

1.咱们的 mysql 中经常写表格，咱们也写一个表格来总结下这三种操作方式。

2.看下面表格：

动作	删除最小单位	是否保留该表	残暴指数
delete	可精确到每一条记录	是	小打小闹
truncate	干掉所有数据	是	彻底扫荡
drop	干掉所有数据	否	毁尸灭迹

**********说明**********

1.这里说一下 truncate 的运行原理，其实它的操作就是先执行一个 drop 来彻底删除这个表，然后执行一个 create 来建一个同名的表。

2.等我们学完“修饰符”，大家的理解会更加深刻。

**********说明**********

1.其实删除数据是一件非常危险的事情，我们在学习权限那一部分的时候会告诉大家，绝对不能让过多的人有删除数据的权限。

2.删除数据本身就异常危险，即使我们进行了备份，我们误删数据依然会影响系统的正常运行。

3.如果黑客删除了我们的数据，这就更加危险了，它们往往知道哪些数据是绝密数据。

4.抛开安全性不谈，单就是造成的数据碎片，也够我们整理一段时间的，而且在学习数据库优化的时候，我们用到索引，删除数据的同时必须更新索引，性能上也是个问题。

5.因此，真正的商业应用中，删除数据的例子并不是很多见，主要是它的负面作用太大了，而且超危险。

6.大家可能感觉不到危险，因为我们是在操纵自己的数据库，但是，如果你进入的 xxx 大型公司的数据，脑子一热，一个 drop，大家都哭了。

**********解决之道**********

1.很多人会说，我不删除数据库，那么我们很常见的“删除帖子”、“删除留言”都是怎么实现的呢？

2.其实我们可以在创建数据库的时候加一个字段(is_delete)，如果为 1 表示“已删除”，为 0 表示“未删除”，默认取值为 0.

3.当数据显示给用户的时候，如果 is_detele 这个字段为 0，不显示给用户，用户就以为我们执行的是删除，其实执行的是修改。

第五节：修饰符

**********建表语句**********

1.我们在创建数据表那一节里面留下了很多问题木有解决，其实这个问题，我们学完了这本书，也不能彻底解决。

2.我们这一节谈论的是修饰符，所谓修饰符，它的学名叫做“完整性约束”，是为了保证数据库中的数据的完整性而制定的一套规范。

**********主键的概念**********

1.对于一个事物，我们有时候可以用一些特征或者信息来唯一的确定它，在数据库中，也有类似的概念，我们称之为“主键”。

2.以前我们有固定电话，要想唯一的确定一个电话的号码，可以通过“区号-电话号”这种形式，比如我很久以前的一个座机电话就是“022-7324178”，当然，现在不用了，大家不要打骚扰电话。

3.这样我们创建一个描述电话号码的数据库，定义为 phone，用 gid 这个字段表示区号，用 num 这个字段表示电话号码，用 sid 表示身份证号，age 表示年龄。

4.那么这里的 group 和 num 这个组合就构成了一个主键，那么主键有什么特色和特点呢？

**********主键的特征**********

1.主键有如下两大特征：唯一性和排他性。

2.所谓唯一性，即一个表里面只能有一个主键，关系数据库不允许我们有两个主键，它的根本原因是为了防止存储冗余数据。

3.所谓排他性，即我们一个表里面不允许有两条记录有共同的主键，它的根本原因是为了防止我们存储错误数据。

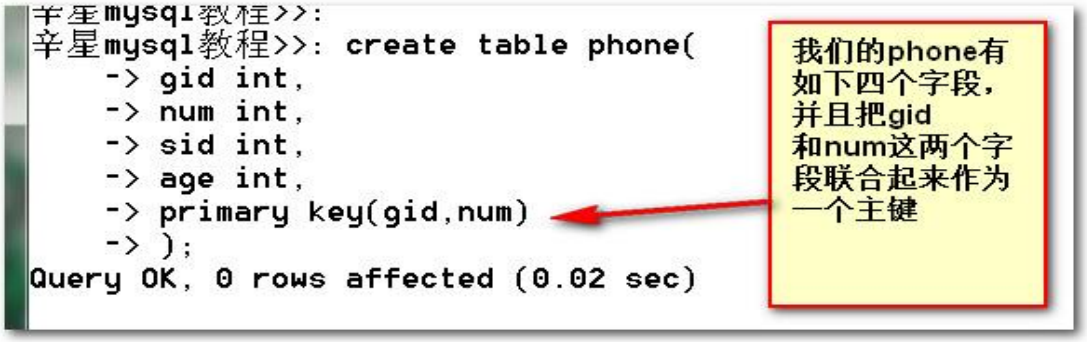
4.就像中华民族用身份证号作为公民身份的唯一标识(唯一性)，而且公民的身份证号必须唯一(排他性)，可见数据库的很多思想已经渗入到我们日常生活中，我们对照着学习，很好理解。

**********主键的使用**********

1.前面讲了一页理论，接下来我们设置一下主键，并且建一个数据库。

2.要声明一个主键，可以在创建数据表的时候就声明，我们可以在写完所有的字段之后，加一个 primary key(字段 1，字段 2.....)这样来表示一个主键。

3.我们执行如下 sql 命令：



4.这样向里面插入数据的时候，我们不可能会出现两条记录的 gid 和 num 是完全一样的，两条记录可以有一样的区号(022 表示天津)，可以有一样的电话号(110 表示匪警),但是绝对不允许两条记录的区号和电话号都一样。

5.我们先向里面插入一条数据：

6.我们继续插入一条数据：

7.我们会发现，我们想插入主键重复的数据的时候根本做不到，因为 mysql 会拒绝执行我们的 sql 语句。

8.这就从很大程度上避免了错误的产生。

**********唯一字段主键**********

1.这里我们使用了两个字段作为一个主键，其实并不常用。我们更多的时候应该用一个字段来作为主键，就像我们的身份证号一样，这样的最大好处就是方便。

2.我们除了可以在声明完所有字段之后使用 primary key 这种方式，还可以在这一列结束的时候加一个修饰符 primary key，表示这个字段是一个主键。

3.咱们通常用一个正整数类型的字段表示主键，而且它通常还有一些其他特性，比如自增(下面会提到,这里先记住使用 auto_increment 修饰即可)、非空(不允许该条记录的其他任何一个字段有值但是它没有值).

4.因此，咱们指定一个正整数的 id 主键的方式是这样滴：

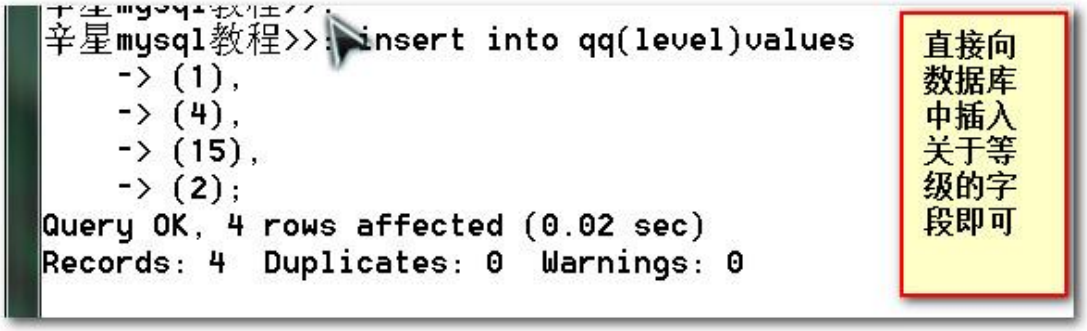
【id int primary key auto_increment】，而且我们向数据库中填充数据的时候，不必指定它的值，它会自动增加，灰常方便。

5.比如我们建立一个 qq 表把，第一个字段是 qq 号，第二个字段是 qq 等级，那么该建表语句：

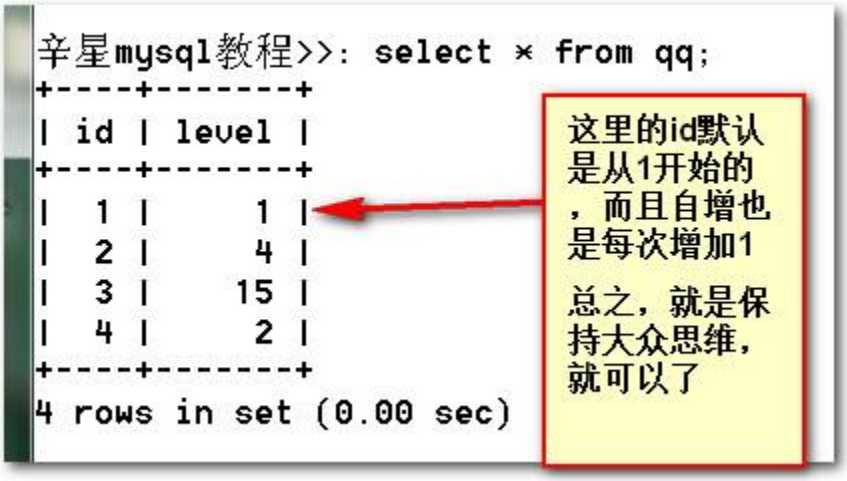
```
辛星mysql教程>>:
辛星mysql教程>>: create table qq(
    -> id int primary key auto_increment,
    -> level int
    -> );
Query OK, 0 rows affected (0.01 sec)

辛星mysql教程>>:
```

id字段表示qq号，level字段表示等级

6.比如我们申请了四个 qq 号，然后等级一次设置为 1,4,15,2，那么对应的 sql 操作：



7.那么我们看一下效果：



**********主键总结**********

1.一个表只能有一个主键，就是所谓的“一山不容二虎，一表不容二键”。

2.主键不要重复，就是所谓的“主键之间相互排斥”。

3.我们在实际的设计中，通常指定一个正整数类型(无符号整数)类型的主键，并且设置为自增。

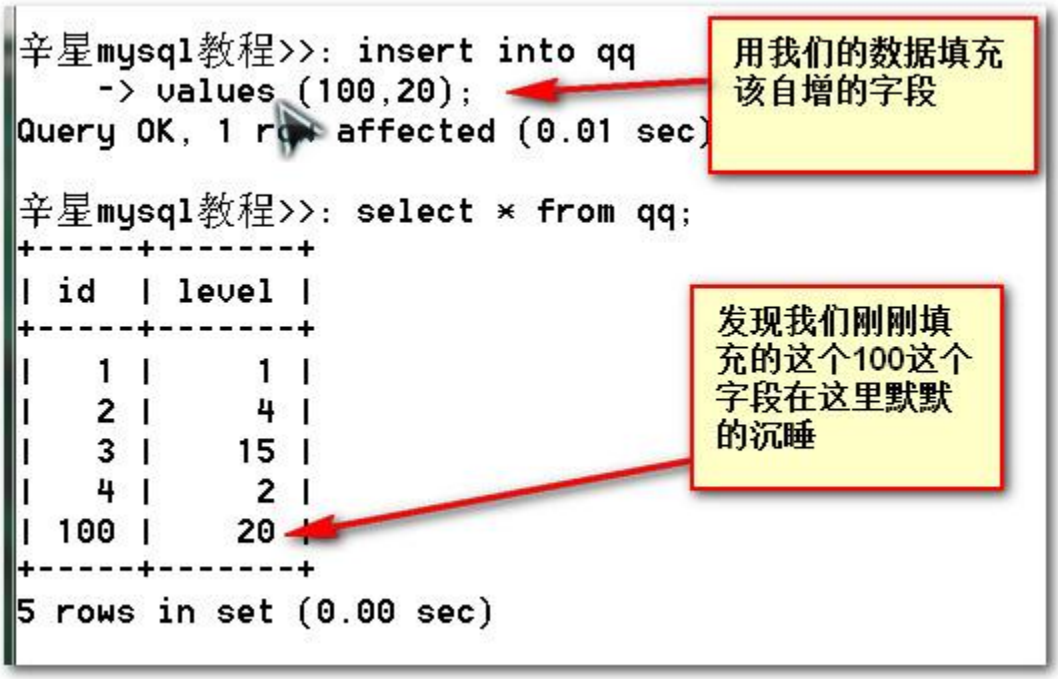
**********自增**********

1.自增只能用在整数上，而且每次插入的下一条记录中的对应字段比上一条记录中的对应字段的值加 1.

2.设置自增之后呢，我们还可以向里面填充值，这个时候用我们的值填充该字段。

3.如果我们没有设置相应的字段，它会进行默认的自增操作，就是比上一条记录的对应字段加 1 放入本条记录的对应字段。

4.我们看一下效果，还是以上一个 qq 表为例把，我们插入数据的时候同时制定 id 和 level，大家看一下效果：



**********非空**********

1.我们可以在一个字段的后面加上一个修饰符 not null 表示非空。

2.如果我们指定了一个字段非空，那么当我们插入一条记录的时候，如果不指定该字段的值，就无法插入。

3.也就是说，此时 mysql 给我们抱一个错误，然后拒绝执行。

**********不可重复**********

1.我们可以在字段的后面使用 unique 来表示该字段的值不能重复。

2.如果我们试图插入该字段重复的值，那么 mysql 会报错并拒绝执行。

**********默认值**********

1.有时候我们可以指定默认值，就是在我们插入了一条记录之后，但是没有向某个字段进行赋值，这时候可以使用默认值的方式让该字段自动设置一个值。

2.我们在字段的后面添加一个【default 默认值】 来表示默认取值为多少。

**********统一演示**********

1.我们创建一个表，它的一些字段拥有上述三种修饰符的其中几种。

2.看下面代码：

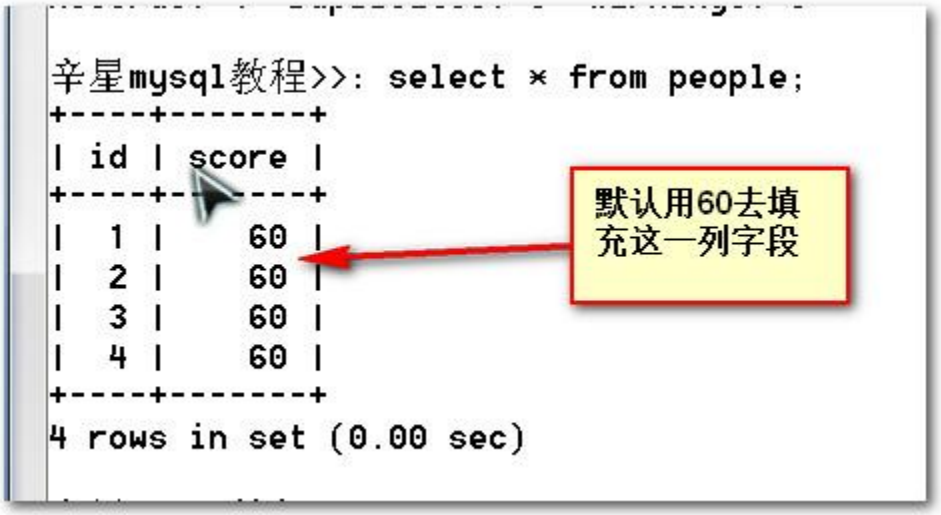
```
辛星mysql教程>>:
辛星mysql教程>>:
辛星mysql教程>>:
辛星mysql教程>>: create table people(
    ->  id int not null unique ,
    ->  score   int not null default 60
    -> );
Query OK, 0 rows affected (0.01 sec)

辛星mysql教程>>:
```

id是非空且不可重复的
score是非空且默认取值为60

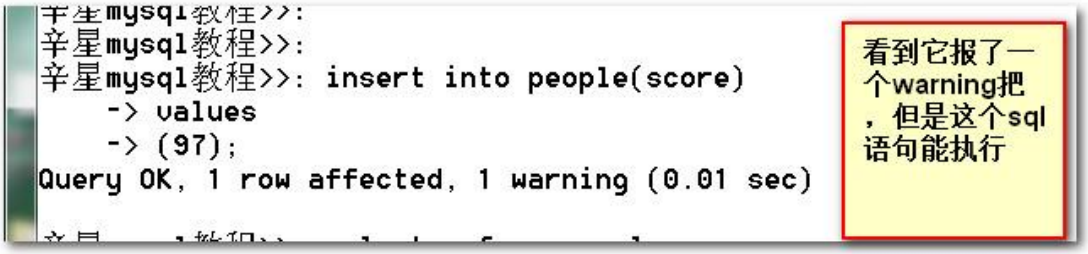
3.那么这里我们创建了一个 people 表，它有两个字段，第一个是 id，非空且不允许重复，第二个是 score，非空默认且默认取值为 60.

4.我们先插入四条数据：

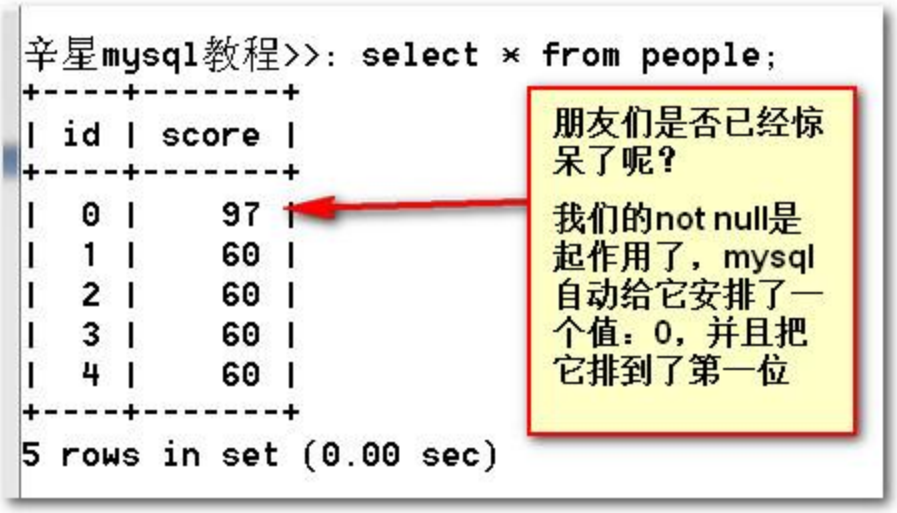
5.然后我们看一下效果：



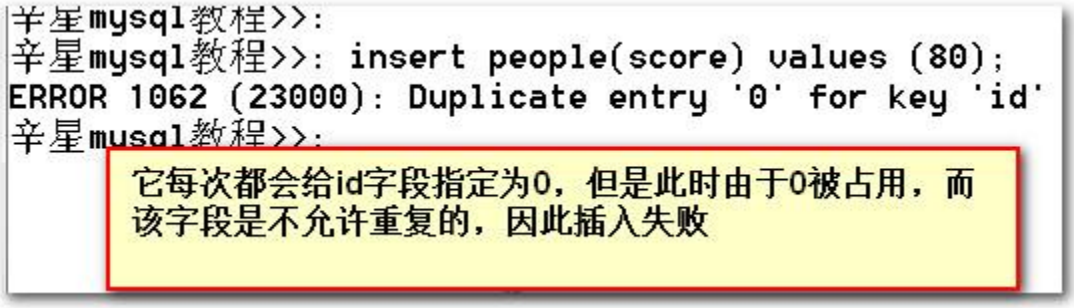
6.如果我们避开 id 这个非空的字段，直接插入 score 这个字段，发现结果是这样子滴：



7.然后发现该 people 的数据是这样子滴：



8.可能有些朋友会想，我继续给 score 插入几个值(不指定 id 的值),会怎么样呢？答案是会插入失败，因为会产生重复的值，看下面执行语句：



9.我想原理大家此刻应该明白了吧。

**********小结**********

1.我们这一小节学的这些比如 unique，not null，default 等等这种都是对一个字段起作用的。

2.最重要的就是理解主键的概念以及掌握其使用方法，这个是重中之重。

3.此外还有外键、引擎、字符集等一些概念虽然也算常见，但是，我还是把它们放到了其他地方，我想给读者一个清晰的知识体系。

4.辛星 mysql，期待您的关注。

第三部分：mysql 数据查询

第一节：查询基础

**********为何查询这么难**********

1.如果查询数据数据都像 drop 和 truncate 那样，一次性查看所有数据，那直接 select * 就搞定了，我也不会花一章来讲解它。

2.后来我看过一篇博客，就是介绍为何 mysql 中插入、修改、删除数据都如此简单，但是查询数据这么难。

3.原因之一就是业务的复杂性，比如我们从学生表中取得它所在的系，然后在系表中取出系所在的地址。这就涉及到跨表操作，在一个 sql 中搞定倒不是很难，但是业务逻辑还是蛮复杂的。

4.原因之二就是对于效率的要求，由于绝大多数对数据库的操作都是查询操作，因此，对某些网站，查询数据库的效率往往决定着网站的响应速度。

5.因此，我们为了效率，把容易理解执行速度慢的 sql 语句变成了不容易理解但是执行速度快的 sql 语句，加大了学习难度。

**********整理信心**********

1.不知道大家会不会打退堂鼓：查询那么难，学不下去了。

2.知难而退不是真正的强者应该做的事，我们应该拿出红军二万五千里长征的顽强毅力把它彻底征服。

3.有辛星这么优秀的导师，难道你对星哥的教授能力那么不自信吗？

4.当然，你有更好的心得，不妨告诉我：xinguimeng@163.com

5.让中国人学编程变得更简单，提高国人的整体编程水平，传播尖端科技，这才是我们终生为之奋斗的目标。

6.不废话了，直接开干。

********不看冗余信息***** 指定列名*****************

1.前面我们用 select * 这种查询的方式，其实是效率最差的方式。

2.我们很多时候只需要查询我们需要的那些列就可以了。

3.语法格式：【select 列 1，列 2,..... from 表名 其他条件】

4.实例：

5.咱们的数据只有一行，不过咱们查找的都是咱们需要的信息，这样会减小数据库的开销，进而提高效率。

6.咱们 select * from phone；中的*就是一个通配符，查询所有信息，虽然书写简单，但是给 mysql 的压力很大，因为太多的任务需要让 mysql 去做。

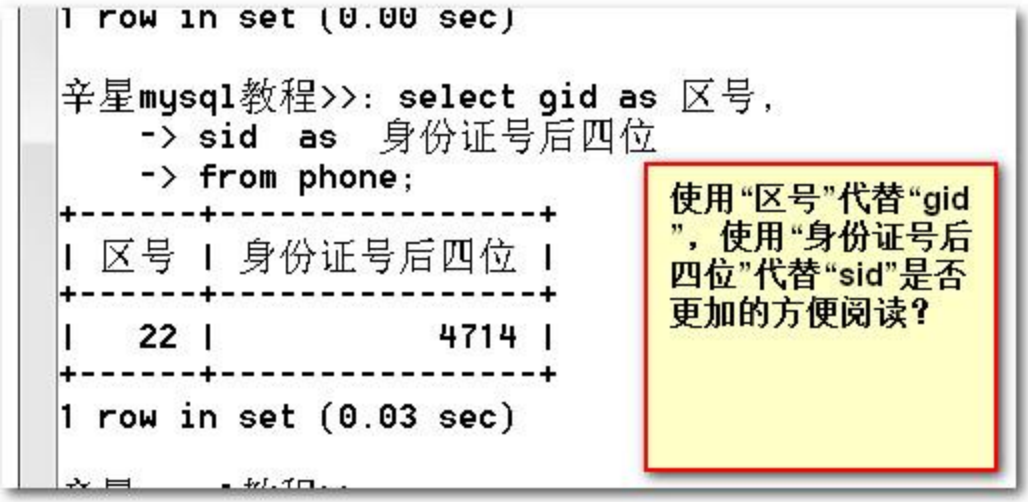
*********显示信息更人性化*********指定别名*************

1.有时候我们需要给检索出来的列指定别名，这样显示出来的信息会更加人性化。

2.指定别名也很简单，我们使用【列名 1 as 别名 1，列名 2 as 别名 2，.....】这种格式。

3.值得注意的是，我们在数据库中建表的时候一般是不用中文的，但是我们使用 select 显示的时候，是可以中文显示的，也就是说，我们使用中文别名。

4.看下面截图：



5.上面我们把 gid 这一栏的信息的标题改为“区号”，我们把 sid 这一栏信息的标题改为“身份证号后四位”，显示界面更加友好。

6.其实别名的作用远不止这些，后面大家会看到更多关于别名的精彩应用。

7.其实这里的 as 是可以去掉的，就像前面 insert into 中的 into 一样，是可以去掉的，我不建议大家去掉。

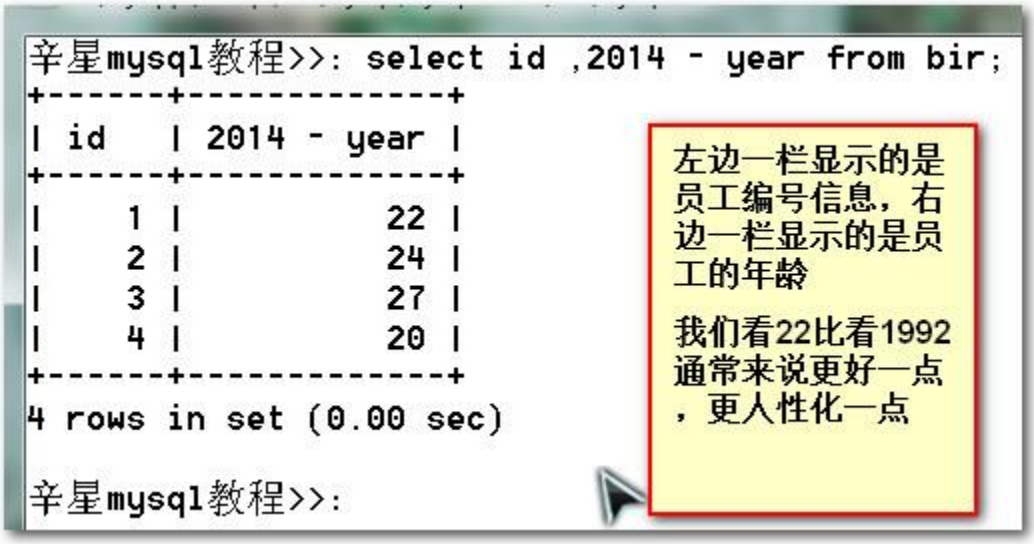
******运算********* 得到想要的数据**********

1.有时候呢，我们存储的信息，并不是我们最终想要的信息。

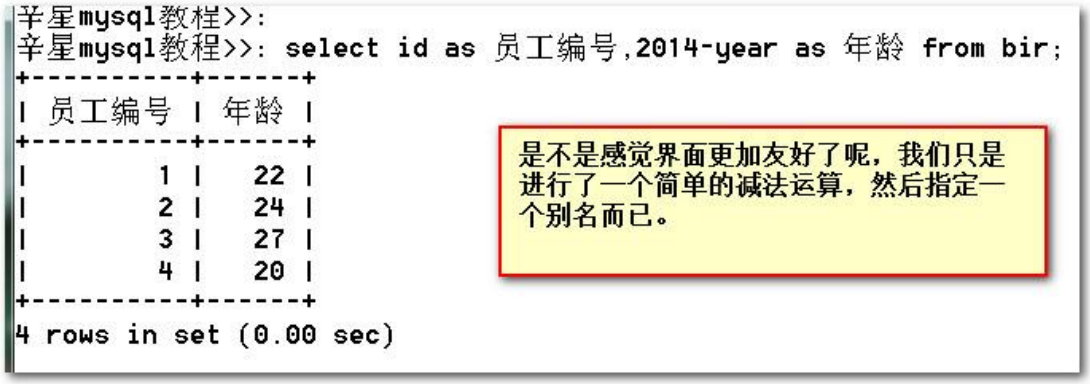
2.比如在存储的时候，我们往往只存储其出生年份，但是我们查看信息的时候，我们往往更关注其年龄。

3.比如我有这样一个表 bir，它存储的是人的生日信息，看一下：

4.那么我们怎么让它变得更加人性化呢？我们考虑到新手的感受，分为两步。第一步我们先得到员工的年龄，我们使用今年的年份2014，减去year这一列的数据就可以了，因此，我们使用如下的sql语句【select id , 2014 - year from bir;】,看下面效果：



5.然后我们指定一下别名就可以了，看我操作：



6.这里的运算对于常见的加减乘除都是支持的，注意除法使用“/”运算符，结果为浮点数。

7.也支持列和列的运算，比如我们的数据库中记载了货物的长度、宽度、高度、密度，那么我们可以直接用这四个数据得到货物的吨位这一属性信息，当然这个得要求货物比较整齐的才行。

**********统计信息**********

1.我们对列的处理绝不仅仅限于加减乘除等这种很简单的手工计算。

2.它还可以进行信息的统计，当然，我们需要使用一些统计函数，常用的有 count(用来计数)、sum(用来求和)、max(求最大值)、min(求最小值)、avg(用来求平均数)。

3.比如下面我们查看我们这个小型团队的年龄结构：

```
辛星mysql教程>>: select
    -> count(year) as 员工总数,
    -> 2014 - min(year)  as   最大年龄,
    -> 2014 - max(year)  as   最小年龄,
    -> 2014 - avg(year)  as   平均年龄
    -> from bir;
+----------+----------+----------+----------+
| 员工总数 | 最大年龄 | 最小年龄 | 平均年龄 |
+----------+----------+----------+----------+
|        4 |       27 |       20 |  23.2500 |
+----------+----------+----------+----------+
1 row in set (0.00 sec)
```

我们在mysql数据库中只存储year一个字段，但是我们通过mysql，能很轻松的获取这些统计信息

这些都很好理解把

4.就像我们在数据库中只保存一个 year 这一个字段，但是我们通过强劲的 mysql，能够获取足够丰富的统计信息。

**********mysql 函数以及计算器**********

1.可能有 Linux 编程经验的或者使用 python 编程经验的会感觉那种交互式的脚本使用起来简直爽到爆。

2.但是我们强劲的 mysql 也是支持的，比如我们想查看当前的时间，可以使用【select now(); 】来输出当前时间：

```
辛星mysql教程>>:
辛星mysql教程>>: select now();
+---------------------+
| now()               |
+---------------------+
| 2014-08-16 14:24:40 |
+---------------------+
1 row in set (0.07 sec)
```

好吧，你赢了，今天是2014年8月16日，下午两点半

3.这里说一下这个 now()是 mysql 内置的一个函数，它不需要参数，直接使用 select 查看其返回值即可。

4.我们还可以把 mysql 当做一个计算器，如下示例：

```
辛星mysql教程>>:
辛星mysql教程>>:
辛星mysql教程>>: select 1992 * 9 *13;
+---------------+
| 1992 * 9 *13 |
+---------------+
|        233064 |
+---------------+
1 row in set (0.00 sec)

辛星mysql教程>>:
```

相当于自带一个计算器，是不是很实用？

5.这种交互式的编程模式，还是非常舒服的，至少我这个 python 党这么感觉。

**********说明**********

1.本教程并不是一个手册，由于 mysql 中的这些函数我们这里并不会用到多少。

2.感兴趣的话可以去搜索下，我对函数这一块就不展开了。

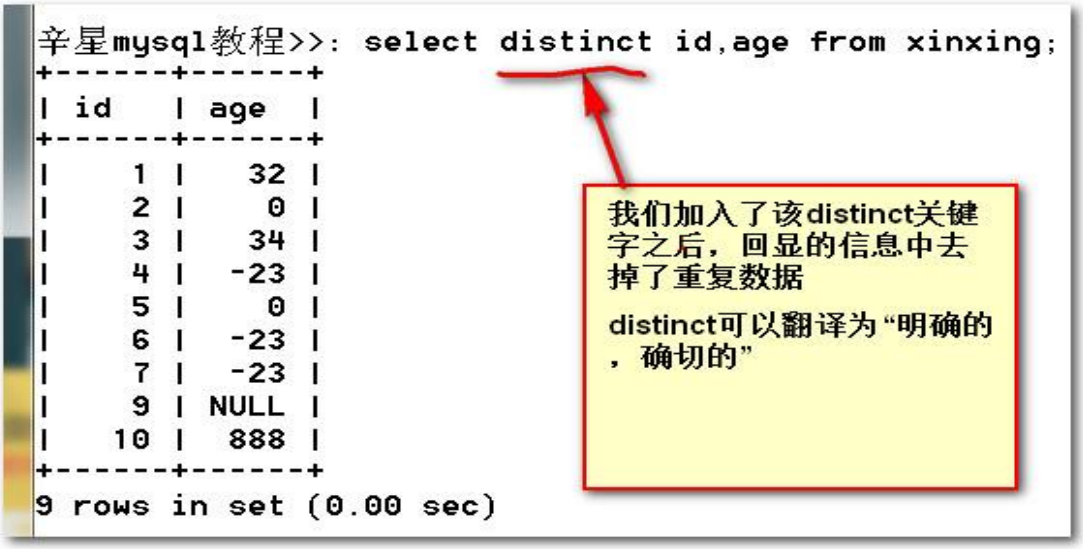
**********去除重复数据**********

1.有时候我们会看到完全重复的数据：

```
辛星mysql教程>>:
辛星mysql教程>>: select id ,age from xinxing;
+------+------+
| id   | age  |
+------+------+
|    1 |   32 |
|    2 |    0 |
|    3 |   34 |
|    4 |  -23 |
|    5 |    0 |
|    6 |  -23 |
|    7 |  -23 |
|    9 | NULL |
|   10 |  888 |
|   10 |  888 |
+------+------+
10 rows in set (0.00 sec)
```

这里由两行重复数据，很多时候我们不想看到重复数据

2.那么我们不想显示重复信息应该怎么办呢？我们只需要在 select 关键字之后，在被选择的字段之前加上一个 distinct 即可，看下面截图：



3.我们看到，此时此刻的重复信息就不在显示了。

**********小结**********

1.这一小节我们主要涉及检索信息的一些方式，比如别名，比如对检索的列进行一些处理。

2.当然，还有一些零散的知识点，比如 distinct 去除重复的列，比如内置函数，比如常见的统计函数。

第二节：where 子句

**********条件表达式**********

1.早在前面讲 update、insert、delete 的时候，我们就接触过 where 子句了，但是我们那里的使用非常简单，也不系统，这里我们就系统的分析一次。

2.where 子句的特点就是如果后面的表达式为 true，那么就把这些信息进行相应的处理，不管用于修改、删除，还是查询。

******第一类*********比较运算符**********

1.比较运算符用于比较两个表达式，mysql 支持的常用的比较运算符有：=等于、<小于、<=小于等于、>大于、>=大于等于这些都很常见。

2.<>表示不等于，!=也表示不等于。

3.<=>表示两者相等或者都为空。也就是该符号两边的值如果都为空值，那么它会返回 true。如果一边不为空值，那么必须两边的值相等。

4.这个我们去你前面用的也比较多了，我这里只是简单的举个例子：

```
辛星mysql教程>>: select  * from xinxing
    -> where id >5;
+------+------+
| id   | age  |
+------+------+
|    6 |  -23 |
|    7 |  -23 |
|    9 | NULL |
|   10 |  888 |
|   10 |  888 |
+------+------+
5 rows in set (0.03 sec)
```

从结果集中可以看出，这里的起始行的id从6开始

******第二类******between**********in**********

1.我们用 between .....and 来表示选择一个范围中的数据，比如我们想看大于等于 3 并且小于等于 7 之间的数据，可以使用【where between 3 and 7】。

2.看下面操作：

```
辛星mysql教程>>: select * from xinxing
    -> where id  between 3 and 7;
+------+------+
| id   | age  |
+------+------+
|    3 |   34 |
|    4 |  -23 |
|    5 |    0 |
|    6 |  -23 |
|    7 |  -23 |
+------+------+
5 rows in set (0.00 sec)
```

这里取出的id都是从3开始（包含）到7结束（结束）的数据

3.有些时候我们想取一些离散的数据，我们可以用 in 操作符。比如 id 为 2,4,5,9 这四条数据，那么我们的 where 子句可以这么写【where id in (2,4,5,9)】.

4.我们如下操作：

```
辛星mysql教程>>:
辛星mysql教程>>: select * from xinxing
    -> where id in (2,4,5,9);
+------+------+
| id   | age  |
+------+------+
|    2 |    0 |
|    4 |  -23 |
|    5 |    0 |
|    9 | NULL |
+------+------+
4 rows in set (0.00 sec)
```

我们只取出id为2,4,5,9这四个数据即可
这里的in后面跟一个结果集

等我们后面讲到了子查询，也可以用子查询的结果

5.not 操作符表示取反的意思，它可以和 between....and 结合使用，还可以和 in 结合使用，都表示取相反的数据。

6.比如说我们查看那些 id 不是 2,4,5,9 的数据：

```
Database changed
辛星mysql教程>>: select * from xinxing
    -> where id not in (2,4,5,9);
+------+------+
| id   | age  |
+------+------+
|    1 |   32 |
|    3 |   34 |
|    6 |  -23 |
|    7 |  -23 |
|   10 |  888 |
|   10 |  888 |
+------+------+
6 rows in set (0.04 sec)
```

这里选取的内容不是2,4,5,9的数据

**********第三类**********是否为空**********

1.如果一条记录中的某个字段为 NULL，那么表示该字段并没有存储数据。

2.有时候我们对该 NULL 数据比较关注，我们可以用【where 字段名 is null】来查找那些为 null 的字段，用【where 字段名 is not null】来查找那些不为 null 的字段。

3.看下面操作：

```
辛星mysql教程>>: select * from xinxing
    -> where  age is null;
+------+------+
| id   | age  |
+------+------+
|    9 | NULL |
+------+------+
1 row in set (0.00 sec)
```

查询age为NULL的所有字段

******数据类型**********char**********

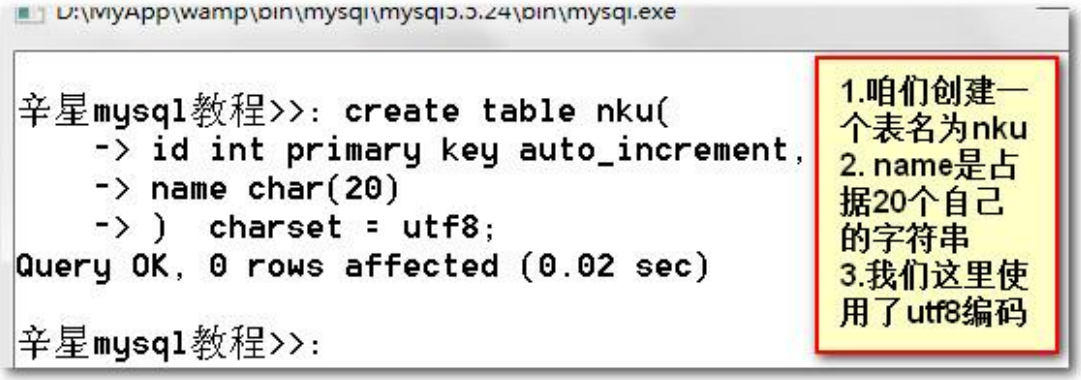
1.在我们继续向前推进之前，我们再认识一种数据类型才行，那就是 char 类型，它表示字符类型。

2.我们创建数据表的时候，通常需要在它后面指定我们使用多少个字符，比如 char(20)表示 20 个字符大小的字符串。

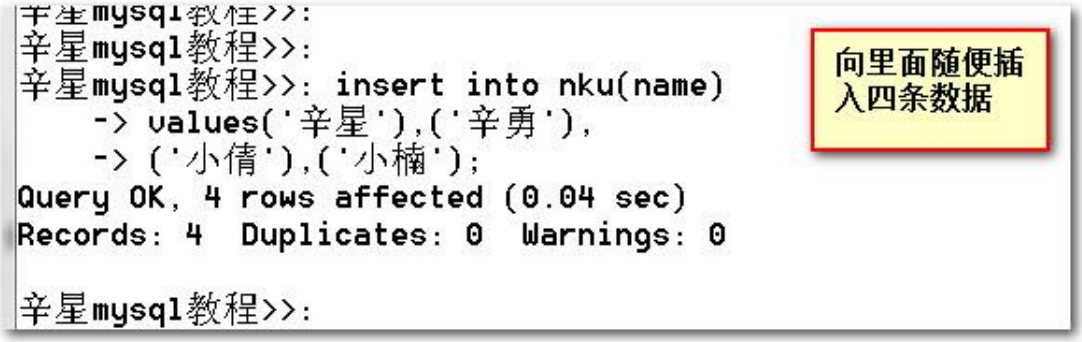
3.我们可以在 char 类型中使用中文，但是前提我们必须合适的字符集，如果使用了无法显示中文的字符集，那么是无法有效的使用中文的。

4.咱们可以在建表的最后加上一个 charset = gbk 或者 charset = utf8，这样就支持中文了。

5.咱们先创建一个表，它包含 id 和 name 两个字段，id 是 int 类型的主键，name 是 char(20)，看下面代码：



6.然后咱们向里面插入四条数据：



**********模式匹配**********

1.like 运算符用于指出一个字符串是否与指定的字符串相匹配，咱们这里的 char 类型是足够的。

2.我们使用 like 运算符的时候，通常和“%”或者“_”一起使用，%通常代表零个或者多个字符，而_代表单个字符。

3.而且 mysql 默认是不区分大小写的。

4.在 where 条件中我们通常这么写【where 字段名 like 匹配字符串】。

5.实例代码：

```
Database changed
辛星mysql教程>>: select * from  nku
    -> where
    -> name like '%辛%';
+----+------+
| id | name |
+----+------+
|  1 | 辛星 |
|  2 | 辛勇 |
+----+------+
2 rows in set (0.00 sec)
```

我们选取那些name中含有“辛”字样的记录

6.其实我们还可以在 like 前面加一个 not 来取反，比如：

```
辛星mysql教程>>:
辛星mysql教程>>: select * from nku
    -> where name not like '%辛%';
+----+------+
| id | name |
+----+------+
|  3 | 小倩 |
|  4 | 小楠 |
+----+------+
2 rows in set (0.00 sec)

辛星mysql教程>>:
```

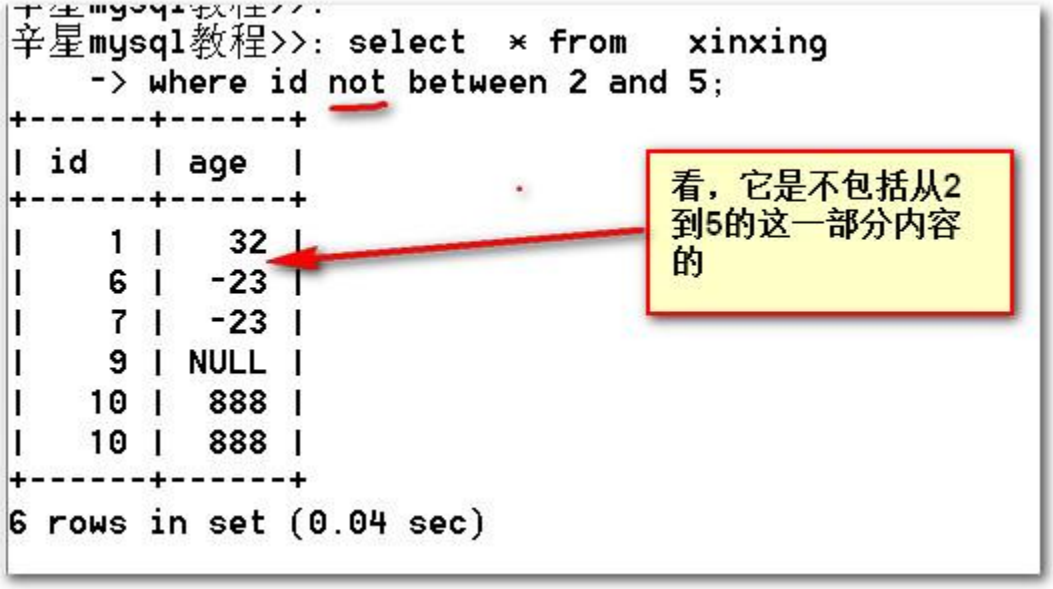
选取name字段中没有“辛”字样的那些记录

**********说明**********not 写在哪里**********

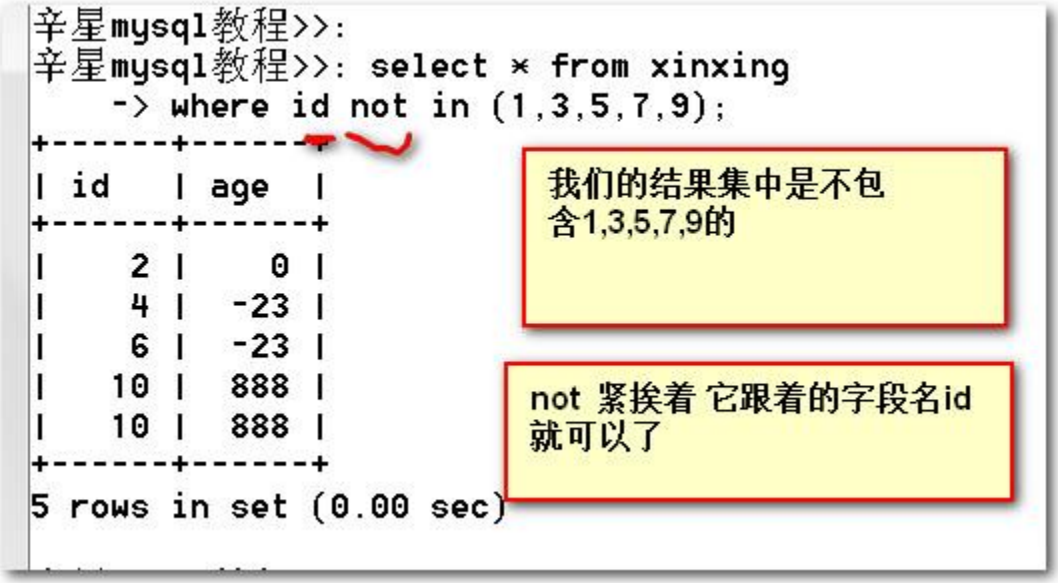
1.可能导致新手朋友们很迷茫的一件事就是 not 应该写在哪里。

2.其实很好理解，not 应该写在它所起作用的那个字段的后面，紧挨着它写，肯定不会错。为什么，原因很简单，因为字段后面跟逻辑表达式，我们直接对逻辑表达式取反就可以了。

3.比如我们的 between .... and 这套格式：



4.比如我们的 in 运算符，那么看下面示例操作：



5.应该很好理解吧。

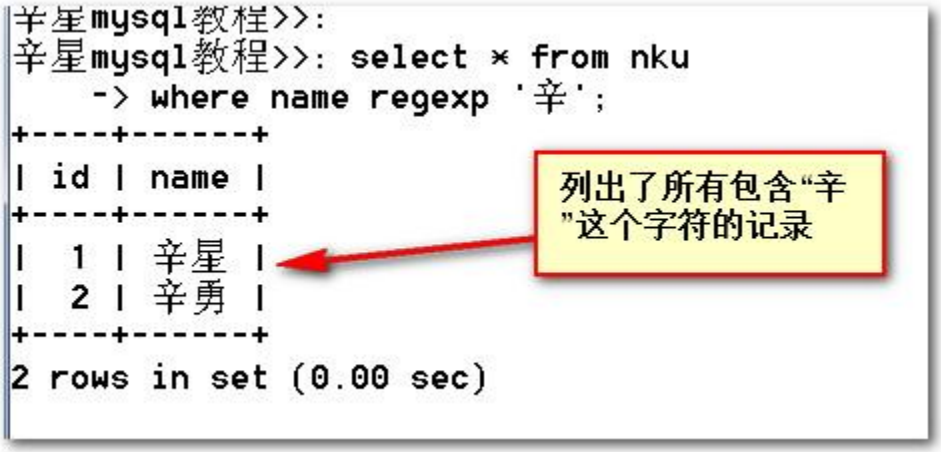
**********正则表达式**********regexp**********

1.regexp 运算符用来执行更复杂的字符串比较运算，它是 regular expression 的缩写，也就是“正则表达式”。

2.regexp 和 rlike 是同义词，也就是说二者可以互换，效果是一样的。

3.它的使用和 like 很相似，但是必须记住我们是正则匹配，不是使用通配符匹配，它的使用如下【where 字段名 regexp 匹配字符串】。

4.我们看下面实例演示：

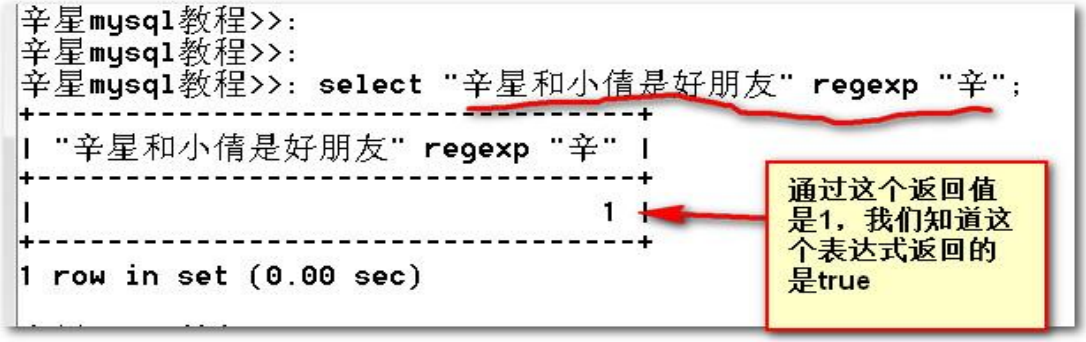


5.如果读者对正则表达式不熟悉，可能会听起来很费力，可以跳过这部分或者搜集下相关资料，我在写 python 的教程的时候写过正则表达式，这里不写了。

**********regexp **********like**********

1.可能有些读者已经晕掉了，regexp 究竟和 like 有神马区别呢？

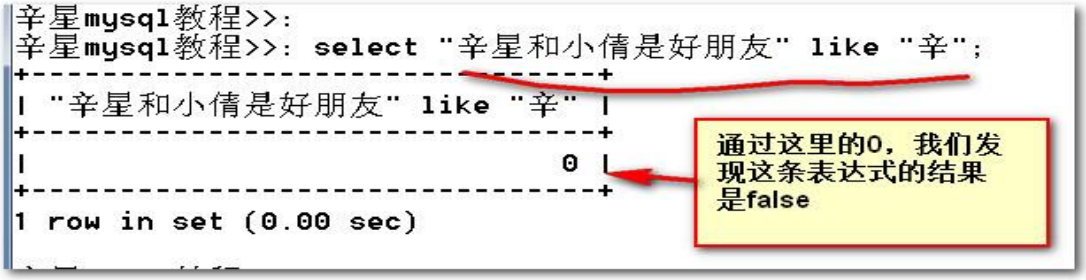
2.答案很简单，regexp 是正则匹配，也就是说，它只要 regexp 的右边是左边的一个子集就可以了，因此【“辛星和小倩是好朋友” regexp “辛”】会返回一个 true，看下面：



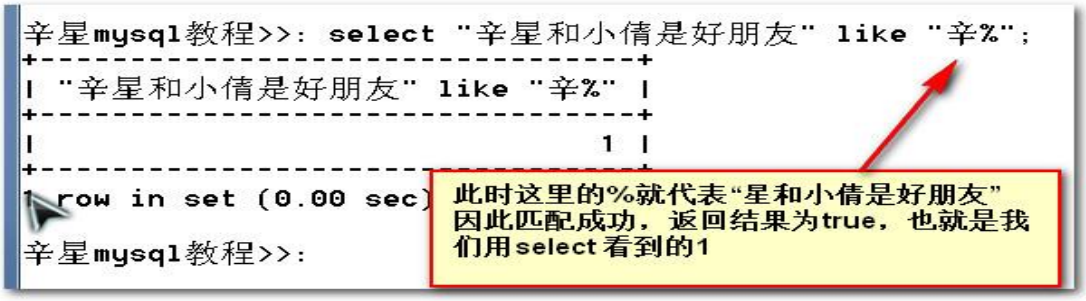
3.既然它会返回一个 true，那么自然会包含在我们的结果集中。

4.而 like 是非正则匹配，也就是说，它要求 like 运算符两边不是局部和部分的关系，而是必须一一对应才行。

5.举个例子，【“辛星和小倩是好朋友” like “辛”】会返回一个 false，看下面代码：



6.但是【“辛星和小倩是好朋友” like “辛%”】会返回一个 true，看下面代码：



7.那么拿到实战中呢？看下面代码：

8.大家应该明白“正则匹配”和“非正则匹配”的区别了吧，简单来说，一个是包含，一个是相等。

**********小结**********

1.总体来说，这一节讲的是 where 子句能干什么，where 子句虽然貌似很庞大，但是我们进行细分之后，发现它很系统。

2.对于 where 条件常用的几个运算符，我们用表格来说明。

3.看该表格：

运算符种类	具体运算符	功能
比较运算符	<,>,=等等	进行简单的比较运算操作
模式匹配(模糊查询)	like	非正则匹配
	regexp(rlike)	正则匹配
范围比较	between...and	连续的范围
	in	非连续的单个取值的集合
空值比较	is (not) null	判断字段是否非空

4.这一节一定要掌握透彻，如果读者掌握的不好，建议多读几遍，跟着实例去亲自敲敲代码，体会一下。

5.这一节如果读者掌握的不好，那么子查询就无法进行，子查询是建立在它的基础之上的。

第三节：子查询

**********什么是子查询**********

1.本来子查询可以放到 where 那一节里面去的，但是我为了表示重视，可以单独作为一节。

2.我们可以在一个查询嵌套到另一个查询、插入、修改、删除中去，被嵌入的这个查询就是一个子查询。

3.子查询也有人翻译为“子选择”或者“嵌套选择”，其实都是一回事。

**********为什么需要子查询**********

1.通过我们第二节 where 条件的学习，大家可能发现它的功能已经比较强大了。

2.包括我们后面还会学习到各种连接和联合，我们对数据库的处理能力会更加强大。

3.mysql 从 4.1 版本开始支持子查询，它的出现会大大的简化我们 sql 语句的书写。

5.子查询最大的好处就是我们原本需要用多个 select 语句或者没办法用连接完成的工作使用子查询来完成，关于连接查询，我们后面会讲。

**********按照结果集来分类子查询**********

1.我们首先要明确一点，那就是子查询查询到的是一个结果集，这个结果集有大有小，我们可以根据它的种类来分类。

2 第一类就是标量子查询，它返回的是一个具体的数值，比如我们在员工表中查找工号最小的年龄，它就会返回一个特定的值，比如 23.

3.由于标量子查询返回的是一个精确值，因此它也经常和where子句中的比较运算符一起搭配使用。

4.第二类是列子查询，它返回的是一列值，它的一个特点就是这一列值具有相同的数据类型，因此它特别适合和where子句中的in操作符一起用。

5.第三类就是行子查询，它返回的是一行值，常见的就是我们查询一条记录，比如我们就查询员工号为02321的这个员工的姓名和工龄，那么这就返回一个1行2列的一条记录。

6.第四类就是表子查询，它返回的是N行M列值，比如我们常用的select *一般来说返回的就是N行M列的值。

7.其实行子查询就是表子查询中N为1的一个特例，它们呢无法在where子句里出现，但是我们可以从这个表里再次筛选信息，它可以像一个表一样，被from子句引用，也可以被exists引用。

**********按照子句来分类子查询**********

1.这个太容易了，只要我们明白了上面按照结果集是怎么分的，那么这个就几乎不费力了。

2.用在where子句中的叫做where型子查询。

3.用在from子句中的叫做from型子查询。

4.用在exists子句中的叫做exists型子查询。

*********两种分类对比图表**********

按子句类型来分	按结果集来分
where型子查询	标量子查询(常和比较运算符一起用)
	列子查询(常和in一起用)
	行子查询(只能和等号一起用)
from型子查询 exists型子查询	行子查询、表子查询 (能用from的地方基本也都能用exists)

**********理论与实战**********

1.上面我讲了一堆理论，不知道大家理解的怎么样，反正数学出身的我感觉思路还是蛮清晰的。

2.如果读者感觉有点迷茫，也不要紧，咱们先跟着下面把这些实例给敲完一遍，跟着体会一下。

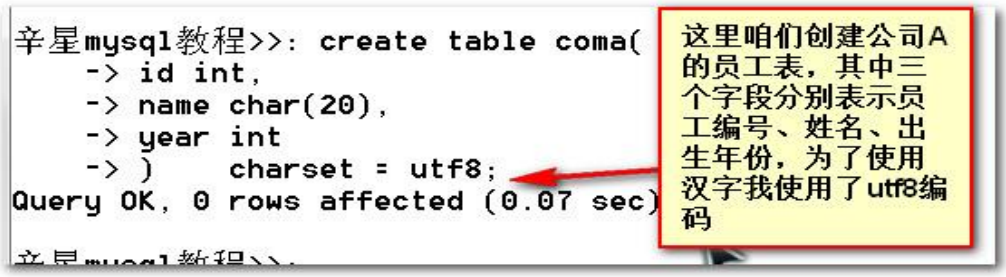
3.有了实践的底子，然后再去看理论，可能就好理解的多了。

4.其实吧，不理解理论也可以，只要看懂子查询，会写子查询就勉强过关。

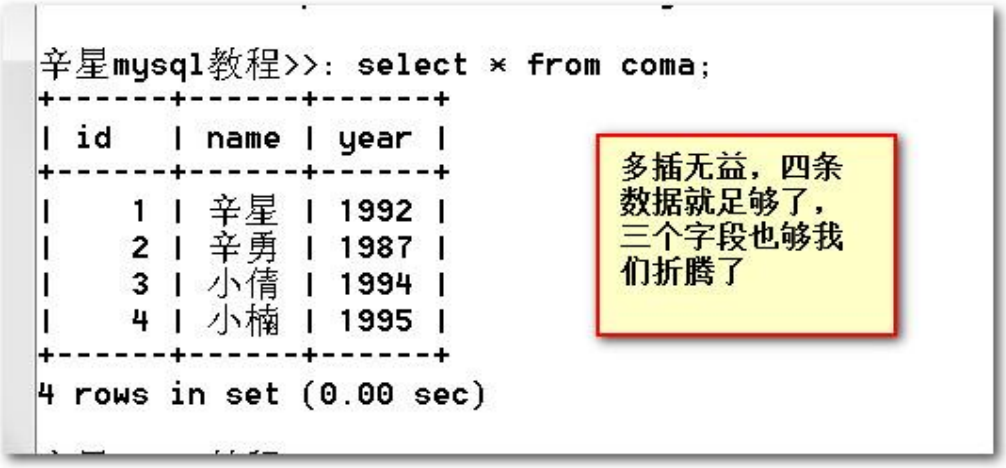
**********先建两张表**********

1.虽然说子查询并不需要两张表，但是使用两张表毕竟会直观很多。

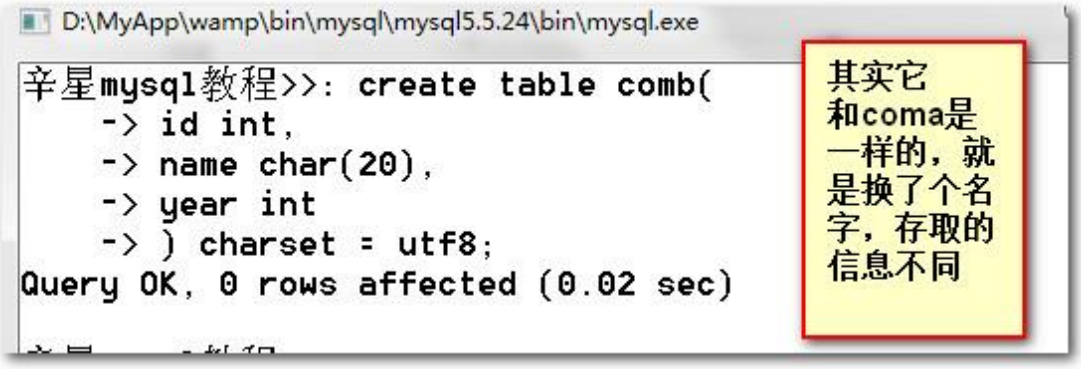
2.第一个是A公司的员工表，我们让表名为coma，首先建表如下：



3.然后填充数据如下：



4.然后我们建立公司 B 的 comb 这个表：



5.然后我们向 comb 插入一些数据：

**********where 型子查询实战一**********

1.咱们先练习最简单的，也就是我们的子查询得到的是一个具体的数，比如说我们先从 coma 里面求出 id 最大的这个员工的编号，然后我们根据此编号，从 comb 里面找出比它不小的编号的员工的编号、姓名和出生年月信息。

2.这个业务要求还是挺简单的，假定我们从表 coma 里面得到的这个数据记为 X，我们就可以写如下的 sql 语句：

select id, name, year from comb where id >= X;

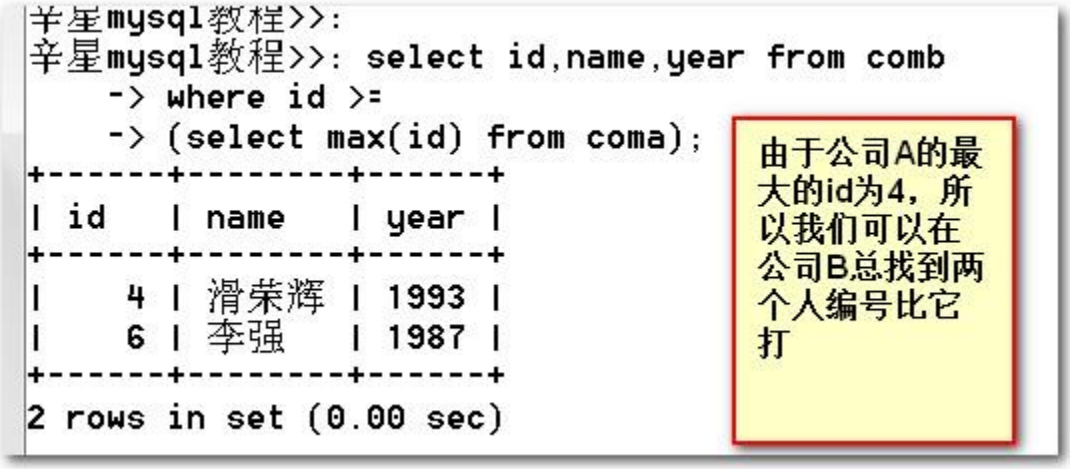
3.然后我们再分析这个 X 应该怎么写，它是得到 coma 表中最大的那个 id 值，因此应该这么写：

select max(id) from coma ;

4.因此最终的 sql 语句就是这样子滴：

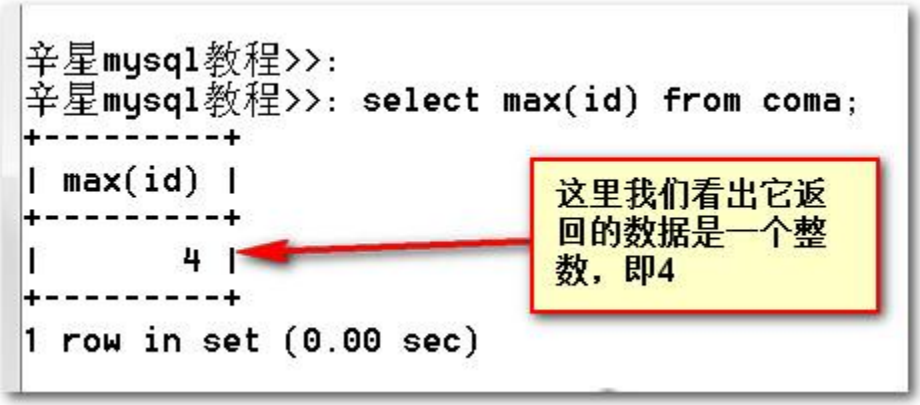
select id，name，year from comb where id > = (select max(id) from coma);

5.看下面执行效果：



6.说明：这里的内层查询 select max(id) from coma;返回的是一个具体的数据，也就是 4，我们可以用 4 替换为外层查询的那一串子查询即可。

7.对于内层子查询得到的数据：



**********where 型子查询实战二**********

1.有这样一个业务要求，让我们用一个 sql 语句来找到公司 A 中具有和公司 B 中相同编号的员工的编号、姓名、出生年份信息。

2.如果某童鞋想了想说：用连接查询很简单，当然是很简单哈，我们这里用子查询来做。

3.假设我们从内层查询得到的结果集为 X，我们先构造外层查询：

select id，name，year from coma where id in X；

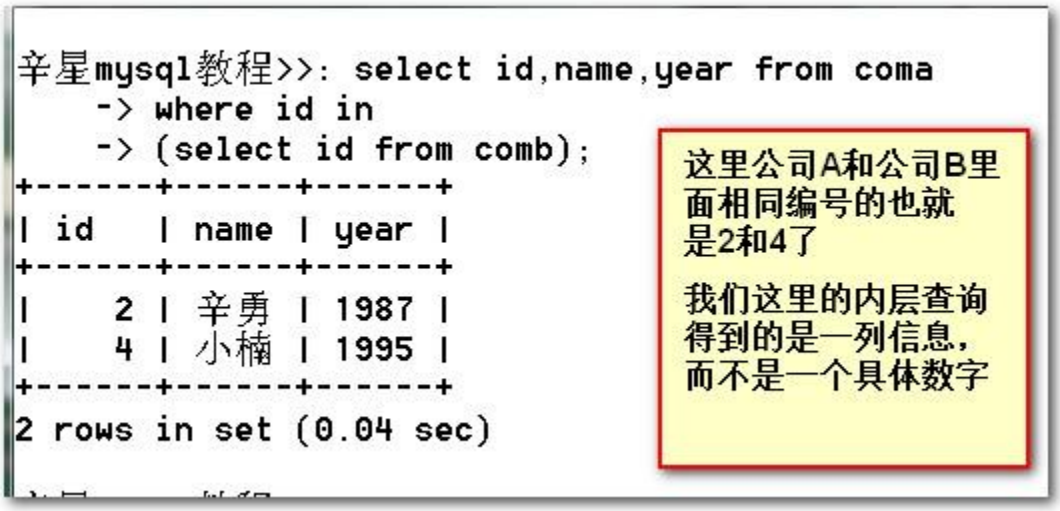
4.然后我们构造内层查询，我们只需要把公司 B 的员工的编号统一拿出来来就 ok 了：

select id from comb ；

5.然后我们的总的 sql 语句可以这么写：

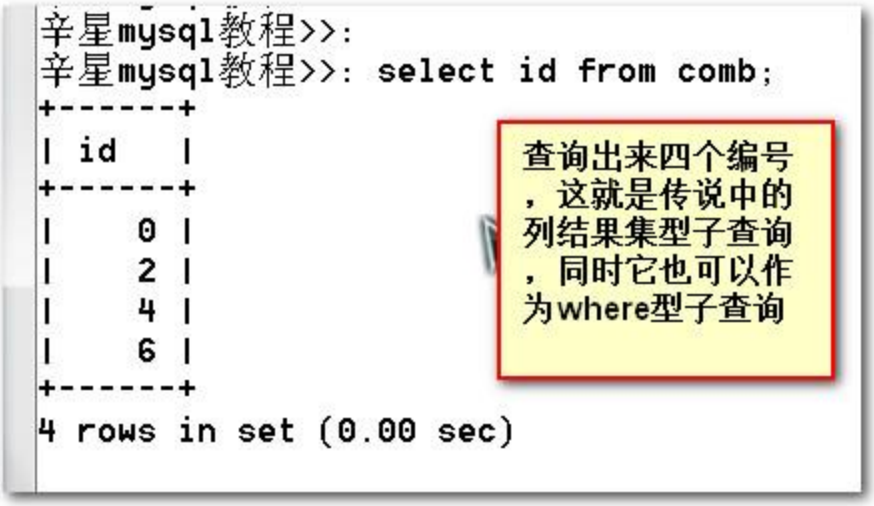
select id，name，year from coma where id in (select id from comb);

6.然后我们发现执行效果如下：



7.在 where 型子查询实战 1 里面我们的内层查询得到的是一个具体的数，这里我们得到的是一列信息。

8.我们可以看一下效果：



9.这里我们的内层查询得到的是一列数据，就相当于一个集合，我们可以使用 in，来判断我们外层的结果集是否在这里面。

**********from 型子查询实战**********

1.当初构造这个表感觉还是结构太简单了，一时没想到很有意义的 from 型子查询的例子，那我们就为了练习使用 from 型子查询来写一个稍微别扭点的需求把。

2.假如我们需要随机生成一个数据，并且我们从公司 A 的员工里面取出 id 不大于该数字的并且名字里含有“辛”这个字符的所有员工的工号、姓名、年龄信息。

3.首先假设内层获取的数据是 X，那么我们可以使用：

select * from X as xingge where name regexp ‘辛’；

4.然后我们向 X 中只需要产生一个随机数，并且让 id 小于这个随机数就可以了(这里我们用到了 floor 和 rand 这两个随机函数)：

select * from coma where id < = floor(rand()*5);

5.然后我们把这个拼成一个 sql 语句即可，那么我们 下面开始执行它，我们一起来看看效果：

6.当然多执行几次，发现也有取得的值不同的情况：

7.这个例子并不算太恰当，因为它不使用子查询更简单，但是它使用子查询却没有任何的语法错误，在后续版本中，我会从一开始就修正这两个表，让他们更加合理。

8.所谓 from 型子查询就是由于咱们子查询得到的结果是一个 N 行 M 列的一个数据表，我们可以继续从它里面进行 select，非常方便。

9.但是注意，此时必须给该数据表指定一个别名。

**********提前免疫**********

1.我记得我以前一个老师喜欢说一句话：“先给你们打个预防针”，那么我就给大家提前免疫一下，也就是咱们先预先说明一些知识点。

2.我们前面可能在两个查询中会用到两个表，比如 coma 和 comb，那么当我们需要用到这两个表中的字段的时候，比如 coma 和 comb 都有 id，name，age 字段，那么，我们要是想使用这些字段，应该怎么办呢？

3.我们可以以表名.字段名的方式来指定，比如 coma.name 就是表示表 coma 的 name 字段，比如 comb.id 表示 comb.id 字段。

**********exists 型子查询**********

1.如果把一个查询大致分为外查询和内查询，那么如果内查询的结果为 true，那么外查询的结果就显示出来，否则内查询就不会显示出来。

2.比如有这么一个业务逻辑，我们在公司 B 中找出所有和公司 A 中有相同年份出生的人，那么我们可以这么写：select * from comb where exists (select * from coma where comb.year = coma.year);

3.接下来我们分析一下这个运行原理：我们先从 comb 这个表中取出一条数据，然后进行 exists 判断，如果 select * from coma where comb.year = coma.year 可以取出内容，那么该条内容就显示在结果集中，否则就不会出现在结果集中。

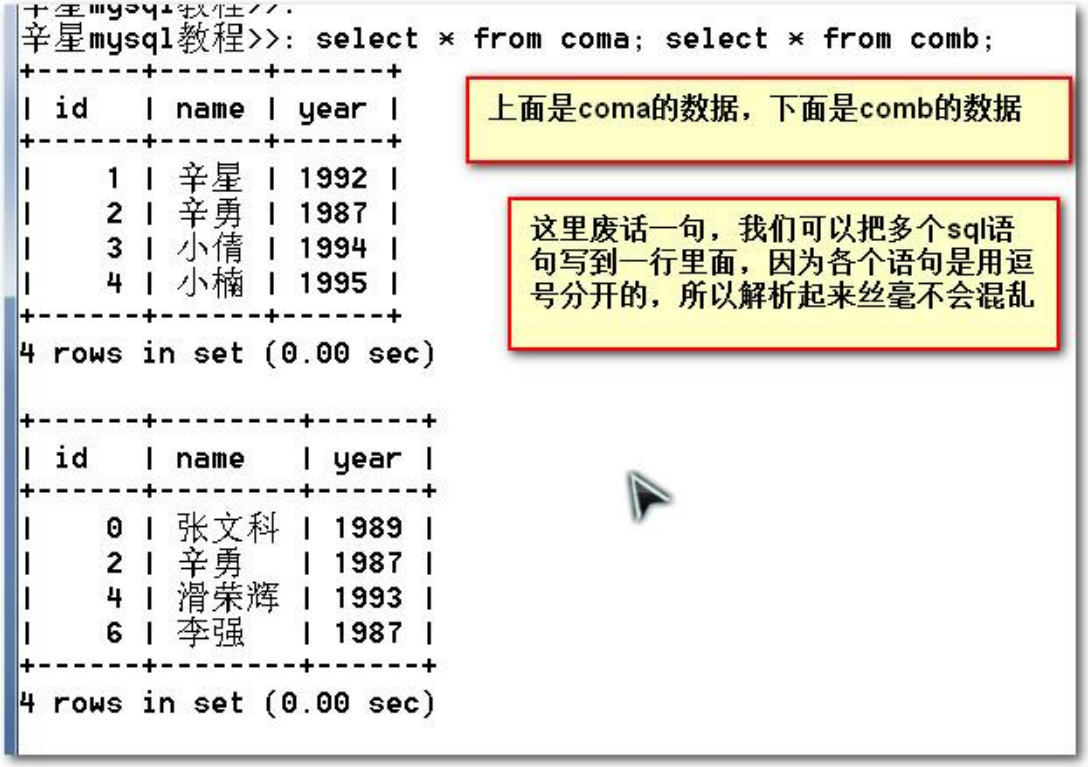
4.然后我们再次从 comb 表中取出一条数据，然后再次进行 exists 中判断。也就是说，我们只会从 comb 表中取出数据，如果 exists 后面的小括号中的内容为真，则该条内容显示，否则为假。

5. 下面是运行结果：

6.和读者分析的一致吗？如果不一致的话，那么我这个大哥哥的作用体现出来了，我给读者朋友们分析一下。

**********exists 型子查询结果集分析**********

1.我们首先看一下表 coma 和表 comb 的数据：



2.然后我们根据上述子查询语句：select * from comb where exists (select * from coma where coma.year = comb.year);

3.我们首先执行外层查询，即我们首先执行 select * from comb，然后我们取出了一条记录：(0，“张文科”，1989),然后我们执行 select * from coma where coma.year = comb.year;

注意这里的 coma.year 应该被换成 1989，也就是从 coma 表中查找一下是否有 year 这个字段为 1989 的，结果是米有找到，那么该条数据会被删除。

4.然后我们外层查询又会查询出一条记录:(2,"辛勇",1987),然后我们相当于执行 select * from coma where coma.year = 1987;发现在 coma 中有一条数据是(2,"辛勇",1987),因为这两个记录是相同的，因此我们把这条记录保存到结果集中。

5.然后我们继续搜索，又得到一条记录:(4,"滑荣辉",1993),然后我们找一下 coma 中有没有记录的 year 字段为 1993 的，发现没有，于是，该条记录被抛弃。

6.然后我们继续搜索，又得到一条记录：(6,"李强",1987),然后我们找一下 coma 中是否有记录的 year 字段为 1987，发现有一条记录是(2,"辛勇",1987)，因此这两个记录的 year 属性是一样滴，因此，我们的(6,"李强",1987)会被保存到结果集中。

7.经过我们不懈的努力，我们终于搜索完毕，以上就是我手工模拟 mysql 的执行流程，希望对读者有所帮助。

**********总结**********

1.本小节我们主要讲解了子查询，子查询我不知道大家学起来是否困难，如果您在学习本小节中有任何问题和不解，都可以发邮件给我：xinguimeng@163.com 或者联系我的私人 QQ：1808347923。

2.子查询按照语句可以分为 where、from、exists 型查询，还可以根据结果集分为标量子查询、列子查询、行子查询、表子查询。

3.辛星，愿做您终生的朋友。

第四节：group by 和 having

**********易迷惑点**********

1.很多学员说这一部分容易迷惑，既然这样，那咱们就讲慢一点，争取把每一点都说清楚。

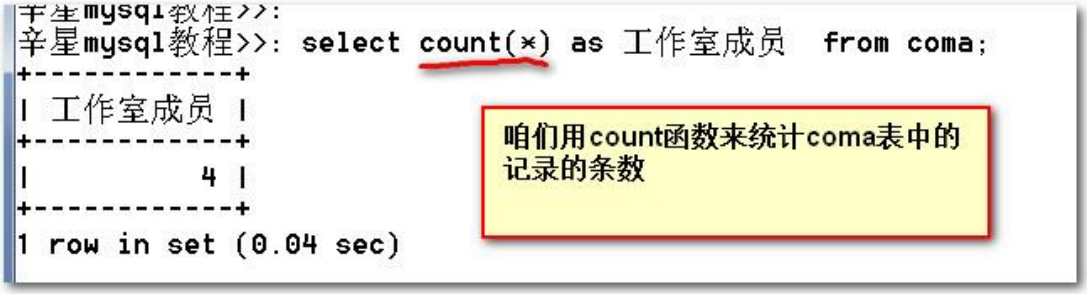
2.有时候老师虽然心里很明白，结果就是把学生给讲晕了，这就是老师表达能力的问题了，看看辛老师的表达能力怎么样。

**********聚合函数**********

1.其实我们前面接触过了，就是sum(用于求和)、count(用于统计个数)、max(求最大值)、avg(求平均值)、min(求最小值)等一系列函数，这些函数的目的是提供一些统计信息。

2.这些函数返回的都是一些固定结果，即一个确定值。

3.比如说我们要查看我们 coma 这个表中的记录的条数，就可以这么写：



4.我们接下来要讲解的 group by 和 having 都是和聚合函数密切相关的。

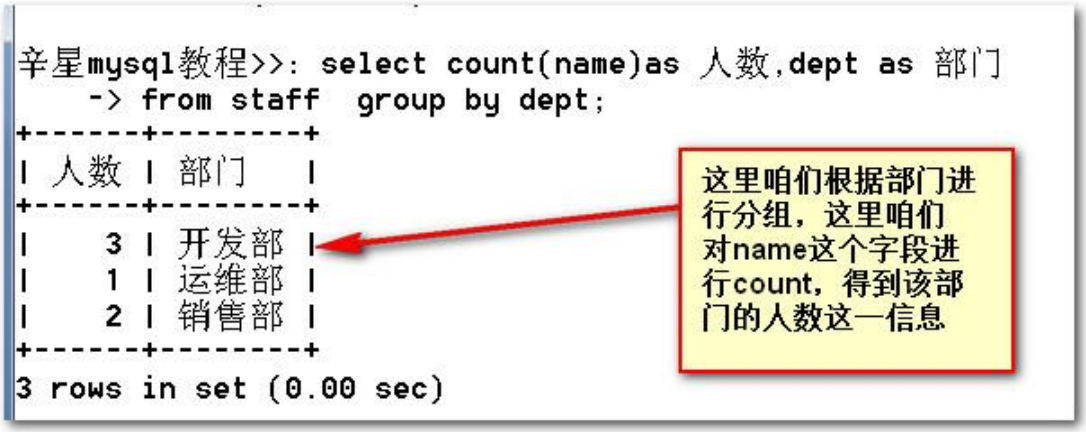
**********group by**********分组显示**********

1.有时候我们希望以分组的形式来查看某些数据， group by 就是为了解决这个需求而产生的。

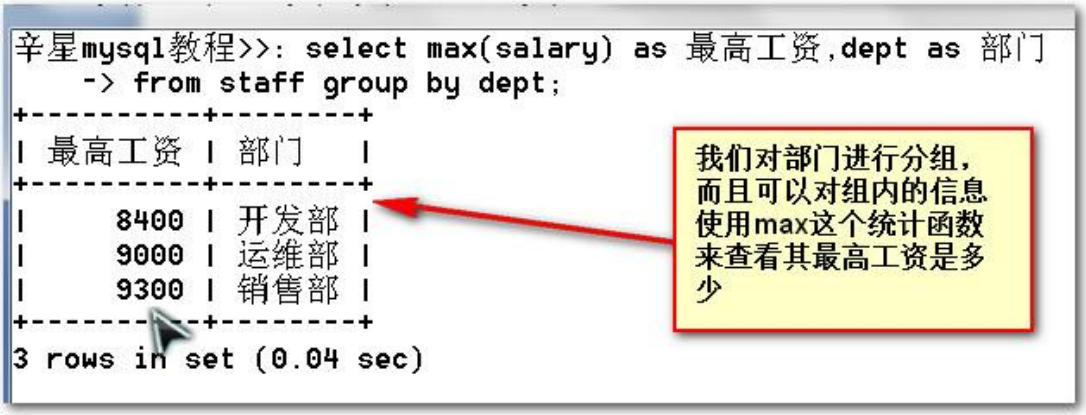
2.比如说我们按照某些字段来分类，那么其余的字段呢?答案是一般使用聚合函数，因为我们只能显示一条记录，而对于一个组中的数据一般都会多于一个。

3. 首先咱们建立了一个 staff 表：

4.然后咱们按照部门来查看它的一些信息，比如咱们看看各部门都有多少人，可以用：



5.比如说咱们想看看每个部门最高的工资是多少：

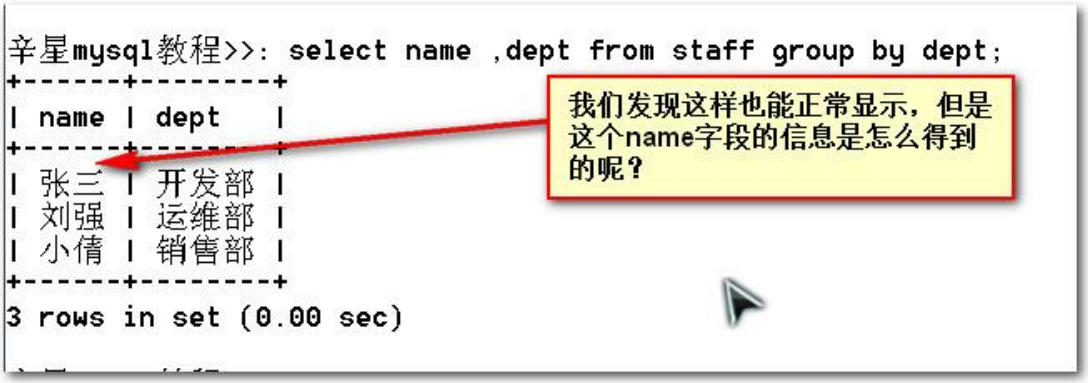


**********基本经验**********

1.咱们想根据哪个分组进行分类，就对哪个分组使用 group by。

2.咱们可以直接显示 group 语句里面分组使用的字段，对该表中的其他字段，一般需要使用聚合函数，比如 sum、max 等函数来得到其统计信息。

3.有人说，我对其他字段不使用统计函数也可以啊，比如：



4.那么有人可能会问，你这个 name 字段是怎么得到其对应信息的？答案是它会寻找同一组中的第一个 name 字段，比如对于同一个 dept 字段中都为“开发部”的几条信息，分别是“张三”、"李四","王五",之所以会显示张三，而不是李四，因为张三是第一条记录，也就是说，此时 group by 默认使用第一条记录作为默认选取的记录。

**********拓展练习**********

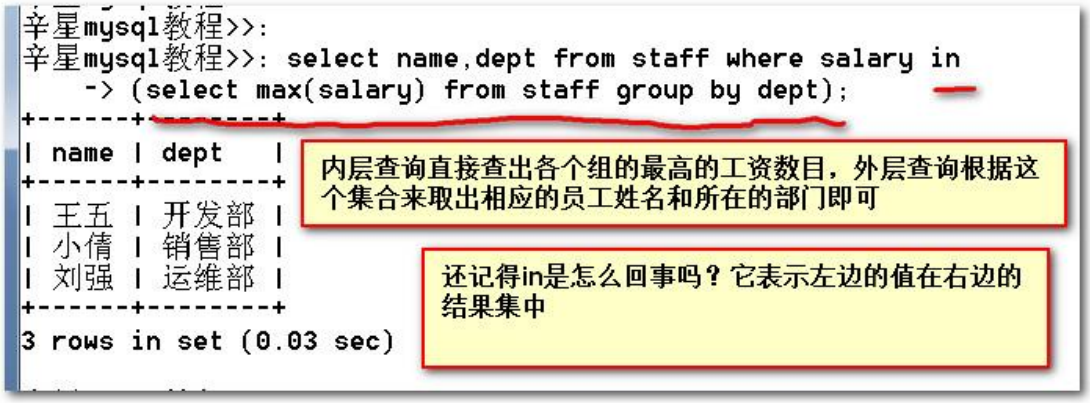
1.此时我突然想到了一点可以练习一下咱们的 group by 语句和上一节学习的子查询。

2.比如我们要查询某个部门的工资最高的员工的姓名，而不是工资数，那么该怎么写 sql 语句呢？

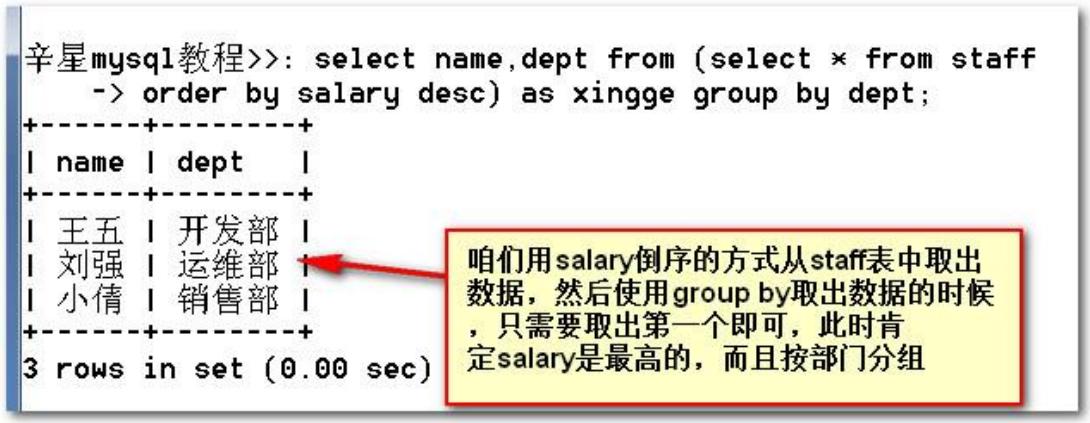
3.当然或许有人提议说咱们可以首先 select 出来工资最高的数目，然后根据此数目得到 id 和姓名，当然，此法可行，但是会造成不小的麻烦。

4.其实这就是一个 sql 语句搞定的事情，读者可以想一下。

5.哈哈，有米有想到，其实这个sql语句有好多种写法，我这里给出一个写法：



6.如果您学习过order by，或者读者学习了order by之后再回过头来，我还可以给出一个写法：



**********说明**********

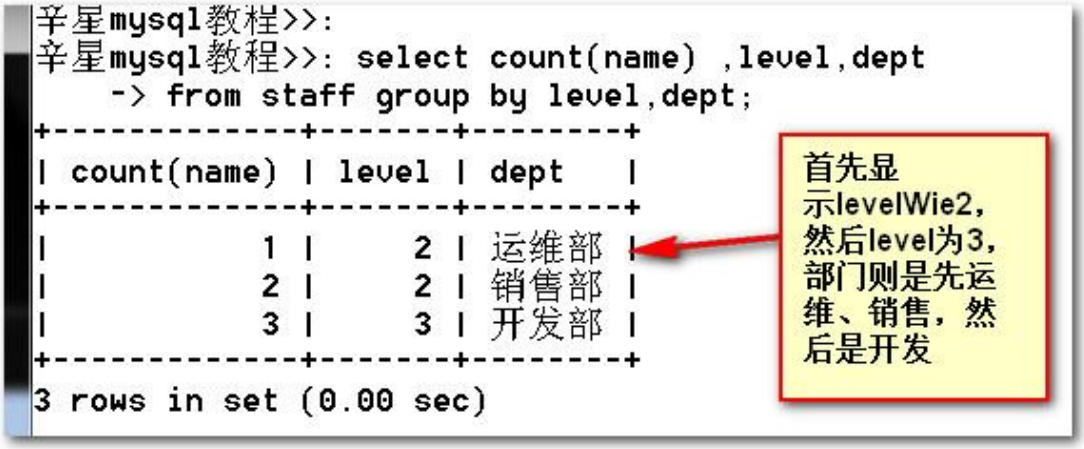
1.读者应该对分组即group有一个相对清晰的认识了吧，它是以组为单位进行信息展示的，因此，它通常对其他列使用聚合函数来表示统计信息。

2.其实group不只是用于一个字段，还可以用于多个字段的，它们彼此之间用逗号隔开。

**********group by********** 多个字段**********

1.还是以我们的 staff 表为例，我们使用多个字段进行分组。

2.比如我们首先使用 level 字段进行分组，然后使用 dept 字段进行分组：



**********having**********

1.have 即“拥有”的意思，它也是表示一些条件，而 SQL 标准要求 having 必须引用 group 子句中的列或者用聚合函数处理过后的列。

2.不过，mysql 对这一标准进行了一些扩展，它允许 having 引用 select 中检索的列和外部查询中的列。

3.我们用 having 子句可以筛选选成组之后的各组数据，因此 having 条件中也通常和聚合函数一起使用。

4.比如我们要检索出部门人数大于 1 个人的部门的名称和部门的人数，那么我们就可以在 group by 之后用一个 having count(name) ＞1，大家注意这里我们使用了 mysql 的扩展，因为我们操作的这个 name 字段并不是 group by 子句里面的，而是我们的 select 的列中的字段。

5.下面开始检索信息：

```
辛星mysql教程>>:
辛星mysql教程>>: select count(name),dept
    -> from staff group by dept having count(name) >1;
+-------------+--------+
| count(name) | dept   |
+-------------+--------+
|           3 | 开发部 |
|           2 | 销售部 |
+-------------+--------+
2 rows in set (0.00 sec)
```

咱们把人数等于1个人的运维部给干掉了

6.如果严格遵循 SQL 标准，那么我们上面的写法有点问题，不过这也是 mysql 给我们提供的扩展那，我们用的问心无愧@_@

7.下面我们写一个严格遵循 SQL 标准的 having 子句，那就是处理的这一列信息是在 group by 子句中的，比如我们除去销售部这一条记录，就可以用：

```
辛星mysql教程>>:
辛星mysql教程>>: select count(name),dept from staff
    -> group by dept having dept != '销售部';
+-------------+--------+
| count(name) | dept   |
+-------------+--------+
|           3 | 开发部 |
|           1 | 运维部 |
+-------------+--------+
2 rows in set (0.02 sec)
```

我们这里having子句中处理的是dept，而这个dept是在group 子句中指定的，因此它符合SQL标准

8.可以看出，咱们在 having 中通过 dept != '销售部';来强硬的规定不显示销售部的信息(注：销售部的小情哭了)

9.此时，容易迷惑的地方就开始出现了，我们是否可以用 having 替代 where 的作用？where 和 having 是什么关系？为什么星哥一直没有用 where？

**********where**********having**********

1.首先就是顺序问题，where 子句必须在 group 子句之前，而 group 子句又必须在 having 子句之前，否则会报语法错误。

2.第二个就是涉及的字段问题，where 可以处理一切字段，但是 having 只能处理那些在 group 语句中出现的字段、select 的列的字段、外部查询中的字段。

3.第三个就是 having 通常是和 group by 连用的，用来进一步说明。

4.可能有人会问：where 和 having 是否有时候会起到相同的作用？答案是：有可能。接下来我给大家演示一下。

5.比如我们用 having 来让它不显示销售部的信息(前面已经做了):

```
辛星mysql教程>>: select count(name),dept from staff
    -> group by dept having dept != '销售部';
+--------------+--------+
| count(name) | dept    |
+--------------+--------+
|            3 | 开发部 |
|            1 | 运维部 |
+--------------+--------+
2 rows in set (0.02 sec)
```

having子句中要求不显示那些销售部的信息

6.比如我们用 where 子句来让它不显示销售部的信息：

```
辛星mysql教程>>: select count(name),dept from staff
    -> where dept != '销售部' group by dept;
+--------------+--------+
| count(name) | dept    |
+--------------+--------+
|            3 | 开发部 |
|            1 | 运维部 |
+--------------+--------+
2 rows in set (0.00 sec)
```

咱们在where子句中规定不显示销售部的信息

7.看到区别了吗？where 必须在 group 之前，having 必须在 group 之后。

**********常见误区**********

1.新手朋友们可能会想：既然 having 也能表示条件，上面也说了有时候可以和 where 起到相同的作用，那是否可以彻底取代 where 呢？

2.答案是不可以，大家牢牢抓住 having 的定义就可以，下面的所有演示，我们评判的准则只有一个：having 中用到的条件要么在 group by 中出现，要么在 select 的列中出现，要么在外查询中出现。

3.比如我们看下面的查询是成功的：

```
辛星mysql教程>>: select * from staff having id >3;
+------+------+--------+--------+-------+
| id   | name | dept   | salary | level |
+------+------+--------+--------+-------+
|    4 | 小倩 | 销售部 |   9300 |     2 |
|    5 | 小楠 | 销售部 |   7500 |     2 |
|    6 | 刘强 | 运维部 |   9000 |     2 |
+------+------+--------+--------+-------+
3 rows in set (0.00 sec)
```

因为什么？因为这里的这个id在我们的select中的列中出现了，所以它不会出错

4.比如我们看到下面的查询是失败的：

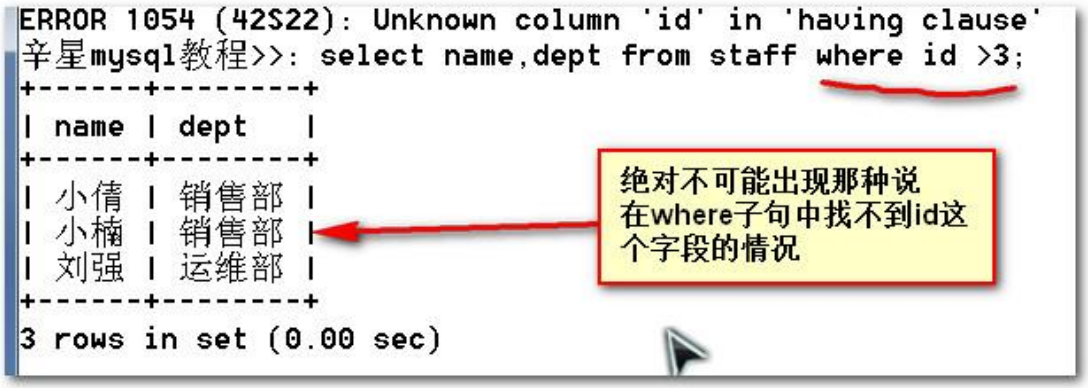
```
辛星mysql教程>>:
辛星mysql教程>>: select name,dept from staff having id >3;
ERROR 1054 (42S22): Unknown column 'id' in 'having clause'
辛星mysql教程>>:
```

原因何在，因为我们没有检索id这一列，但是我们在having中使用了这一个字段，因此，它会很迷茫，进而告诉我们说having从句中找不到id这一列

5.用我刚才给大家的准则解释一下：我们第一次之所以成功，绝对不是靠人品和运气，而是靠的是我们的 id 是 select 中的列的字段，而我们第二次检索之所以会失败，因为 id 不再 select 中的列字段中。

6.但是我们用 where 子句就不是这么回事了，where 中可用的字段在原则上并不受限制，因此，我们前面也大多数使用 where 子句，因为它足够灵活。

7.比如如下示例：



8.这是新手朋友们最容易迷茫的了，而我当年的老师解释的不够清楚。我是尽力去解释清楚的，如果您有什么疑点，请及时告诉我，这样对我们的教程的更新无疑是最能起到实际作用的。

**********图表总结**********

1.有一句话说的好“无图无真相”，虽然这图并不是高清无码的，但是，它还是有点作用的。

2.having 和 where 最容易让新手朋友们迷茫了，我根据我对朋友们的理解绘制的表格：

	where	having
顺序	在 group 之前	在 group 之后
可用字段	一切字段	①group 中的字段 ②select 列中的字段 ③外查询中的字段
通常习惯	where 一般比较喜欢被使用，因为它很灵活，可操作性强	通常用在 group by 语句之后，我也建议大家就这么用，因为这么用基本很少出错

**********经验**********

1.其实我的经验很简单，就两种情况。

2.如果我们使用 group by，可以在它后面使用 having。

3.如果我们不使用 group by，我也懒得用 having。

4.大家记住这个准则，基本就不会错了。

5.而且大家记住这个顺序是灰常重要的：where 子句>>group 子句>>having 子句。

**********小结**********

1.我们首先研究了下什么是聚合函数。

2.然后学习了下什么是 group by 子句。

3.接下来学习了什么是 having 子句。

4.然后我重点比较了下 having 和 where，因为我发现很多新手朋友们对此特别迷茫。

5.其实这些内容还是蛮基础的，如果大家读了我上面写的内容之后发现还是难以理解，那么就不要再读了。

6.此时可以求助搜索引擎，去搜索相关内容来阅读下。

第五节：order by 、limit 和 union

**********放松下**********

1.鉴于上节课内容虽然不难，但是容易让人迷茫，可能初学者还是要花费点时间去理解。

2.我们这节课决定学习点轻松的，之所以轻松，是因为它不存在任何的难理解的地方。

**********limit**********

1.我感觉 limit 子句是最好理解的了，它的作用就是四个字：拦腰截断。

2.语法格式：limit n，m

3.与 pgsql 兼容的语法：limit m offset n

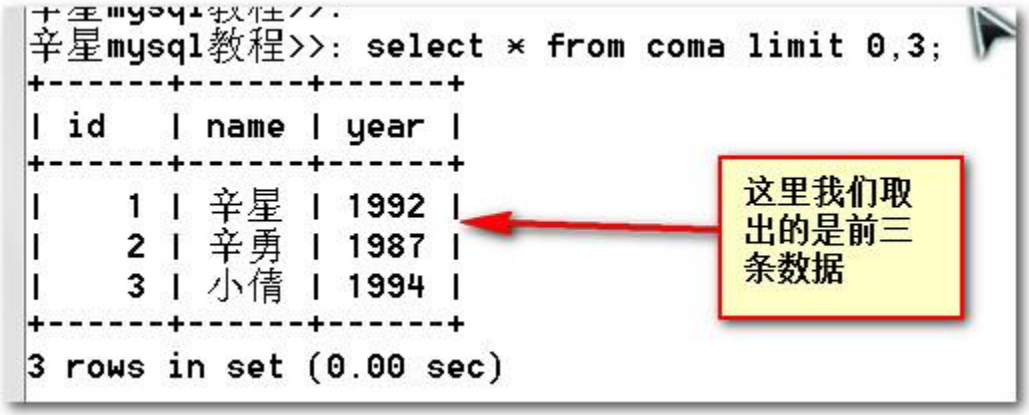
4.效果：返回从第 n 行开始的 m 条数据。

5.注意：这里的计数从 0 开始，而不是从 1 开始，我们下面会看到。

6.简写格式：limit n 相当于 limit 0，n 即从第零行开始取，取出 n 条数据。

**********实战演练**********

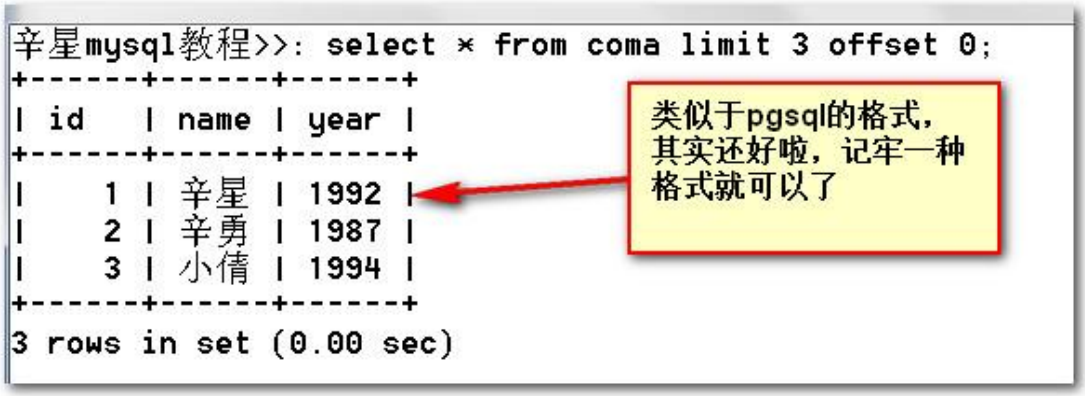
1.咱们从数据库中取出前三行数据：



2.当然使用 select * from coma limit 3；也是可以的。

3.为了与 pgsql 兼容，所谓的 pgsql 就是 PostgreSQL，是另一款伟大的开源数据库。我们的 mysql 还支持 offset 语法，注意的是，取 n 条数据就把 n 挨着 limit 写，从第 m 条数据取就把 m 挨着 offset 写。

4.下面我们用兼容语法来从数据库中取出前三条数据：



**********order by**********排序**********

1.有时候我们想对某个字段进行一下排序，那么就需要用到 order by 了。

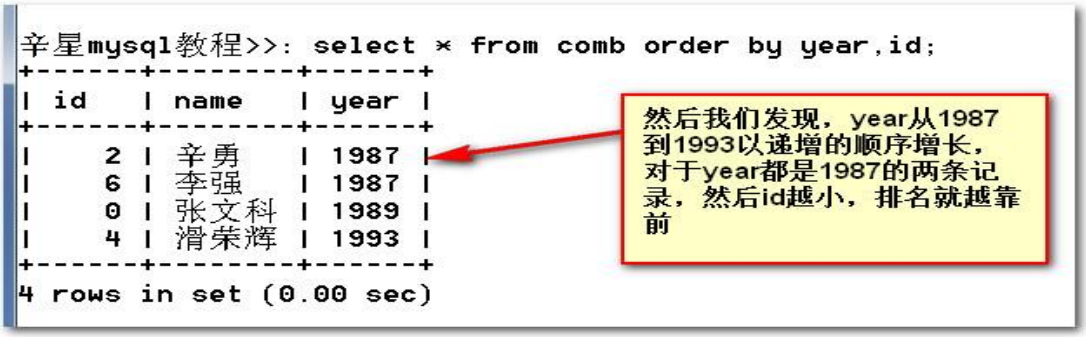
2.它的格式是这样的：order by 字段名 1,字段名 2 [desc|asc];

3.它给我们展示结果集的时候，先根据字段 1 进行排序，然后按照字段 2 开始排.....这样依次进行下去

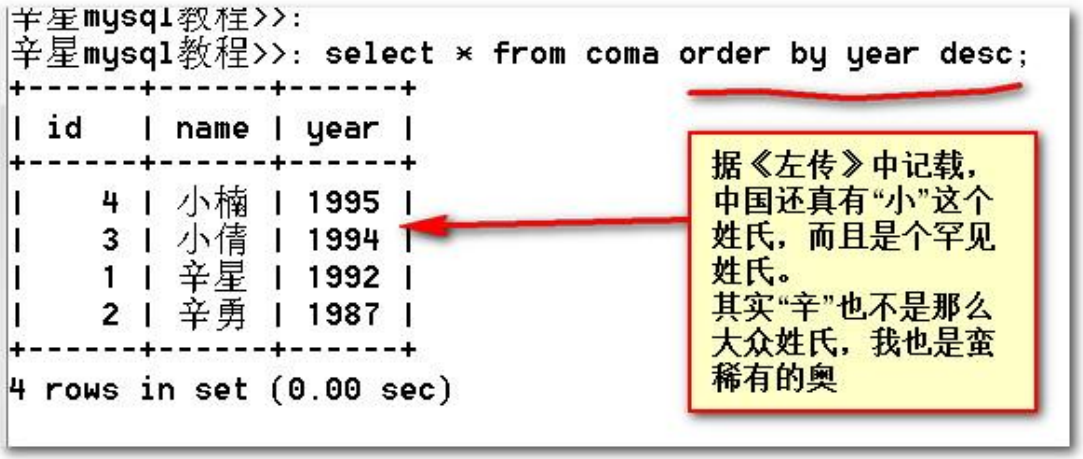
4.这里默认使用升序的，我们可以用 desc 来强制使用降序。

**********实战演练**********

1.我们对 comb 先按照 year 字段进行排列，然后按照 id 字段进行排列：



2.当然我们也可以用 desc 来指定降序排列：



**********总结**********

1.可能有人要说了：还有个 union 没讲呢，怎么就总结起来。其实它和这一次总结没什么关系，它是为了凑数才添加的内容。

2.我们前面依次学习了 where、group by、having、order by、limit 子句，没错，它们在使用的时候的顺序是很重要的。

3.我们在写一个完整的 sql 语句的时候，如果上面的几种子句都用到，那么顺序必须是：where 子句>>group by 子句>>having 子句>>order by 子句>>limit 子句。

4.上面顺序是否好记呢？我写了这么几句顺口溜，希望能帮助大家记忆：

先来范围后限制，---------->where 打头，limit 结尾

然后分组和排序，---------->group 和 order 他俩中间

最后 have 中间去。---------->having 在所有人的中间

5.我感觉从两边向中间记忆比从左向右记忆要好记的多，虽然咱们写 sql 语句肯定是从左向右写啦。

6.您有什么好方法呢，不妨邮件给我。

**********联合**********

1.很多时候我们需要涉及到多表查询，比如我们前面创建了 coma 和 comb 两个表来分别表示公司 A 和公司 B 的员工情况。

2.那我们就可以一次性的把这两个表的信息都取出来，在一个结果集中进行操作。

3.我们对两个结果集可以进行 union 操作，但是这里要求两个结果集的列的数目必须匹配。

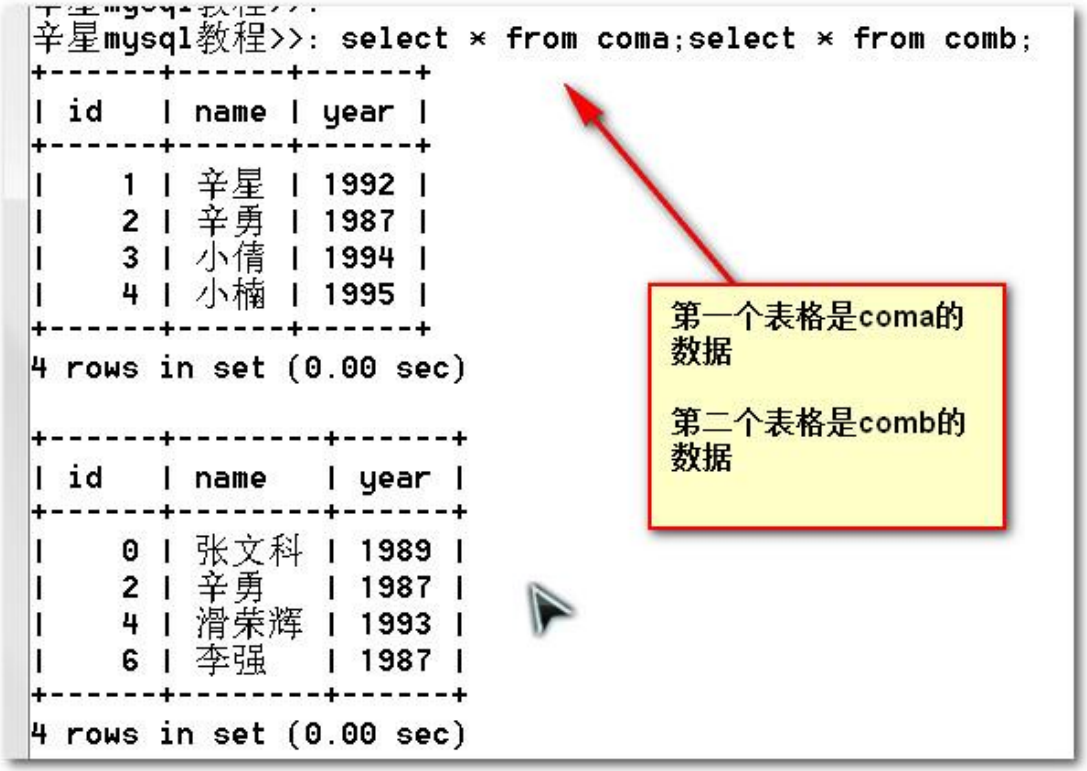
4.语法：select 语句 union [all] select 语句;

5.说明一下，我们这两个 select 语句得到的列数必须是匹配的。

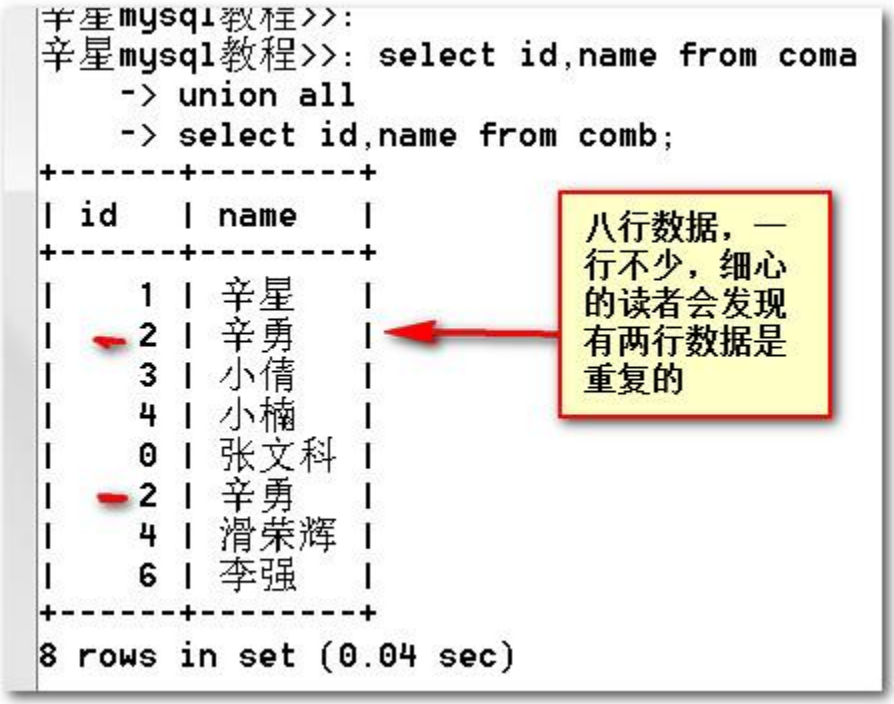
6.这里的 all 是可选的，如果加上 all，表示在结果集中保留所有数据，否则会去除重复数据。

**********union 实战**********

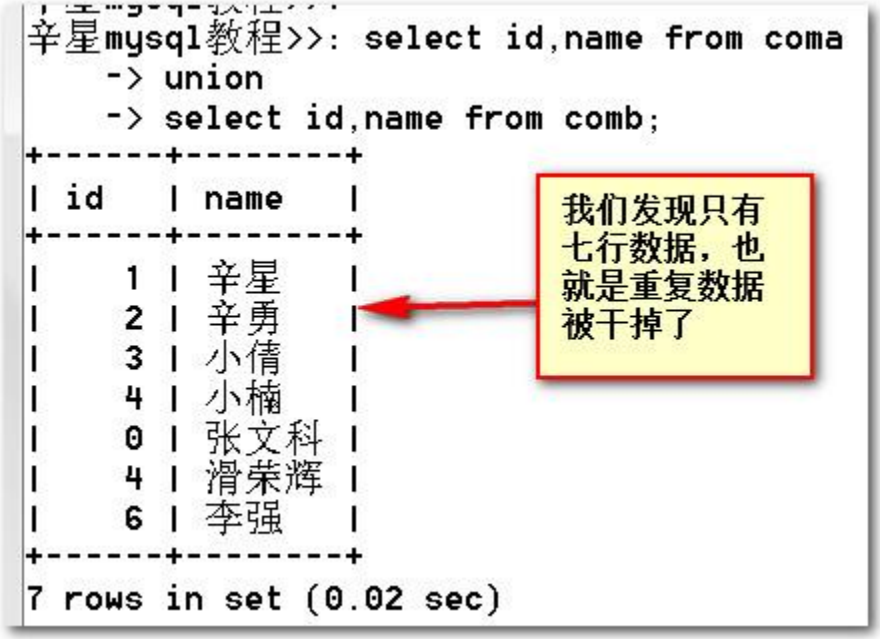
1.下面我们分别看一下 coma 和 comb 的数据把：



2.然后下面我们不妨把两个表的 id 和 name 取出来，并且用 union all 来显示所有数据：



3.如果我们使用 union，那么重复的数据就会被干掉了：



4.应该很好理解把，ok，只需要把它练熟就可以了。

第四部分：mysql 连接查询

第一节：外连接

**********说明**********

1.本来我想把连接放到一节来讲的，结果一个人给我提了个意见，建议我扩展一下内容，于是我把内容修改了几次，就成为一章了。

2.这部分究竟是作为一章还是 作为一节，我感觉都有它的意义，如果是一节的话，那就讲主干知识，如果是一章的话，那基本就是乱七八糟的什么都会介绍一遍。

3.而连接查询本身也是比较重要的一部分，倒不是他有多难，而是因为我们确实需要很多的多表操作。

**********连接**********

1.我们第三部分的查询多数是在一个表里面搞定的，可能有人说：你讲子查询那一节的时候是多表操作的。

2.没错，咱们讲连接也肯定会讲到用连接来优化子查询，因为我们用连接查询代替子查询在很多时候可以减少开销。

3.其实说到“连接”，本身的含义就是“连接多个表”，也就是多表操作。

4.连接种类很多，可以大致分为内连接、外连接、交叉连接、自连接等等吧，我们后面会逐步讲解它们，现在死扣概念也意义不大。

********** 外连接**********

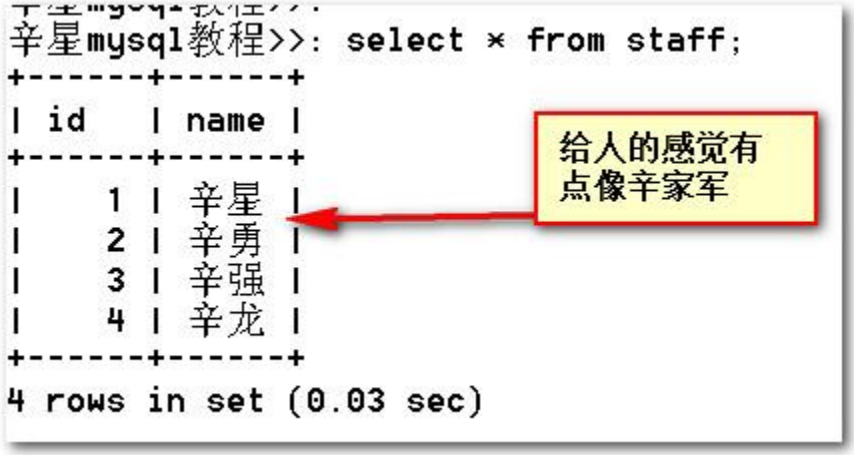
1.想给外连接一个定义，我查了一些资料，都不是很满意，于是给一个自己的定义。

2.外连接就是一个表当做基表，然后根据匹配条件来显示其他表的内容的连接。

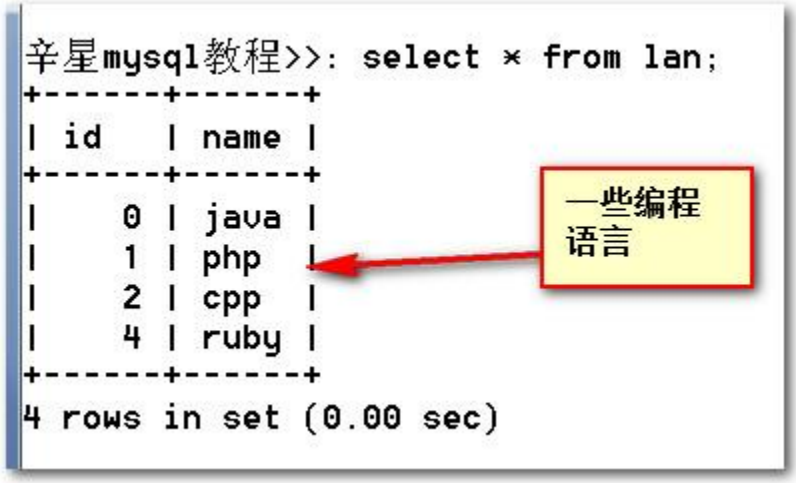
3.它通常分为左(外)连接，右(外)连接，全(外)连接。

**********建表**********

1.在咱们向前推进之前，我们首先建立两个表，第一个就是 staff 表：



2.然后就是 lan 表：

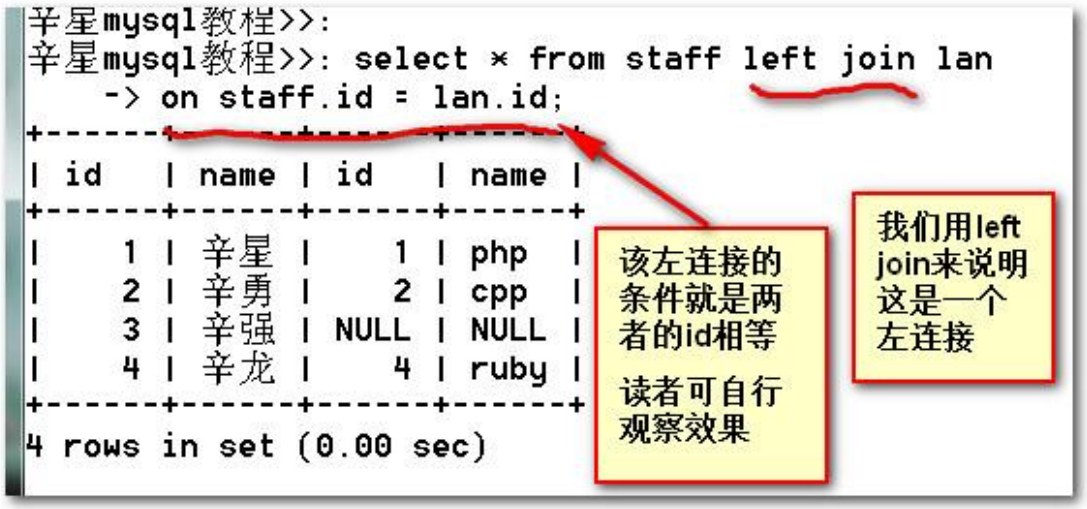


3.我在 php 教程中通过读者给我的邮件中发现，他们都不喜欢跟我建表用一样的名字，因此，我在这里说一下这两个表的用意：两个表都要有一个整型的字段就可以了，咱们待会儿用这个作为外连接的条件。

**********左外连接**********

1.接下来我们就要用左外连接了，它的查询条件是在 lan 表中找到所有和 staff 表中的 id 一样的记录。

2.那么我们可以用左外连接去实现这个功能，看我操作：



3.我们分析一下这个 sql 语句及其运行结果，首先是这个 sql 语句，它的 select * 会列出所有的字段，但是字段此时不能确定，而 staff left join lan 则确定了这是一个左连接，而且 staff 表成为基表，一旦表确定了，那么字段也就确定了，两个 id，两个 name。

4.然后就是 on staff.id = lan.id；这个条件，它说明了我们匹配的条件就是这两者的 id 相等，因此我们看到下面的结果集中，也确实是左右两边的 id 都是一样的。

5.值得注意的是，当左边 id 为 3 的时候，右边由于没有相应的内容，就是 NULL。

6.这里的 left join 就是一个左连接，它两边是两个表。

7.这里的 on 可能是让读者非常迷茫的一点，就像我们前一部分介绍的 having 一样，它的用法经常会和 where 搞不清，那么，接下来我们就分析一下。

**********on**********where**********

1.首先是顺序问题，我们前面学习 group by、having、order by 的时候就说过，顺序很重要，因为它语法就是规定的这么严格。

2.我们的 on 和 where 的先后顺序也是确定的：在同时出现的时候，on 在 where 左边。

3.二者的最大区别在于两点：第一，作用不同，on 子句用于该左连接提供一个准则，where 用于筛选结果集。

4.第二，on 子句和 where 子句的执行顺序不同，做左连接的时候，我们是首先以 on 条件筛选表，然后两个表左连接，但是对于 where 来说来说是在左连接之后再次筛选结果集。

5.可能读者到现在还不很明白，接下来咱们通过实战的方式来解读下，就很简单了。

**********实战解析**********

1.前面说过，on 和 where 的目的不同，on 是为该左连接提供一个准则，where 是左连接完成之后为筛选结果集而进行的一个条件。

2.如果我们直接用 where 进行筛选结果集，则会报错：

```
辛星mysql教程>>:
辛星mysql教程>>:
辛星mysql教程>>:
辛星mysql教程>>: select * from staff left join lan
    -> where  lan.id = staff.id;
ERROR 1064 (42000): You have an error in your SQL syntax;
辛星mysql教程>>:
```

此时该左连接无法进行

3.这里会报语法错误，因为 whee 无力操作两个表的字段。

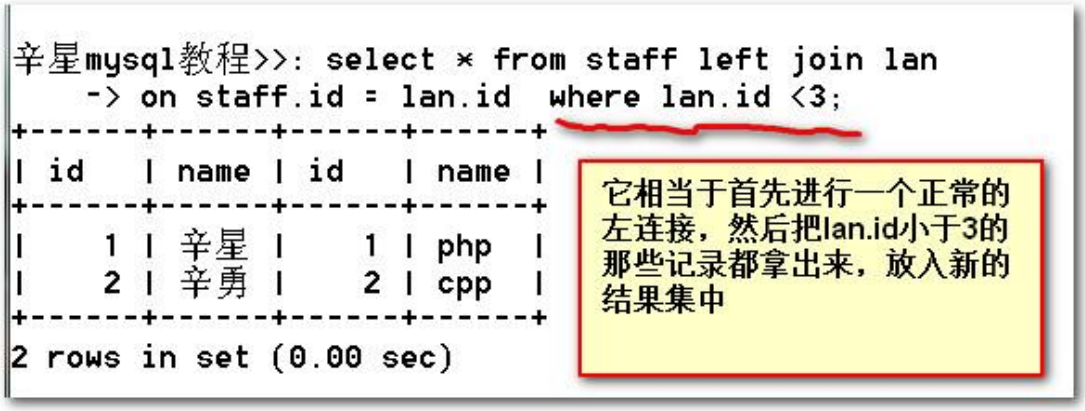
4.接下来我们看的这个就比较有意思了，看上去貌似一样的两个条件，首先在 on 子句内执行：

```
辛星mysql教程>>: select * from staff left join lan
    -> on staff.id  = lan.id and lan.id <3;
+------+------+------+------+
| id   | name | id   | name |
+------+------+------+------+
|    1 | 辛星 |    1 | php  |
|    2 | 辛勇 |    2 | cpp  |
|    3 | 辛强 | NULL | NULL |
|    4 | 辛龙 | NULL | NULL |
+------+------+------+------+
4 rows in set (0.00 sec)
```

本来lan表中有个id为4的，结果也成了NULL

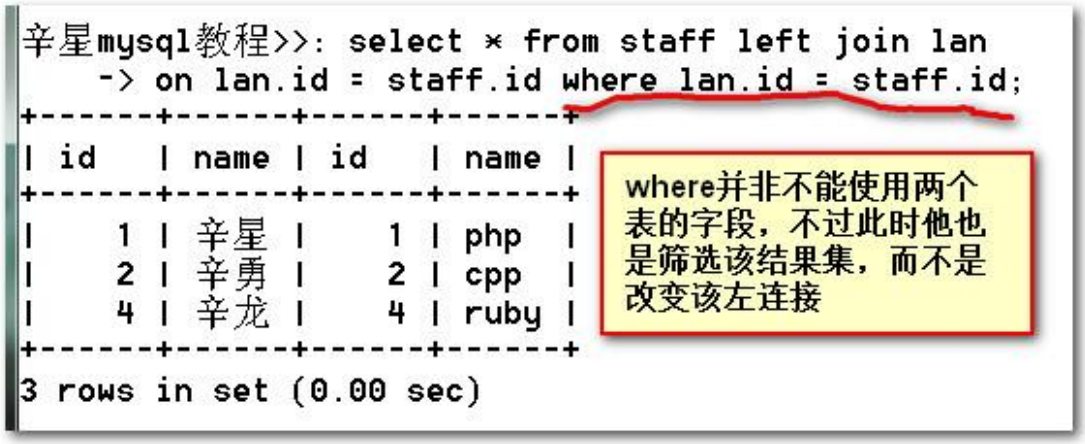
5.我们分析一下这个流程，大家记住一句话：on 子句对左连接起作用，因此，在做左连接的时候，我们发现 lan 表由于需要满足 lan.id <3,因此，它就只能贡献出来 id 为 1 和 id 为 2 的这两条记录，因此，最后结果是两条记录中含有 NULL。

6.我们再看一个 sql 语句的执行效果：



7.我们知道 select * from staff left join lan on staff.id = lan.id；是什么一个结果集，而后面这个 where lan.id < 3；就相当于对这个结果集做一次过滤，筛选出需要的结果集。

8.不知道有没有童鞋考虑过这种写法的结果集，当然我这里直接给出来了：



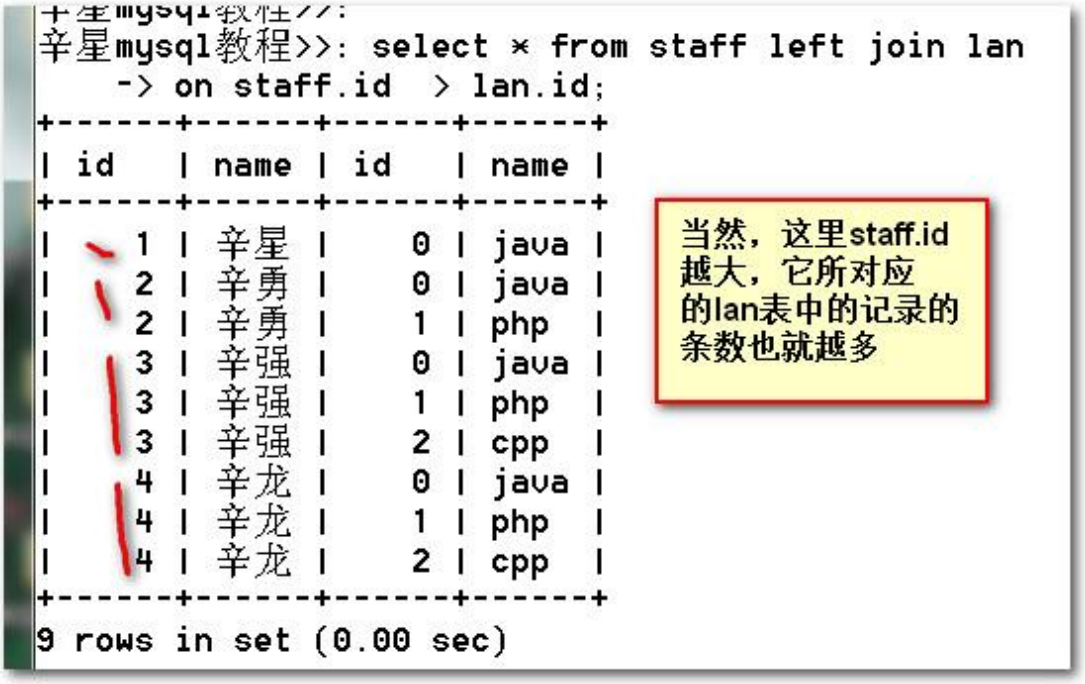
9.这里 on 子句中有 lan.id = staff.id，where 子句中也有 lan.id = staff.id，但是它们的性质却有着本质的区别。

10.on 子句是为该左连接提供一个依据，而 where 子句则是对该左连接的结果集进行一次筛选。

**********说明**********

1.可能对于老手来说，这本来不是事，但是考虑到复杂性，我还是说一下。

2.我们的 on 条件不是只能用等号，还可以用大于号等很多操作符的，下面我就使用大于号来构建左外连接：



**********建议**********

1.我不能确保读者的接受能力，如果您仔细研究了下上面的分析，感觉还是不理解，此时应该求助谷歌或者百度大神了。

2.去网上随便抓几篇这方面的博客，读一下，因为每一个博主的视角都不同，我们从多个视角去看同一个问题，往往会看的更加清晰。

**********右外连接**********

1.如果左外连接读者比较清楚了，那右外连接也就小儿科了，因为它们只是一个字不一样，即把 left 改成 right 就可以了。

2.下面看我操作：

```
辛星mysql教程>>: select * from staff right join lan
    -> on lan.id = staff.id;
+------+------+------+------+
| id   | name | id   | name |
+------+------+------+------+
| NULL | NULL |    0 | java |
|    1 | 辛星 |    1 | php  |
|    2 | 辛勇 |    2 | cpp  |
|    4 | 辛龙 |    4 | ruby |
+------+------+------+------+
4 rows in set (0.00 sec)
```

此时以右表为基准表，用左边的表来匹配右边的表

**********全连接**********

1.全连接就是把左连接的 left 换成 full 即可。

2.但是 mysql 并不支持全连接，执行效果如下：

```
辛星mysql教程>>:
辛星mysql教程>>: select * from staff full join lan
    -> on staff.id = lan.id;
ERROR 1054 (42S22): Unknown column 'staff.id' in 'on clause'
辛星mysql教程>>:
```

mysql报错说无法找到相应的列信息，很抱歉。。
ㄟ(ˊ^ˋ)ㄏ

3.既然它不支持，那就不说了，等写别的数据库教程的时候再说。

**********总结**********

1.其实我们说左连接、外连接就是指的是左外连接、右外连接，因为不可能有左内连接等名称的出现。

2.左外连接就是以左表为基表，以右表为附表，然后根据条件来选择性的附加到基表中。

3.右外连接就是右表为基表，左表为附表。

**********on*********where**********

1.我发现大家单独使用 where 基本都没有疑问，但是与 on、having 一起用就开始有人犯晕了，如果读者对 having 不熟悉了，不妨去前面翻翻，我可是讲的很详细奥。

2.下面我们这个表格来说明主要区别：

区别类型	on	where
主要目的	用该条件进行外连接操作	用该条件来进行筛选结果集
执行时间	它的执行在构建外连接的时候	它的执行在外连接构建完毕，在筛选结果集的时候
位置	on 在 where 的左边	

**********经验**********

1.考虑到 on 与 where 的启用时机不同，而且作用范围也不同，因此，我们什么时候该使用什么，其实基本是确定的。

2.on 子句中只写构建该连接的条件，其他一切都放到 where 里面就 ok 了。

**********备注**********

1.前面忘记说了，我也是刚想起来。

2.咱们用 left join 的时候可以换成 left outer join 的，其意义是一样的，都表示左外连接，只是这里的 outer 可以省略，因为不用说它是外连接，它也肯定是外连接。

第二节：其他连接

**********交叉连接**********

1.所谓交叉连接，就是表与表之间进行笛卡尔乘积。

2.交叉连接可以使用 cross join，也可以什么都不用，直接逗号分隔即可。

3.实战演示：

```
辛星mysql教程>>: select * from staff,lan;
+------+------+------+------+
| id   | name | id   | name |
+------+------+------+------+
|    1 | 辛星 |    0 | java |
|    2 | 辛勇 |    0 | java |
|    3 | 辛强 |    0 | java |
|    4 | 辛龙 |    0 | java |
|    1 | 辛星 |    1 | php  |
|    2 | 辛勇 |    1 | php  |
|    3 | 辛强 |    1 | php  |
|    4 | 辛龙 |    1 | php  |
|    1 | 辛星 |    2 | cpp  |
|    2 | 辛勇 |    2 | cpp  |
|    3 | 辛强 |    2 | cpp  |
|    4 | 辛龙 |    2 | cpp  |
|    1 | 辛星 |    4 | ruby |
|    2 | 辛勇 |    4 | ruby |
|    3 | 辛强 |    4 | ruby |
|    4 | 辛龙 |    4 | ruby |
+------+------+------+------+
16 rows in set (0.00 sec)
```

如果是一个保留全部记录的结果集，那么这个数据量就是所有表的数据的乘积，数据量会瞬间提高几个层次

4.还记得我们上面说过什么吗？where 可以筛选结果集，我们得到的这个结果集有 16 条记录，即 4x4 得到的，那么当每个表数据量较大的时候，这个结果集也会增加的非常迅速，因此，我们可以用 where 条件来筛选很大一部分内容。

5.我们可以只保留那些 staff 的 id 比 lan 的 id 大 1 的数据，于是就有了下面的 sql 语句：

```
辛星mysql教程>>:
辛星mysql教程>>: select * from staff,lan
    -> where staff.id = lan.id + 1;
+------+------+------+------+
| id   | name | id   | name |
+------+------+------+------+
|    1 | 辛星 |    0 | java |
|    2 | 辛勇 |    1 | php  |
|    3 | 辛强 |    2 | cpp  |
+------+------+------+------+
3 rows in set (0.03 sec)
```

我们看到的这三条记录都是满足staff.id = lan.id + 1的

6.可能有人会问：交叉连接主要有什么用呢？答案就是：交叉连接的主要作用就是可以在所有的结果集中进行筛选信息，它基本包含了最全的信息类型，包括咱们上一节讲到的左右连接，它们的结果集都可以在交叉链接里找到。

**********补充*********交叉连接**********

1.交叉连接可以用于多个表，比如下面我们就用了三个表：

```
辛星mysql教程>>:
辛星mysql教程>>: select color,height,width from
    -> color cross join height ,width;
+-------+--------+-------+
| color | height | width |
+-------+--------+-------+
| black |    200 |    30 |
| red   |    200 |    30 |
| black |    190 |    30 |
| red   |    190 |    30 |
| black |    200 |    45 |
| red   |    200 |    45 |
| black |    190 |    45 |
| red   |    190 |    45 |
+-------+--------+-------+
8 rows in set (0.00 sec)
```

第一组里面的color、height、width是字段，第二组是表，它们概念不太一样

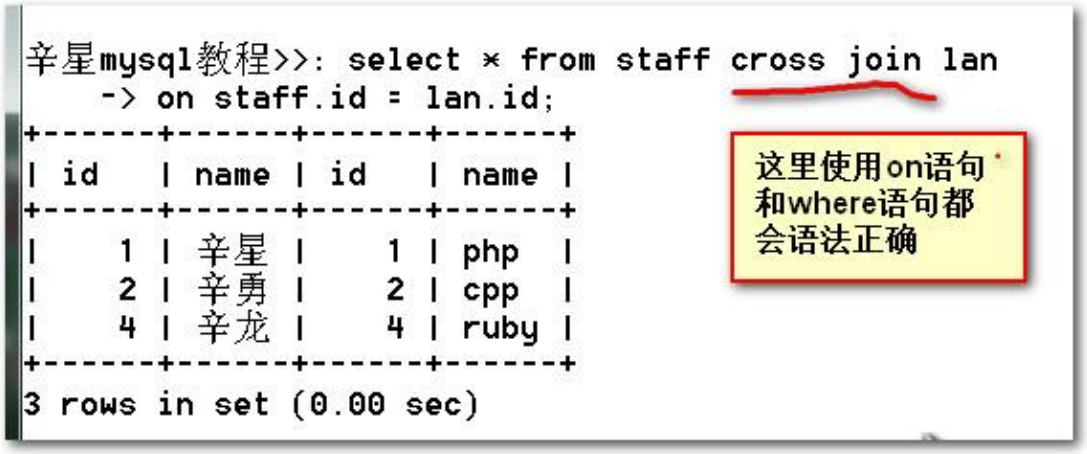
我们使用cross join和逗号 都可以对表进行交叉连接

2.这里的 color、height、width 三个表都含有一个和表名一样的字段名。而且第一个交叉连接咱们使用的 cross join，第二个使用的逗号。

**********逗号与 cross join 的区别**********

1.可能有人会问：逗号与 cross join 在表示两个内连接的时候有无区别？

2.答案是有区别，至少在表现形式上有区别。当我们使用 cross join 的时候使用 on 语句：



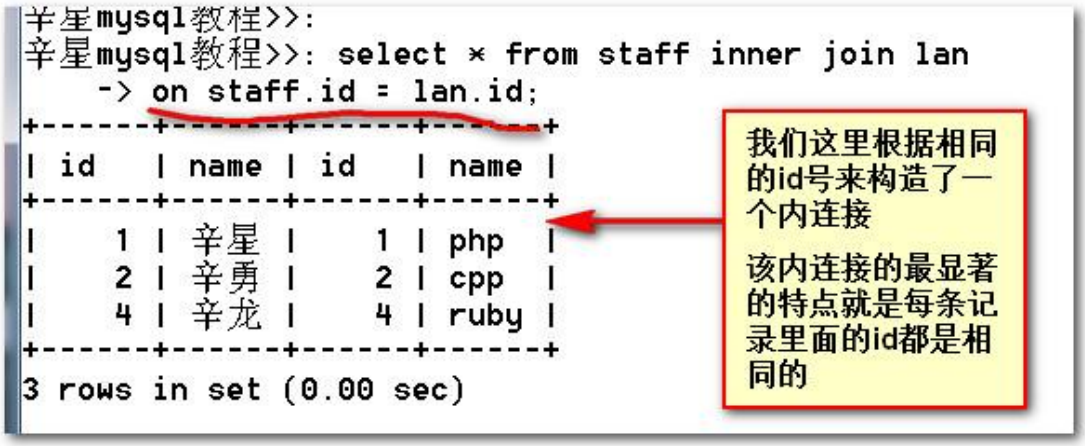
3.但是当我们使用逗号的时候使用 on 语句就会出现以下问题：

**********内连接**********

1.有些童鞋对于内连接、交叉连接、自然连接等概念非常迷茫，就让辛博士给您分析一下。

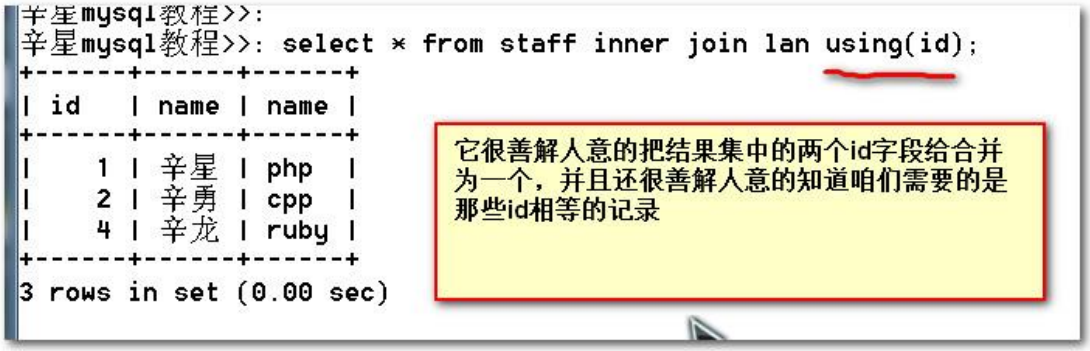
2.内连接咱们使用 inner join 来表示，比如表 a 和表 b 进行内连接运算，就是 a inner join b。

3.内连接是最均衡的连接，它可以使用一个 on 条件来指定构造该连接的条件，有人认为既然是内连接，就应该使用等号来连接，确实，使用等号连接的情况是最多而且是最普遍的，比如如下：



4.既然咱们使用等号的情况那么多，并且此时的 id 还是一样的，都叫“id”，那么何不提供一种语法来简化它的操作呢？于是就诞生了 using 这个语法，它的使用就是把 on 子句换成 using(字段名) 即可。

5.看我操作：



6.那么这里的 using 什么作用呢？它的作用很简单：第一点就是把结果集中的同名的 id 合并为一个，第二点就是根据每个表中的 id 值相同的设置为同一条记录，并且放入该结果集中。

7.由于 using 的存在，很多人就认为咱们的内连接只能是让某些字段相等，其实这种说法是比较片面的，不过它是主要用途。

8.下面我们看一个字段不等的例子：

```
辛星mysql教程>>: select * from staff inner join lan
    -> on staff.id > lan.id;
+------+------+------+------+
| id   | name | id   | name |
+------+------+------+------+
|    1 | 辛星 |    0 | java |
|    2 | 辛勇 |    0 | java |
|    3 | 辛强 |    0 | java |
|    4 | 辛龙 |    0 | java |
|    2 | 辛勇 |    1 | php  |
|    3 | 辛强 |    1 | php  |
|    4 | 辛龙 |    1 | php  |
|    3 | 辛强 |    2 | cpp  |
|    4 | 辛龙 |    2 | cpp  |
+------+------+------+------+
9 rows in set (0.00 sec)
```

我们这里使用了staff.id > lan.id来得到的这一系列的结果固然满足我们的要求，但是不知道读者是否看出了一点问题

9.然后有些读者朋友感觉这和交叉连接太像了，于是就直接不加条件试一试，这一试，就有了下面的结果：

```
辛星mysql教程>>: select * from staff inner join lan;
+------+------+------+------+
| id   | name | id   | name |
+------+------+------+------+
|    1 | 辛星 |    0 | java |
|    2 | 辛勇 |    0 | java |
|    3 | 辛强 |    0 | java |
|    4 | 辛龙 |    0 | java |
|    1 | 辛星 |    1 | php  |
|    2 | 辛勇 |    1 | php  |
|    3 | 辛强 |    1 | php  |
|    4 | 辛龙 |    1 | php  |
|    1 | 辛星 |    2 | cpp  |
|    2 | 辛勇 |    2 | cpp  |
|    3 | 辛强 |    2 | cpp  |
|    4 | 辛龙 |    2 | cpp  |
|    1 | 辛星 |    4 | ruby |
|    2 | 辛勇 |    4 | ruby |
|    3 | 辛强 |    4 | ruby |
|    4 | 辛龙 |    4 | ruby |
+------+------+------+------+
16 rows in set (0.00 sec)
```

有木有发现此时内连接和交叉连接取得的结果是一样的？

**********澄清混乱**********

1.我们知道 cross join 和 inner join 在什么条件都不加的情况下结果是一样的。

2.而且 cross join 可以用 on，也可以用 where，最后得到的结果集是一样的，这点 inner join 同样也适用。

3.可能有人会问了：不是说交叉连接和内连接不同吗？为什么我感觉相同呢。

4.我只能很明确的告诉你：在 mysql 中，内连接和交叉连接是一回事，其他数据库中不见得是这样。

5.很多一开始研究其他数据库的转到 mysql 的时候，会在这里迷茫好久，不过这是 mysql 内部的设定，没办法，只能服从。

**********内连接**********

1.mysql 的内连接 inner join 分为如下三类。

2.第一类就是等值连接，所谓等值连接就是在 on 子句中使用等号来连接，这里的 on 子句不要求两边的字段是相同的，比如 on table1.uid = table2.gid 都是可以的，这一类咱们上面有介绍。

3.第二类就是不等连接，所谓不等连接就是在 on 子句中不使用等号来进行连接，貌似它的选择更加丰富，但是实际上它的应用并不多，因为咱们需要的数据往往是相等居多。

4.第三类就是自然连接，很多书籍上把自然连接和内连接划分为平级的类别，这些书籍大多数都不是专门介绍 mysql 的，我们下面会讲解下自然连接。

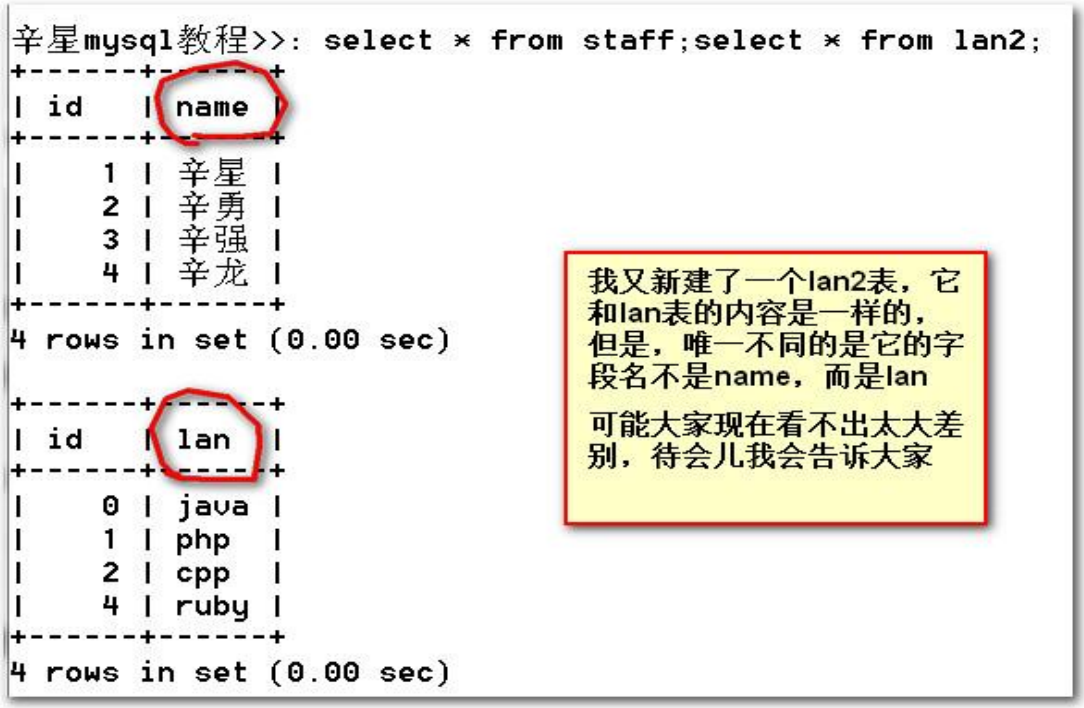
**********自然连接**********

1.自然连接(natural join)是一种特殊的等值连接，即在结果集中把重复的属性列去掉。

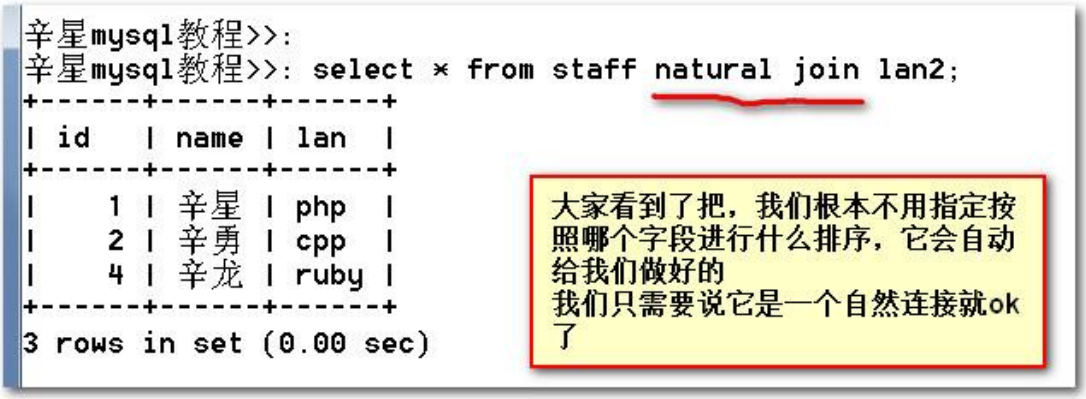
2.自然连接直接使用 natural join 连接即可，无需指定连接时的一些操作，因为这些都是“自然进行”的。

3.从这点上看，只要我们使用了 using，它和自然连接的效果是一样的。

4.这里我们先看两个表格：

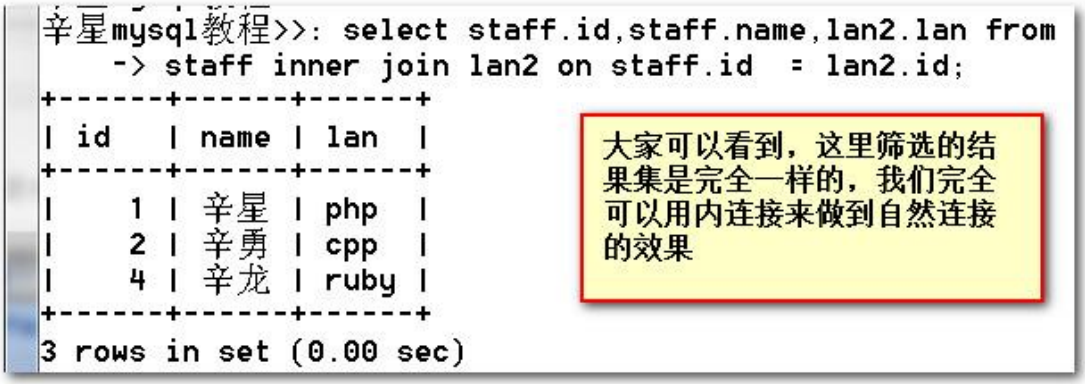


5.然后我们执行一下自然连接：



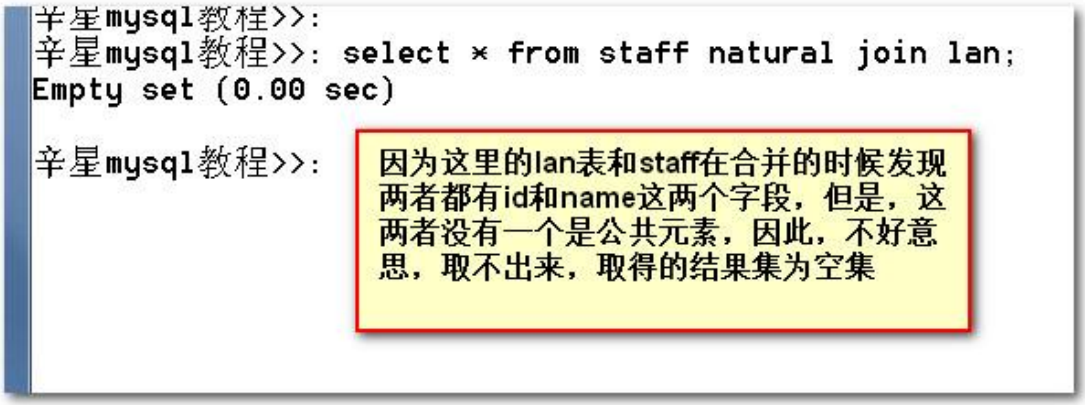
6.其实我们可以理解为这个自然连接：select * from staff natural join lan2；就是等值连接：select * from staff inner join lan2 using(id)；的简单写法，还可以认为它是 select staff.id，staff.name，lan2.lan from staff inner join lan2 on staff.id = lan2.id；的简写形式。

7.下面是执行最后一种写法的截图：



**********分析 lan2 与 lan**********

1.我们发现 lan 表与 lan2 表只有一个字段的名称上的不同，这也就直接导致了 lan 表与 staff 表的自然连接会失败：



2.但是对于 lan2 和 staff 做自然连接就不同了，因为它们的公共字段只有 id 这一个公共字段，因此只需要按照该字段让两个表进行等值连接并且删除重复列即可。

3.可能有人说：难道两个字段相同就无法进行自然连接吗？答案是否定的，比如我们为 staff 表和 lan 表都增加一条公共记录(8,“辛星 mysql”)，那么我们就可以取出它来了。

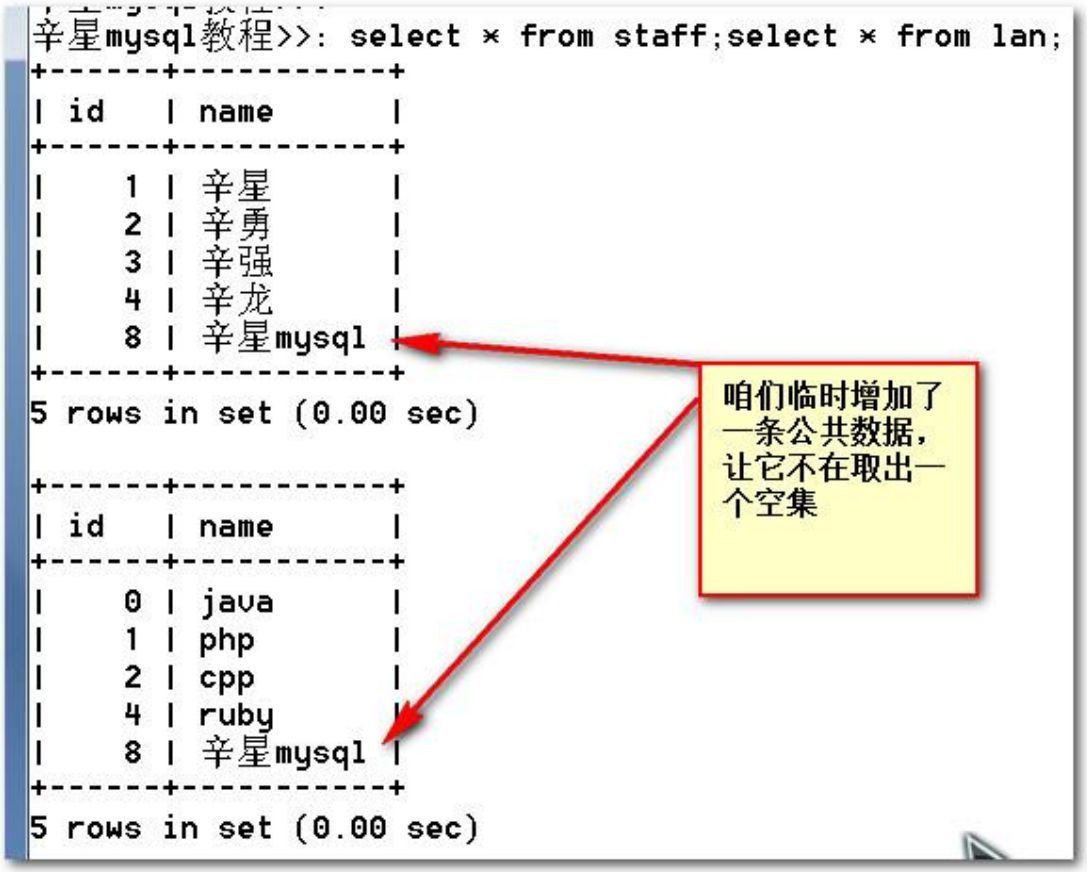
4.这里读者可以自行分析一下，我们自然连接会按照所有的公共字段作为一条记录的判别标准，比如两个表都有 id 字段和 name 字段，那么必须要求这两个表的 id 字段和 name 字段都得一样才行。

5.首先是运行结果：

```
辛星mysql教程>>: select * from staff natural join lan;
+------+-----------+
| id   | name      |
+------+-----------+
|    8 | 辛星mysql |
+------+-----------+
1 row in set (0.00 sec)
```

对natural和lan表进行两个自然连接

6.然后是 lan 表和 staff 表的现有数据：



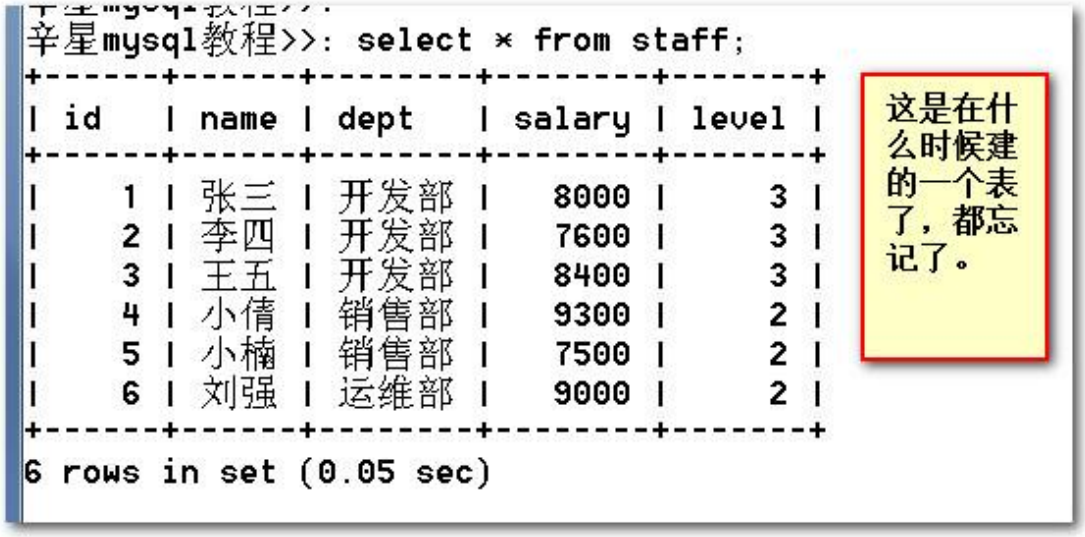
7.我说完了，读者可以自行研究下。

**********自连接**********

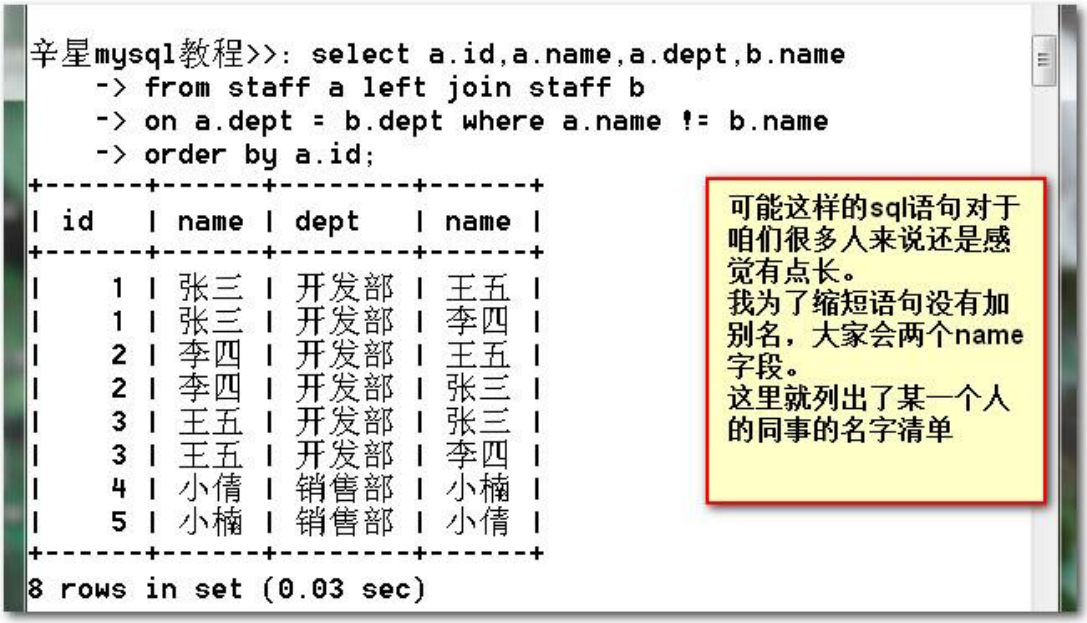
1.所谓自连接，就是自己和自己连接，就相当于把一个表当成两个表来用。

2.那么自连接有什么意思呢？其实自连接就是在本表内找一些信息，但是不使用连接又无法找到(不考虑子查询)。

3.比如说有这样一个表：



4.那么我们要从里面列出某一个成员的所属相同部门的同事，就可以采用如下的 sql 查询：



5.上面咱们这个 staff a left join staff b 中的 a 和 b 都是表的别名，之所以要取别名，是因为咱们用 staff 很难区分哪个表是基表，哪个是附表，因此咱们用别名的方式来好像是使用了 a、b 这两个新表一样。

6.上面的语法倒不难，大家仔细分下应该没问题。

第三节：小结以及若干说明

**********小结**********

1.咱们前面没有总结，那肯定不是我的风格，咱们这里总结一下。

2.下面这个图表给我们进行了一些总结：

大类型	小类型	一句话说明
外连接	左外连接	左表为基表，右表为附表来填充
	右外连接	右表为基表，左表为附表来填充
	全外连接	mysql 不支持
内连接	均等连接	使用＝运算符进行的内连接
	不等连接	不使用＝运算符进行的内连接
	自然连接	去除重复列的均等连接
交叉连接		mysql 中，交叉连接就是内连接
自连接		自身和自身的连接都叫自连接

**********说明**********

1.内心的矛盾：本来我一开始只是想讲讲简单的连接操作，然后把大多数内容移到 mysql 进阶那本书里面去。

2.但是和某些人交流了下，它们说还是在这里写的多一点把，因为他感觉这部分内容比较难。

3.我想了半天，最后感觉我们最基本的操作已经讲完了(前两节)，接下来就是技巧性的东西了，比如如何高效的使用连接查询，如何用连接查询代替子查询，以及 mysql 进行查询的一些算法等等。

4.因此我这里果断的决定，把这些技巧性的东西转移到进阶或者优化那一本中，而让这本书更加基础，更能体现“简单 入门”的特点。

第五部分：数据类型

第一节：数据类型简介

**********为何需要数据类型**********

1.第三代编程语言普遍支持“数据类型”，比如 Pascal 支持“记录”，比如 C 支持“指针”，比如 Python 支持“字典”，比如 PHP 支持“资源类型”，比如 Java 支持“类”，当然诸如“字符串”、“整型”、“浮点型”、“布尔类型”是基本绝大多数编程语言都会支持的，因为它们太常用了。

2.虽然它们风格迥异，实现原理也不尽相同，但是究其根本原因，是为了方便开发者的使用。

3.我们 mysql 为了存储数据的时候达到更好的效率，对数据类型的划分可能和编程语言有较大区别，它更多的是用来确定存放使用的空间大小和类型。

**********mysql 的数据类型**********

1.mysql 共计支持 27 种数据类型，可能有些人听到之后立即晕掉了，说：那不得学十天半月的？其实，远没那么复杂，我整理了一个表格，大家先来个初印象。

2.mysql 数据类型分类表格：

所属类别	具有数据类型数目	功能
整数类型	tinyint、int 等 5 种	存储整数数据
浮点类型	float、double 等 3 种	存储小数数据
字符类型	char、varchar 等 12 种	存储文本信息
时间日期类型	date、time 等 5 种	用于记录时间
复合类型	enum 和 set 共 2 种	用于其他功能

**********说明**********

1.数据类型这一块对于数据库建模的作用尤其明显，不过鉴于数据库建模本身就比较重要，考虑的要素比较多，一般也不会让新手去做，所以，咱们 mysql 光速入门这一本不会提到数据库建模的原理。

2.选择一个合适的数据类型是灰常重要的，比如存储一个 ip 地址，咱们以南开大学选课系统为例，它的 ip 是“222.30.32.10”，那么这里咱们存储明文，不必考虑加密。

3.这里读者还没有接触具体类型的话，可能看起来有点吃力，不过先大概了解下，等学完本部分，也可以再回过头来看。

4.第一种有人说咱们存储一个整数“222030032010”就可以了，那么用 bigint，占据八个字节。第二种认为咱们存储一个字符串“222.30.32.10”，那么使用 char 和 varchar 有些区别，总体来说占据字节数从 7 到 15 个不等。

5.第三种认为 ip 地址的每一个数据可以转化为两个 16 进制数，于是该 ip 可以存储为 de1f200a，它此时需要 8 个字节。

6.第四种认为我们存储 3726516234 就可以了，它只需要存储为 int 即可，占据四个字节。可能有人很纳闷你这个数据怎么计算来的，其实也很简单，它就是通过：
222*256*256*256+30*256*256+32*256+10 这样计算得到的，然后我们转化为实际 ip 的时候通过取商和余数即可。

7.下面的列表可以反映出存储同样的信息占用的空间大小：

存储方式	占用空间
整体字符串存储	7 个字节到 15 个字节
整体整数存储	8 个字节
伪 16 进制字符串	8 个字节
伪 256 进制整数	4 个字节

8.当然啦，具体还可以使用的方法有很多，这里列出了几种比较好想到的，可以看出，从 8 个字节到 4 个字节，存储效率差了一倍，随着数据量的提升，它的性能一下子就看出来了。

第二节：数值类型

**********数值类型**********

1.所谓数值类型，就是存储实数的类型，我们通常可以分为两类：整数类型和浮点数类型。

2.许多不同的子类型对于这些类别中的每一个都是可用的，每个子类型支持不同大小的数据，并且 mysql 允许我们指定数值字段中的值是否有正负之分或者用零填补。

**********整数类型**********

1.mysql 支持五种整数类型：tinyint、smallint、mediumint、int 和 bigint。

2.这些类型在很大程度上是相同的，只是它们根据所占空间大小的不同会导致存储的值的范围不同。

**********具体类型**********

1.我们前面说过 int 有五种类型，现在具体剖析一下。

2.下面是这五种类型：

类型名称	占据字节数
tinyint	1
smallint	2
mediumint	3
int(或者 interger)	4
bigint	8

3.它的每一种类型都可以通过修饰符 unsigned 指定为无符号数据，否则默认是有符号的。

4.如何计算某种数据类型的取值范围，我会在下面详细讲解下，这里先不说了。

**********有符号*****无符号**********

1.如果读者学习过 C 和 C++等编程语言，那么对有符号和无符号非常了解，如果读者只是学习过 PHP 或者 Python，那就对这方面可能不够，我讲一下。

2.所谓无符号数，字面意思是无正负之分，实际是不包含负数，只包含 0 和正数。

3.所谓有符号数，就是即包括整数，也包括负数，当然还有 0.

4.上面是定义上的区别，下面我们来看一下它在应用上的区别。

5.比如我们有 1 个字节的空间，一个字节是八位，可以存储 256 种状态，我们可以对应为 256 个整数，比如 00000000 表示存储数据为 0,比如 00010001 表示存储数据为 17 等等。

6.一个字节的空间，可以代表 256 个数据，如果存储为有符号的，则范围是从-128 到 127，如果存储为无符号的，则范围从 0 到 255.

7.也就是说，同样的存储空间，使用有符号和无符号这两种规则导致的存储的数据的大小是不同的。

**********根据存储类型**********计算存储范围*****

1.其实我们上面了解了有符号和无符号的区别，接下来我们计算下具体的数据类型应该怎么计算其大小范围。

2.比如 tinyint 是 1 个字节(byte)，也就是 8 位(bit)如下表：

1	0	0	1	0	1	1	0

由于每一位都有 0 和 1 两种存储状态，因此它的值共计有 2 的 8 次方种状态，也就是 256 种状态。

3.对于无符号来说，存储的范围最大值就是256减去1，为什么要减去1呢，因为我们必须分配一种情况给0，即存储范围从0到255.

4.对于有符号数来说，我们通常采用的是补码来表示，读者可以去搜索下，很简单，我这里不深入讲了。

5.对于1字节的数据，可以存储256种情况。如果是有符号数，那么它的存储范围是分成负数和非负数两部分的，负数那部分就是256除以2加一个符号，即从-1到-128，非负数那部分从0到127。

6.总之，我们分成三步：

第一步，根据所占字节N求得情况数m，其中m＝2的N次方。

第二步，对于无符号数，范围从0到m-1.

第三步，对于有符号数，负数部分范围从-m/2到-1，非负数部分从0到(m/2)-1,也就是说整体范围为(-m/2,(m/2)-1)

**********修饰符**********

1.mysql在整数类型后面可以跟一个宽度指示器，它可以把显示的数据值加上到指定的长度来显示，比如int(12)就表示不足12位的数字在屏幕上占据12个数字的位置。

2.注意的是，该宽度指示器并不会影响int的取值范围，我们上面也讲了，int的取值范围是由它所占数据空间的大小决定的，而不是由该宽度指示器决定。

3.该宽度指示器只是影响给我们显示的格式，而且默认使用空格填充，比如int(12)如果显示23，那么先输出十个空格，然后输出23这个数据。

4.我们可以使用unsigned修饰符来规定字段只保存正值。

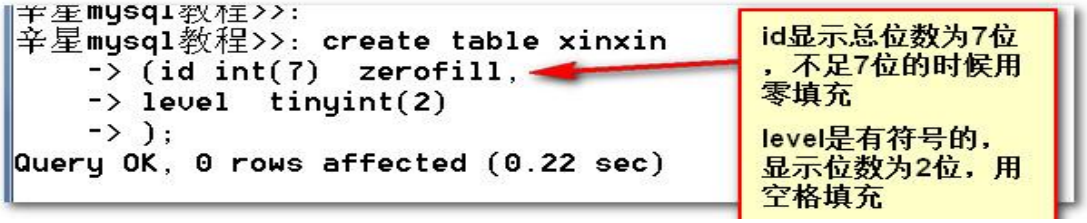
5.zerofill修饰符规定用0填充而不是默认的空格来填充。

6.必须注意的是，由于负数在前面补零无任何意义，因此，使用了 zerofill 就意味着它是无符号数据。

**********实战演示**********

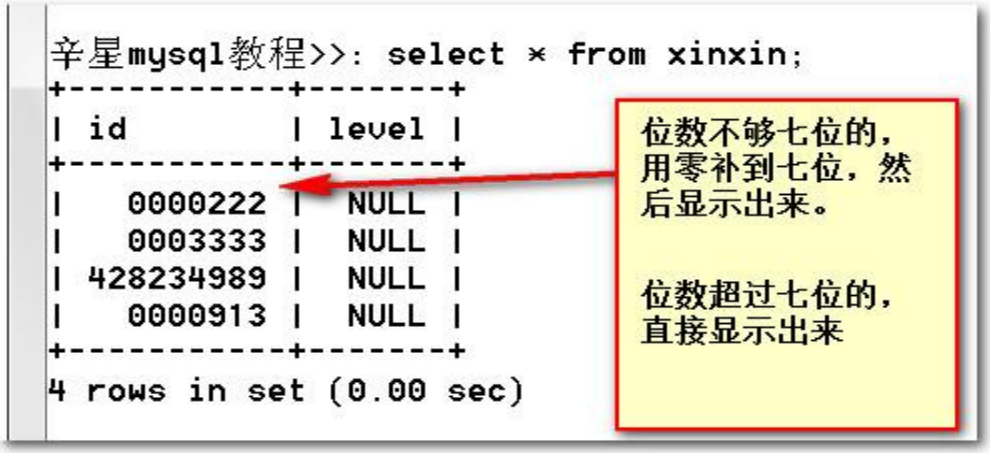
1.不知道经过上面的讲解，大家是否有点迷茫，我决定使用实战的方式来具体解释下。

2.我们创建一个表，格式如下：



3.我们先插入几个比较合理的数据，比如：

4.然后我们检索一下，看一下效果：



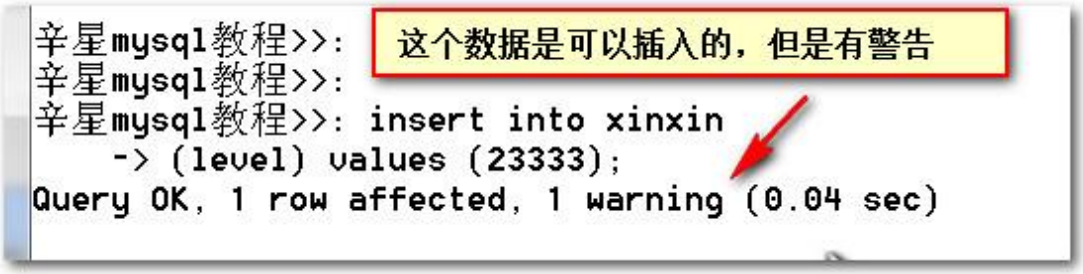
5.我们会发现，位数不足七位的，它会用零填充到七位，然后显示给我们，位数超过七位的，mysql 不会因为它超过宽度指示器来截断它。

6.有童鞋告诉我说，它存入的某些数据被截断了，其实跟宽度指示器没关系，是因为它超出了 mysql 相应数据类型的存储范围，比如用 tinyint 存储 232333233，肯定存不了那么大。

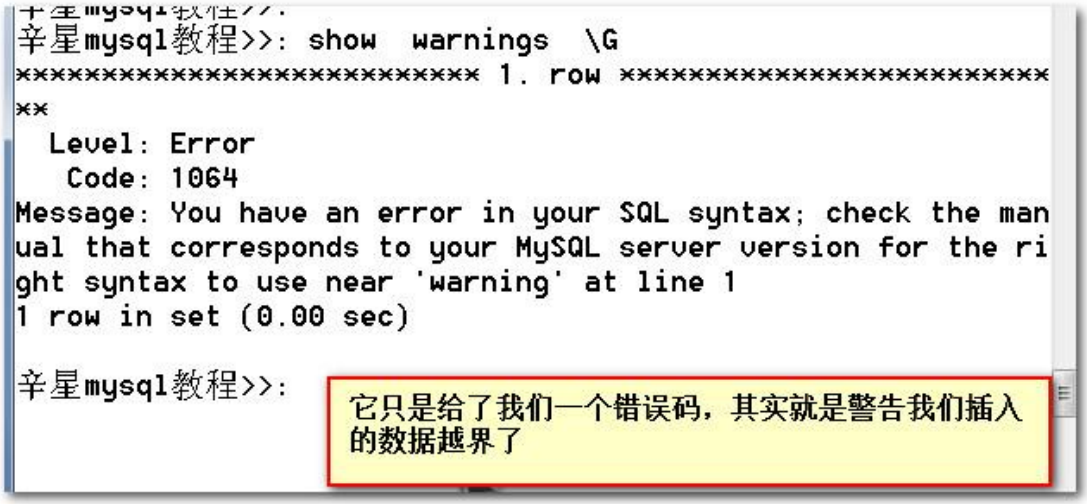
*********自动截断*****超出范围怎么办***************

1.如果我们存储的字段信息超出了 mysql 的相应的数据类型可以存储的范围，mysql 会根据允许范围最接近它的一段截断之后，再存进去。

2.比如说 tinyint 如果是有符号数的时候，存储大小为-128 到 127，比如我们非要存一个很大的数，比如 23333，看下面截图：

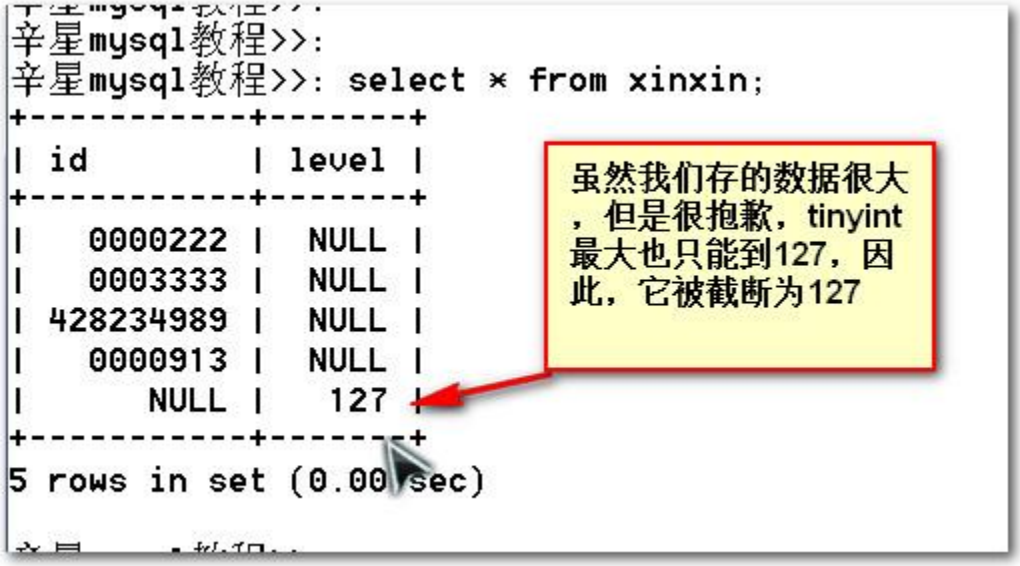


3.mysql 显示给我们插入数据成功，但是有一个警告，下面我们可以使用 show warning 来看一下这个警告：



4.不过呢，我们看了警告发现它只给一个代码，新手朋友可能看到这个代码无法立即反应过来它是干神马的。

5.我们从数据库检索一下数据：



**********不符规定**********类型转换**********

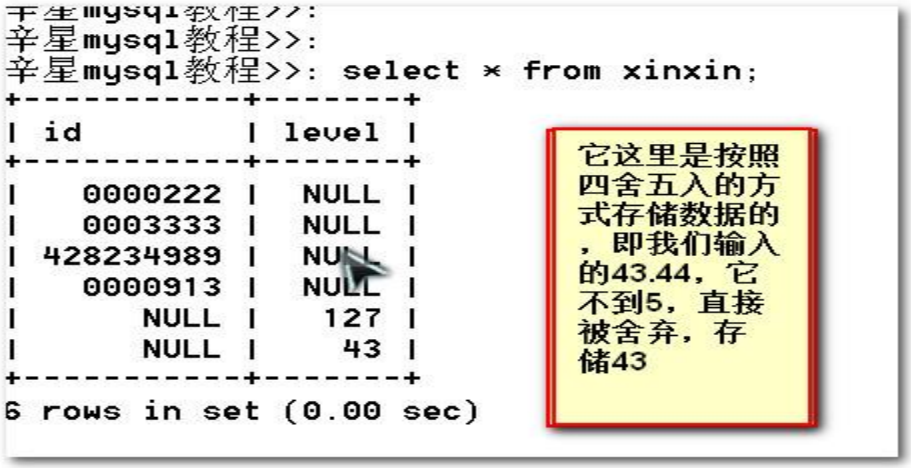
1.有些时候，总有些人好奇心很强，比如它想研究下用整型类型存储浮点数会怎么样，用整形类型存储字符串会怎么样。

2.这就涉及到 mysql 的类型转换规则了，不过这里并不是太常用，因为我们的开发过程中一般的类型都是匹配的。

3.下面我们存储一个小数进去：

4.可以看到，一切正常，我们存进去的数据是多少呢？

5.看下面截图：



6.那么我们存储一个字符串呢，看下面示例：

7.那么我们看看数据库里都存了些什么东西吧：

8.对于从字符串到整数类型的转换，我想各个编程语言都会讲到，这里我就不赘述了。

**********浮点类型***定点类型**********

1.很多语言都有浮点数的概念，其实浮点数是一个统称，咱们通常用浮点数来表示小数。

2.但是小数和浮点数没有必然联系，我们也可以用定点类型来表示小数，当然也可以用浮点类型。

3.所谓浮点数，有一套它的 IEEE 规范，比如 Java 采用的就是 IEEE754 标准。有些编程语言还分为单精度和双精度，比如 c，有些语言就只有浮点型，比如 php。

4.mysql 中的小数我们可以采用浮点型，也可以采用定点型，第一版中我不想给大家讲太深入的原理了，毕竟是入门教程，我后面会写一些关于内部的数据处理的教程，期待您的关注。

**********mysql**********小数**********

1.在 mysql 中，小数可以用定点类型来存储，此时数据类型为 decimal，也可以用浮点类型来存储，单精度浮点数为 float，双精度浮点数为 double。

2.下面是它的表格：

数据类型	字节数	所属类型
float	固定的 4 字节	浮点类型
double	固定的 8 字节	浮点类型
decimal	decimal(M,D) 由 M 和 D 决定	定点类型

3.我下面会分为浮点数和定点数两部分分别解释清楚，其实呢，我到后面数据库建模和数据库优化那部分还会介绍，希望大家在这里学好，否则后面的有关数据类型的介绍肯定会像听天书一样。

**********浮点类型**********

1.咱们的浮点类型也分为有符号的和无符号两种情况，即有unsigned修饰的无符号，否则有符号。

2.咱们的浮点类型也支持宽度指示器，只是它支持两个，比如float(4,2)这种，它一般用两个数来表示，前面那个表示显示的总位数，通常称之为“精度”，后面那个表示显示的小数点后的位置，通常称之为“标度”。

3.对于小数点后面的位数超出允许的范围之后，mysql会自动将它四舍五入为最接近它的值。

4.这些都和我们介绍整数的时候很像。

**********浮点类型的大小**********

1.float和double的大小都是固定的，原因很简单，float占据四个字节，double占据八个字节。

2.当然，double表示的范围比float要大，至于范围的计算，比较复杂一些，遵循IEEE规范。

3.float在表示无符号数的时候，最小可以到-3.4乘以10的38次方这个数量级，最大可以到3.4乘以10的38次方这个数量级，从贴近0的角度来看，负数可以是-1.17乘以10的-38次方这个数量级，整数可以到1.17乘以10的38次方这个数量级。

4.float在表示有符号的时候，最大可以表示到3.4乘以10的38次方这个数量级，从贴0角度来看，可以达到1.17乘以10的-38次方数量级。

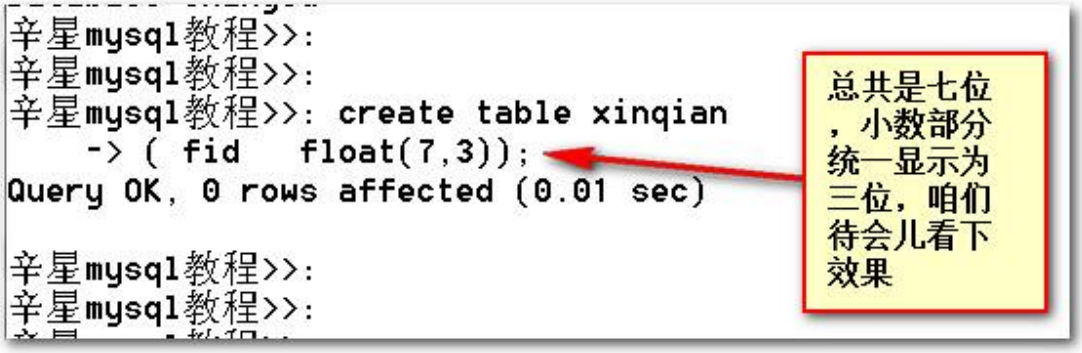
5.double的话，是10的308次方这个数量级，具体我不想展开了。

6.它是一个教程，不是一本手册，因此，它不列举那些毫无意义的只是用来展示的数据。

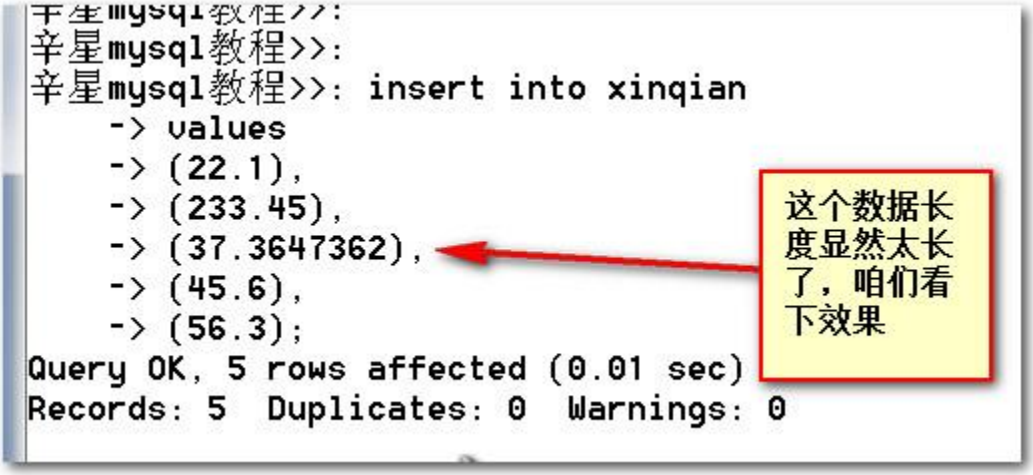
7.因为我列举了，读者也不住，我也记不很清。

**********浮点数操作实战**********

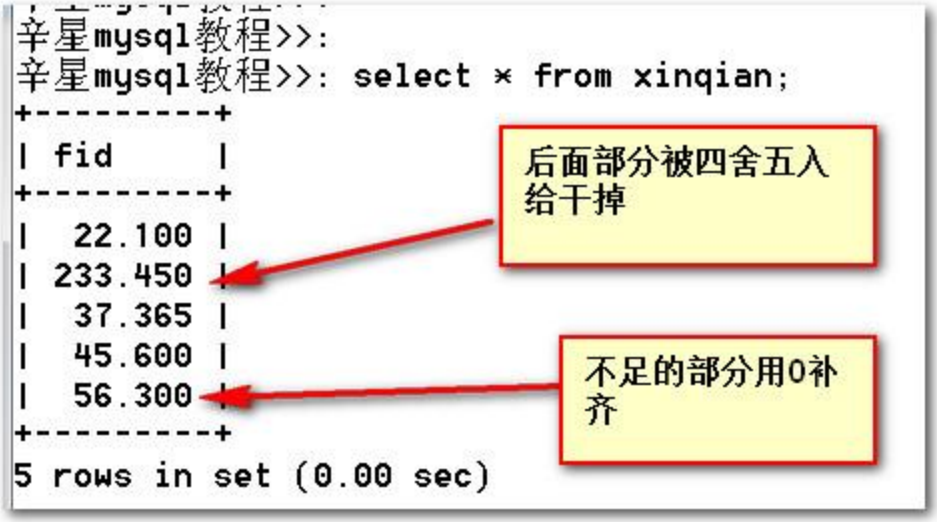
1.咱们首先创建一个表：



2.然后向里面随机添加一些数据：



3.咱们发现小数点后统一显示为三位，不足的用后缀0补齐，多余的干掉：



4.当然干掉的规则就是四舍五入，很简单。

**********定点数**********decimal**********

1.我们用 decimal 来定义一个定点数，使用语法就是 decimal(m,d)。

2.在 decimal(m,d)中，m 表示该数据的最大位数，称之为精度，d 表示小数点右侧的位数，称之为标度。

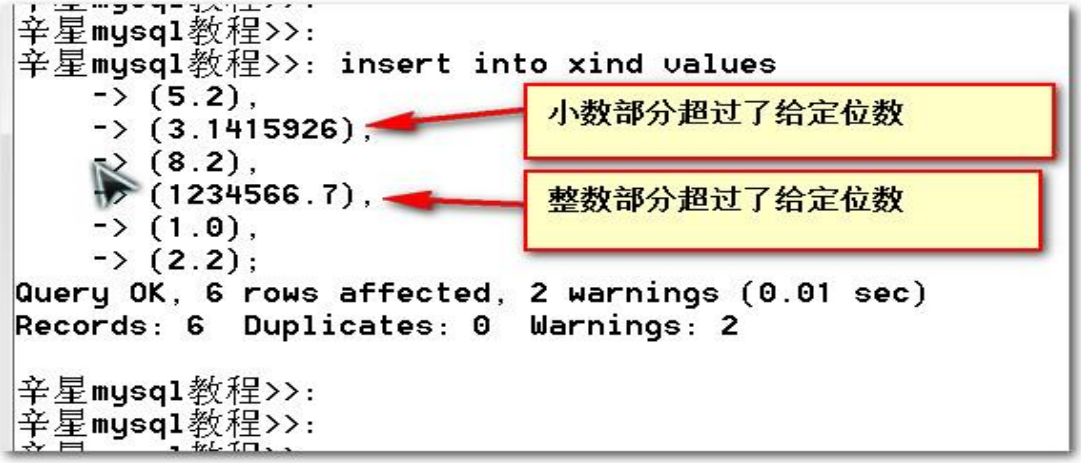
3.对于 decimal(5,2)表示该数据的最大位数是 5 位，其中小数点后面占据 2 位，比如存储 22.34 都是可以的。

4.对于 decimal(m,d)所占据的字节空间,如果 m>d,就为 m+2 个字节，否则为 d+2,不过我们绝大多数都是需要 m>d 的。

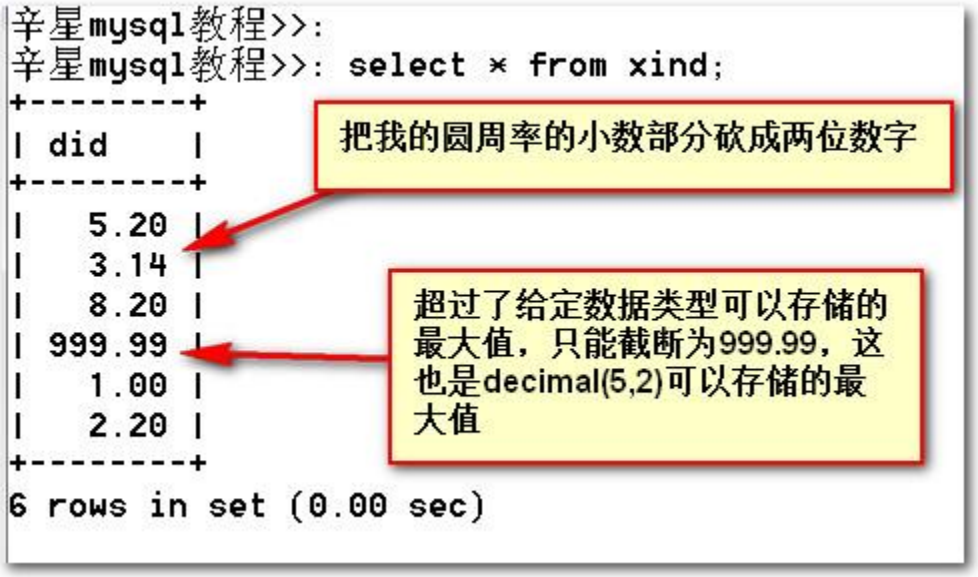
**********decimal 实战**********

1.首先我们创建一个表：

2.然后向里面随机插几条数据：



3.然后检索一下数据库，看看都存了神马东西：



4.当我们传入的最大值超过了 decimal(m,d)能存储的最大值的时候，它会自动存储最大值。

**********总结**********

1.这一节我们学习了整数类型的 5 种和小数类型的 3 种。

2.下面我做了一个表格，我们 来一起看一下，有兴趣吗？

类别	具体数据类型	占据空间	备注
整数类型	tinyint	1 字节	用于小型数据
	smallint	2 字节	
	mediumint	3 字节	
	int(integer)	4 字节	
	bigint	8 字节	
小数型	float	4 字节	
	double	8 字节	
	decimal	不确定	可定制性强

第三节：字符串类型

**********前言**********

1.mysql 提供了 10 个基本的字符串类型，可以存储的范围从一个简单的字符到巨大的文本块或者二进制字符串数据。

2.其实很多其他信息，比如整数、浮点数、时期时间等信息也可以转化为字符串类型然后存储。

**********类型表格**********

1.对于该字符串类型，共计十种，它们很相似，所占的存储空间大小上有区别，它们的存储空间大小在很大程度上决定了它们存储数据的能力。

2.我做了一个表格：

对应类型	大小(以字节为单位)	描述
char	0-255	定长字符串
varchar	0-65535	变长字符串
binary		字节字符串
varbinary		字节字符串
tinyblob	0-255	短二进制字符串
tinytext	0-255	短文本字符串
blob	0-65535	长二进制字符串
text	0-65535	长文本字符串
mediumblob	0-16777215	中等二进制字符串
mediumtext	0-16777215	中等文本字符串
longblob	0-4294967295	极大二进制字符串
longtext	0-4294967295	极大文本字符串

3.可以看出，它们的不同很大程度上是因为它们所占据的空间的大小，所占据的空间越大，能够存储的数据量也就越多。

4.这些意思表示什么，我们后面还会介绍。

**********char 类型**********

1.我们用 char 类型表示定长字符串，它的使用规则就是在圆括号内使用一个大小修饰符来定义，而且该大小修饰符决定了它所占据的空间。

2.该大小修饰符为 0-255，当我们向里面填充数据的时候，比指定长度大的值将会被截断，而比指定长度小的值默认用空格填充。

3.比如我们指定 name 字段为 char 类型，且占据 20 个字符，则使用如下定义：【name char(20)】即可。

**********varchar 类型**********

1.varchar 类型可以理解为 char 类型的变体，它也可以后跟一个小括号，小括号中填写内容为它至多占据的空间。

2.该修饰符大小就比较大一些了，它是从 0 到 65535.

3.比如我们定义 name 字段为 varchar 类型，比如用【name varchar(20)】来存储 20 个字符。

**********char*********varchar**********

1.char 和 varchar 存储的文本种类都是字符串，同样的信息都可以被这两种类型存储。

2.对于 char 和 varchar 中有一个整数指示其大小，char 和 varchar 的处理方式是不同的。

3.char 会把这个大小视为值的大小，长度不足的情况下用空格补齐。

4.varchar 类型会把它视为最大值并且只是用存储字符串实际需要的长度(增加一个到两个额外的字节来存储字符串本身的长度)来存储值。

5.存储短于指示器的 varchar 类型不会被空格填充，但是长于指示器的值仍然会被截短。

**********图解其存储模型**********

1.加入我们存储三个字符串，分别是“xin”，“qian”，“xiaonan”，那么我们用下图简单的描述其存储模型。

2.这里我们使用 char(8)来存储，对于英文字母来说，一个字符用一个字节来存储。

3.对于 char 的存储方式简图(占据字节 24)：

xin	qian	xiaonan

4.这是以字节为单位来刻画的：

x	i	n						q	i	a	n				x	i	a	o	n	a	n	

5.对于 varchar 的存储方式(占据字节 16)：

3	xin	4	qian	7	xiaonan

6.这是以字节为单位刻画的：

3	x	i	n	4	q	i	a	n	7	x	i	a	o	n	a	n

7.我也没用什么专业的作图工具，可能大家看起来有点丑陋，不过大致就是这个意思。

**********再看 char 和 varchar**********

1.我们有了上面的一些理解之后，再次比较下 char 和 varchar。

2.char(m)类型的数据列里，每个值都占用 m 个字符，如果某个长度小于 m，那么 mysql 在内部处理的时候会给它加上空格来补齐为 m 个字符，但是在检索的时候再把这些空格去掉。

3.在 varchar(m)类型的数据列里，每个值只占用刚好够用的字节再加上用来记录其长度的字节，如果该字字段的长度小于 255，它只需要加一个字节即可，如果该字段的长度大于 255，就需要加两个字节，因此它占用的实际空间为 m+1 或者 m+2。

4.在数据表中，如果每一个字段的长度是固定的，那么每一行数据的长度也会是固定的。

5.如果数据表中有一个字段的长度是可变的，那么该行数据的长度就是可变的。

6.如果某个数据表里的数据航的长度是可变的，那么为了节约存储空间，mysql 会把这个表中的长度大于 4 个字符的固定长度的数据列转化为相应的可变长度类型。

**********经验**********

1.一般来说，使用 char 类型会加速检索的速度，它很像我们 c 语言中的数组，因此访问速度超级快，由于在内存中整齐的排列，因此访问 速度超快。

2.使用 varchar 类型会节省存储的空间，提高空间利用率。

3.因为 varchar 类型可以根据实际内容动态改变存储值的长度，所以在不能确定字段需要多少字符时使用 varchar 类型可以大大地节约磁盘空间、提高存储效率。

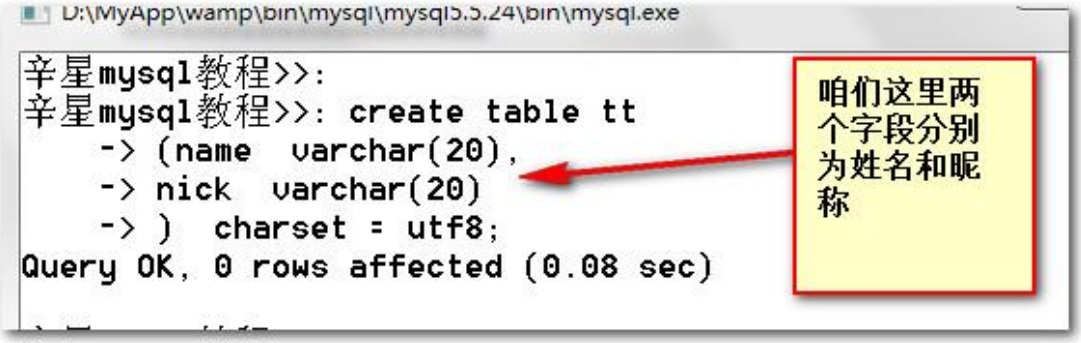
**********字符集**********

1.前面也零零散散的说了一些字符集的东西，这里还是再说一下把，我们在建完表之后加一个 charset = utf8 或者 charset = gbk 都可以显示中文。

2.这里我们使用 utf8 编码。

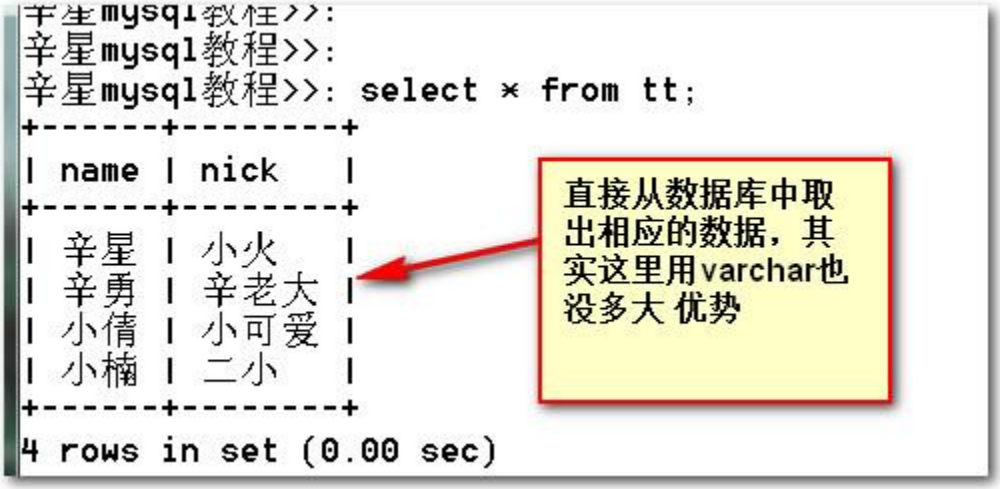
**********接触 varchar 实战**********

1.我们用 varchar 来建立一个 tt 表：

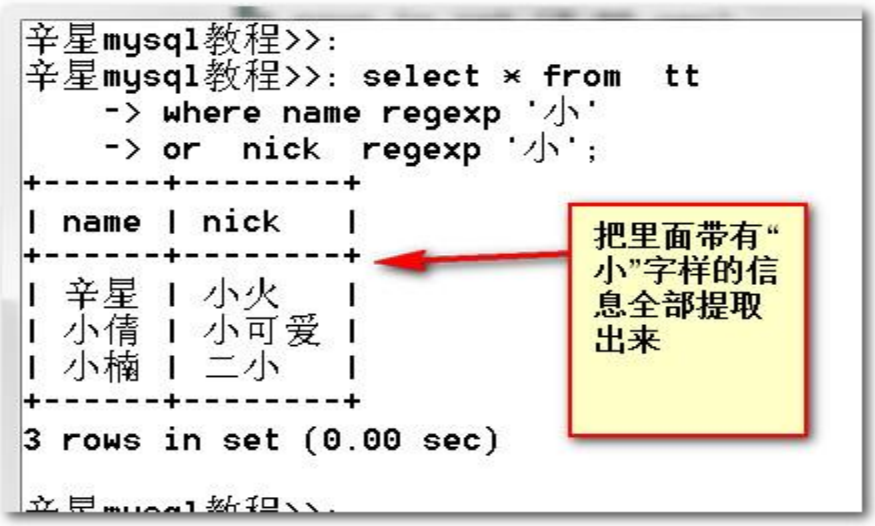


2.然后我们向里面插入几条数据：

3.然后我们从数据库中检索一下相应的信息：



4.搜索里面和“小”有关的信息：



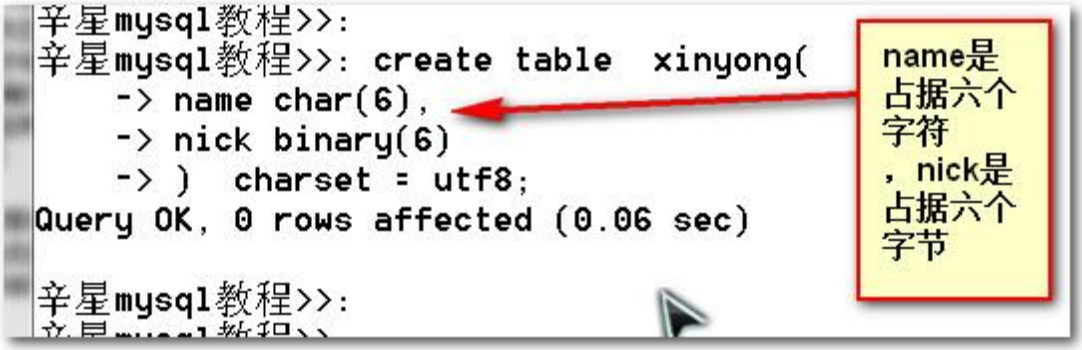
**********binary**********varbinary**********

1.包括一些培训机构出来的童鞋都没听说过这两个类型，确实，它们在 mysql 手册的描述信息也不多，而且实际应用中也并没有那么广泛。

2.binary、varbinary、char、varchar 都可以存储字符串文本信息，差别在于内部的处理格式，给外面展现的好像是一样的。

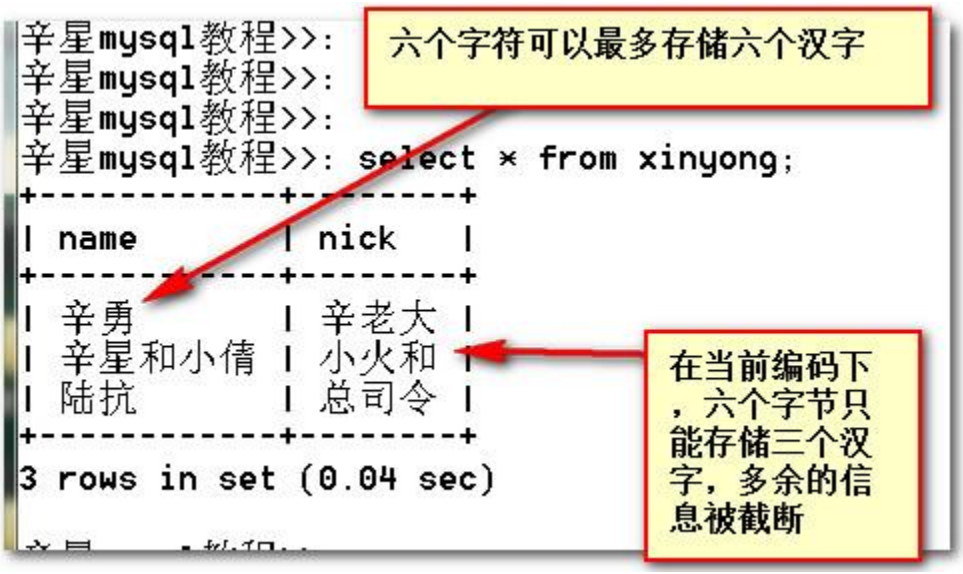
2.binary 是按照字节来计数的，举个栗子，我们 name bianary(6)表示 name 这个字段占据六个字节，而 name char(6)表示占据六个字符。

3.可能上述描述并不是很直观，我们实战演练，首先来一个表：



4.然后我们向里面插几条信息：

5.然后我们显示一下：



6.我们发现 name 是 6 个字符信息，而 nick 是 6 个字节信息，那么 name 字段可以最多存储六个汉字，而 nick 则在该编码之下只能存储三个汉字，多余的部分会被截掉。

**********text**********blob**********

1 对于一些较大的文本，我们通常考虑其他的数据类型。

2.此时我们需要借助于 text 家族和 blob 家族，这两大家族都根据我们需要存储信息的大小，有不同的子类型。

3.text 和 blob 可以用于存储博客、邮件等文本信息。

4.其实他们两个就可以当做字符串来用，只是它们表示的字符串比较长，比如一篇博文，里面可能有很多内容。

**********text**********blob**********

1.blob 是 binary large object 的简写，即“二进制大对象”，而 text 本身就是“文本内容“的意思。

2.一般来说，blob 类型区分大小写，text 不区分大小写。

3.它们通常用于存储较大的信息。

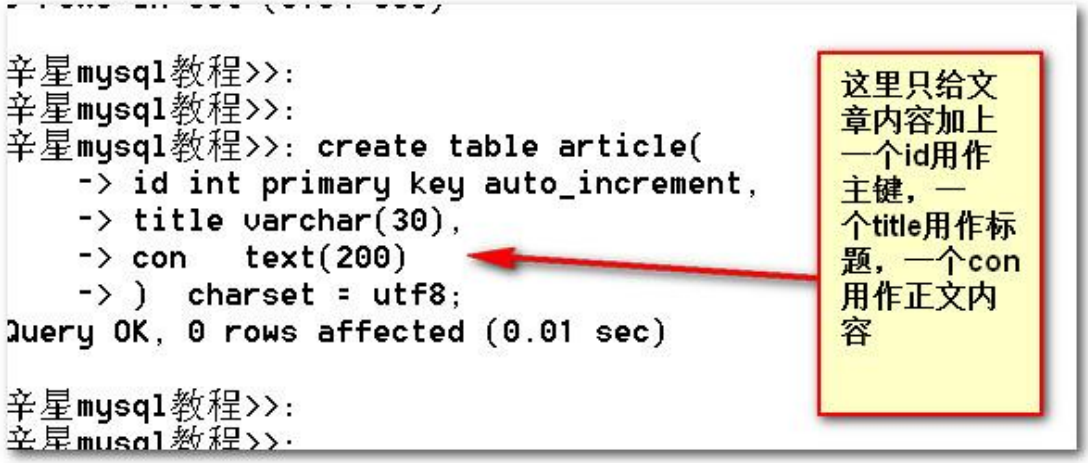
**********经验**********

1.其实这里面又属 text 比较常用一些。

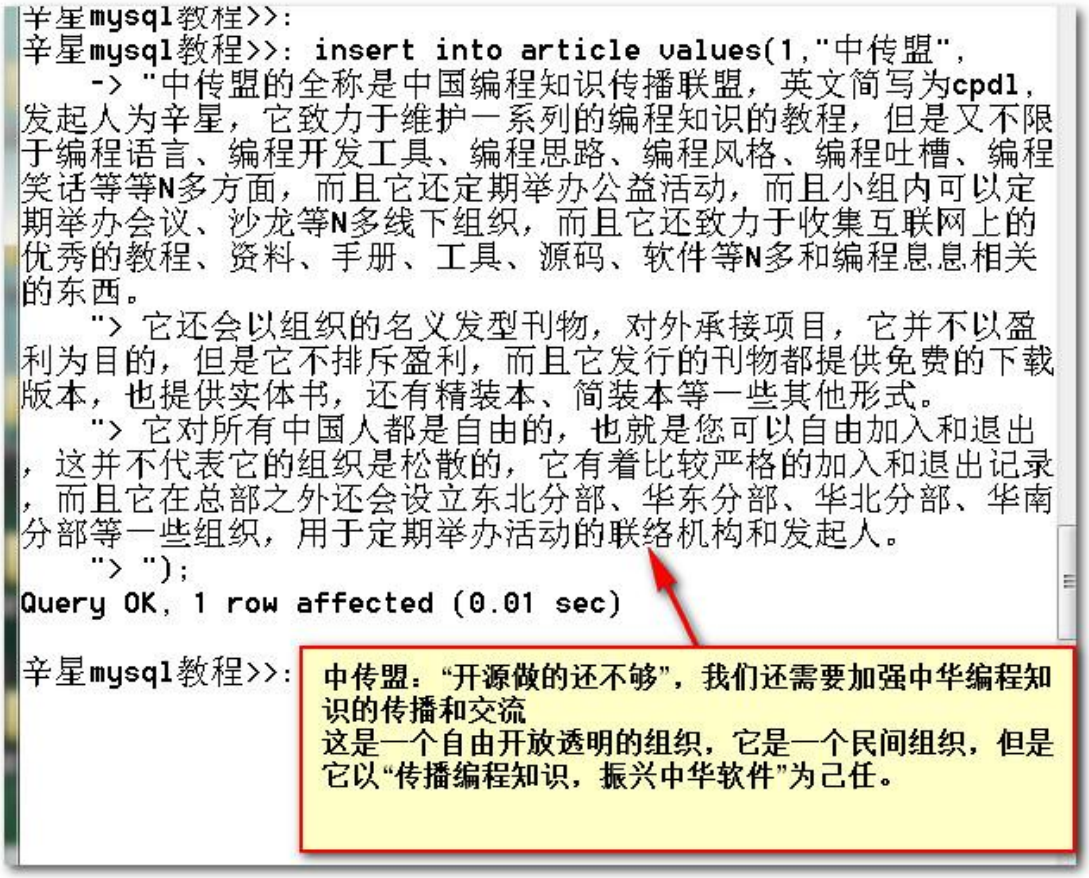
2.因为其名字也容易记忆，而且它也是为存储较大的文本而量身定做的。

**********text 实战**********

1.我们建立一个最简单的文章的列表：



2.然后我们插入一些信息：

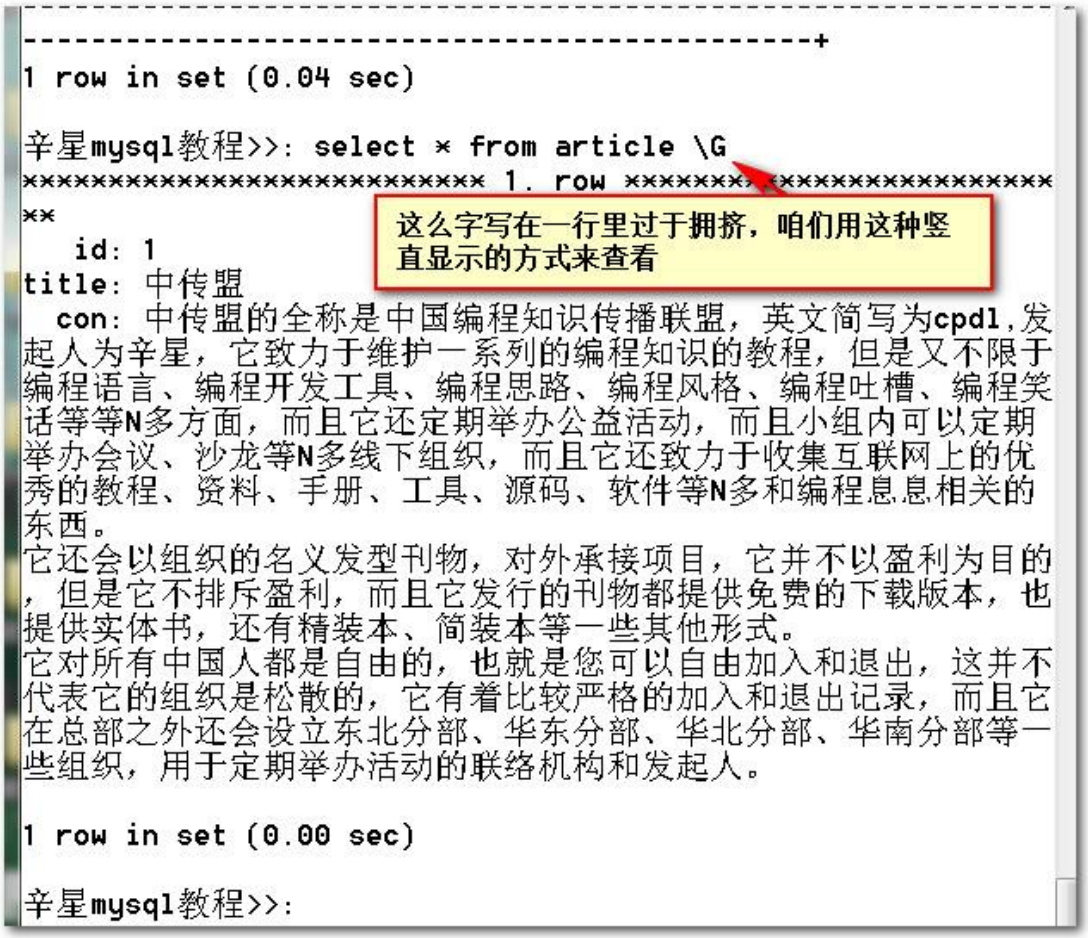


3.我们随便插入一点信息，当然我这里是一个字一个字地敲进去的。

4.打个小广告把，如果说开源运动极大的促进了软件产业的发展，但是它是以企业为单位进行促进的，比如 mysql 是开源的，但是只有使用 mysql 的企业对其源码感兴趣，初学者是没有能力看这些开源的源代码，企业内不使用 mysql 的高手们则往往会被自己的事忙的没有经历去看，因此，软件的开源会从一方面促进软件工业的进步，但是不能彻底的推动软件工业。

5.“编程知识传播联盟”致力于传播编程知识，它诞生于人口基数最大的国家→中国，它包括了面向零基础的低端培训，也会包括面向有三五年工作经验的中端培训，还会包括面向科研机构的高端培训,这些培训资料我们都会及时整理出来并且免费发布的。

6.废话到此为止，咱们看看 article 中都有虾米内容：



**********小结**********

1.这一节咱们介绍了常用的字符串类型，内容还是有点杂的。

2.最常用的就是 char 和 varchar 了，其次 text 也比较常用。

3.当然了，学无止境，前进的道路上，你我共同努力。

第四节:日期时间类型

**********日期时间类型**********

1.在处理日期和时间的值时，mysql 带有 5 个不同的数据类型可供选择。

2.它们可以被分为简单的日期、时间类型、混合日期事件类型。

3.根据要求的精度，mysql 带有内置的功能可以把多样化的输入格式编程一个标准格式，但是我建议使用比较标准的格式，这样最稳。

**********图解 mysql 的时间日期类型**********

1.我们依旧用表格的形式来搞定，以后版本会考虑其他形式。

2.看下面表格：

类型	占据字节	显示格式(非存储格式)	用途
year	1	yyyy	存储年份
date	3	yyyy-mm-dd	存储日期
time	3	hh:mm:ss	时间点(段)
datetime	8	yyyy-mm-dd hh:mm:ss	日期和时间
timestamp	4	yyyy-mm-dd hh:mm:ss	日期和时间

3.上面表格中的 yyyy 表示四位数的年份，比如 1992,1987 等等。mm 表示两位数的月份，比如 08,11 等等，d 表示天，h 表示小时，m 表示分钟，s 表示秒钟，这些我就不赘述了。

**********日期类型**********

1.mysql 用 date 和 year 两种类型来记录日期。

2.year 类型用于记录年份，通常需要四位数，但是我们也可以输入两位数，它会自动根据自己的规则来转化为四位的年份表示方式。

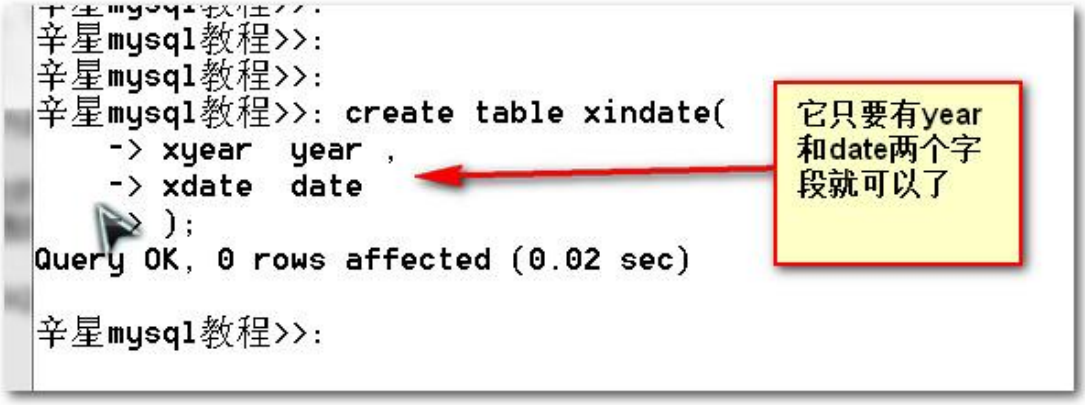
3.转化格式为：70-99 会自动增加一个前缀“19”，而 00-69 会自动增加一个前缀“20”。

4.咱们用 date 类型来存储具体的日期，来精确到某一天。

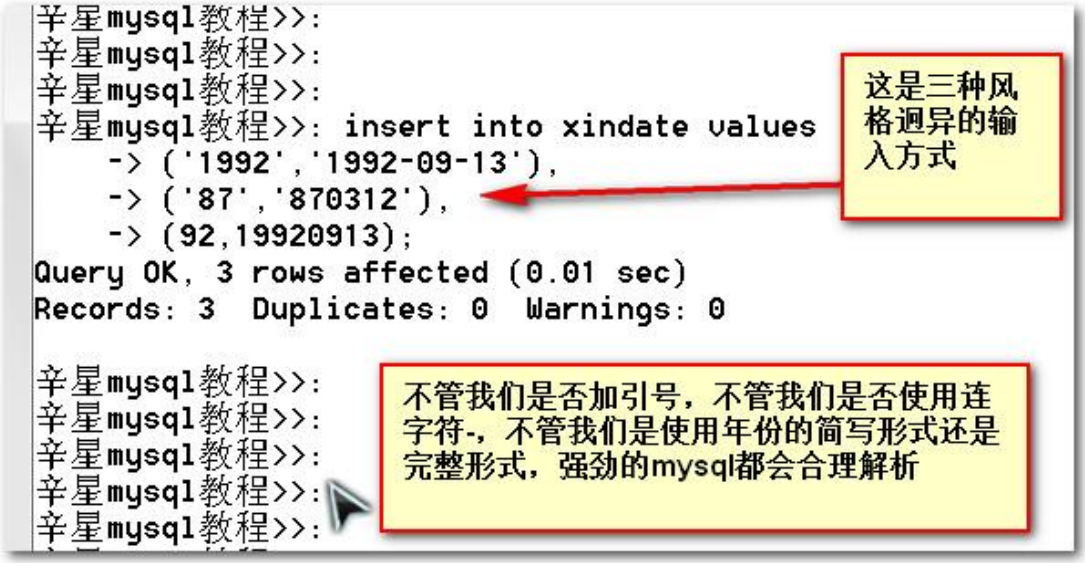
5.大家需要记住 date 的解析格式是“年月日”这种方式来解析我们的输入信息。

**********日期实战**********

1.首先我们建一个表格：

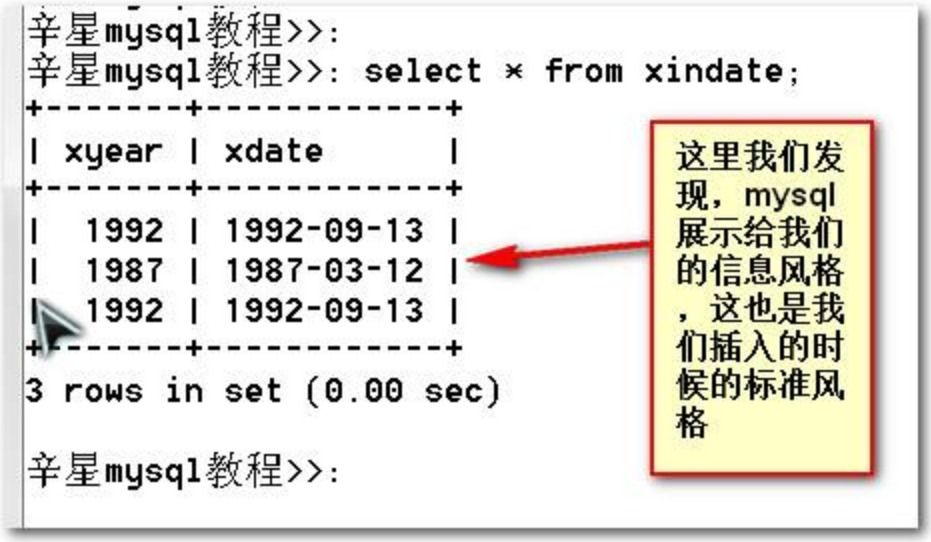


2.然后我们向里面插入数据：



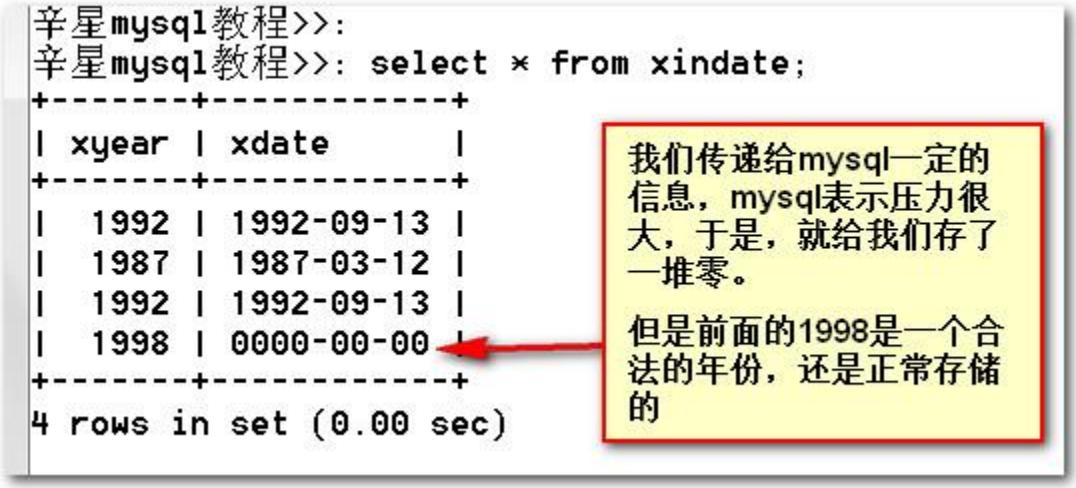
3.不管我们输入的内容的风格多么独特，mysql 还是会任劳任怨的给我们解析完毕，当然，我这里只是列举几种大家普遍比较喜欢用的风格，其中说一下，第一种书写格式是 mysql 推荐的。

4.我们看看 mysql 都存了神马东西：



5.虽然很自由，但是不代表我们可以胡乱填写内容，比如我们输入一个不合法的日期：

6.根据上面的1 row affected 可以看出确实是插入了一行信息，但是插入的这一行信息是什么内容呢，我们看下面截图：



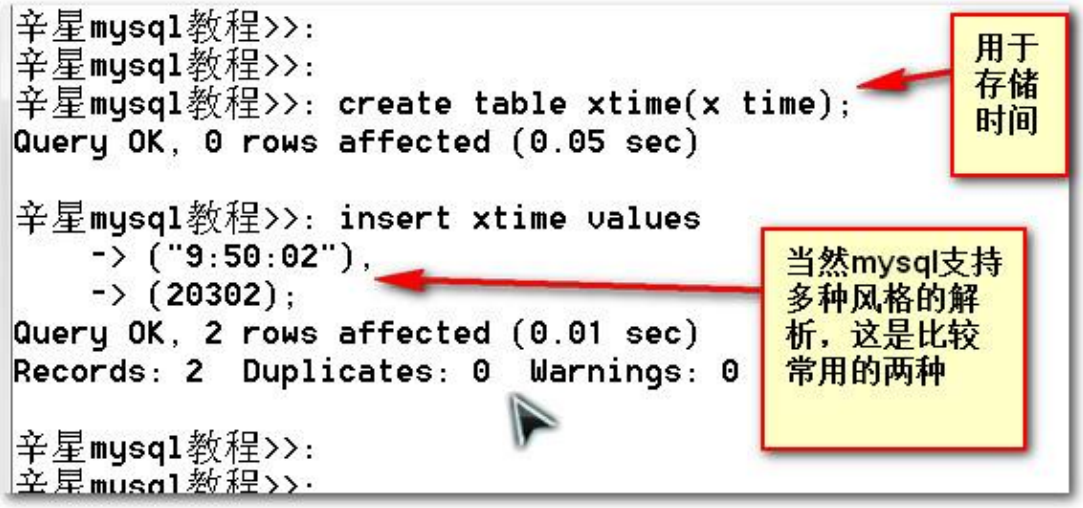
**********time**********时间**********

1.time 类型就是时间类型，它可以表示一个时间点，比如当前时间是“9:50:00”。

2.它也可以表示一个时间段，比如从我今天开始写这个教程到现在共计经过了两个小时零三分钟，我们可以存储为"2:03:00"。

3.就像 date 使用“-”作为分隔符一样，我们在 time 中习惯使用":"来作为分隔符。

4.我们新建一个表格并插入两条数据：



5.然后我们从里面检索一下信息：

**********datetime****timestamp**********

1.这里的 datetime 即“date”和“time”的终极版，它即可以存储日期又可以存储时间。

2.timestamp 翻译为汉语即“时间戳”，它是当前时间到 Unix 元年(1970 年 1 月 1 日 0 时 0 分 0 秒)的秒数，因此它是一个比较大的整数，而且我们直接拿到这个数据，并不能确定它的时间是多少。

3.由于时间的计算比较困难，比如现在时间是 2014 年 8 月 18 日 9:55:00，我的出生时间是 1992 年 9 月 13 日 8:22:12,我想问一下我活了多少秒钟，这个计算就异常费力了，我们需要考虑是否有闰年，总之需要花费一段时间编写一个函数去运算才行。

4.但是使用时间戳，第 3 条提出的问题就都不是问题，因为对于时间戳来说，时间的运算何止是简单，简直就是简单，由于两个数都是整数，直接做个减法就 ok 了。

5. 如果我们不对 timestamp 赋值，它会自动用系统当前的日期和时间来赋值。

**********表面现象 **********本质区别**********

1.就像 char 和 varchar 一样，如果我们都是向里面输入字符串并且检索字符串，那么会感觉它们木有区别，因为同样的操作对两者完全是一样的效果。

2.但是 char 和 varchar 的内部存储机制有着本质的区别，一个是定长存储，一个是变长存储。

3.timestamp 和 datetime 也是这样，它们接受同样的参数，返回的信息的格式也都是 YYYY-MM-DD HH:MM:SS 的 19 个字符的宽度的格式，好像没有任何的区别和变化。

4.它们的区别在于它们的内部存储机制，下面我们再讲。

**********存储的眼光看 datetime 与 timestamp**********

1.datetime 以“yyyy-mm-dd hh:mm:ss”的格式检索和显示，支持的存储范围从“1000-01-01 00:00:00”到”9999-12-31 23:59:59”。

2.但是 timestamp 作为一个时间戳，用的是 Unix 时间戳，因此它的值不能早于 1970 年。

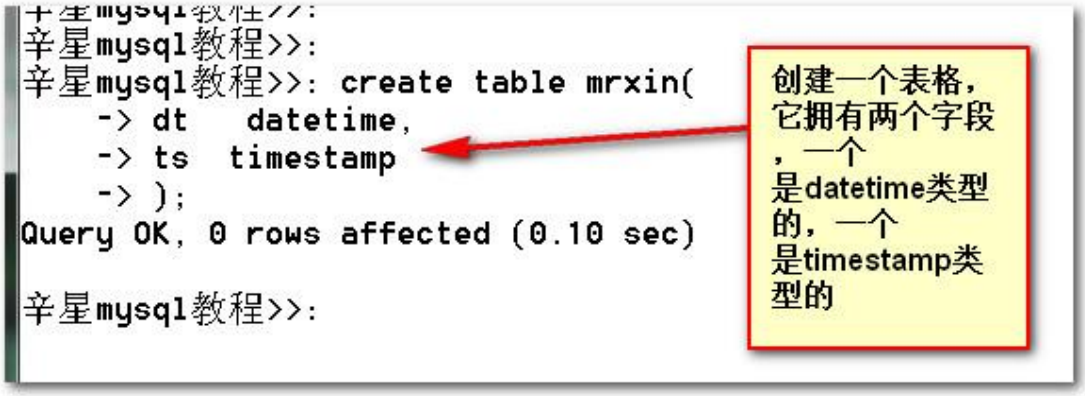
3.timestamp 使用四个字节存储，而且以 utc 格式存储，它会检索当前时区。

4.datetime 则会使用八个字节存储，它不会检索当前时区。

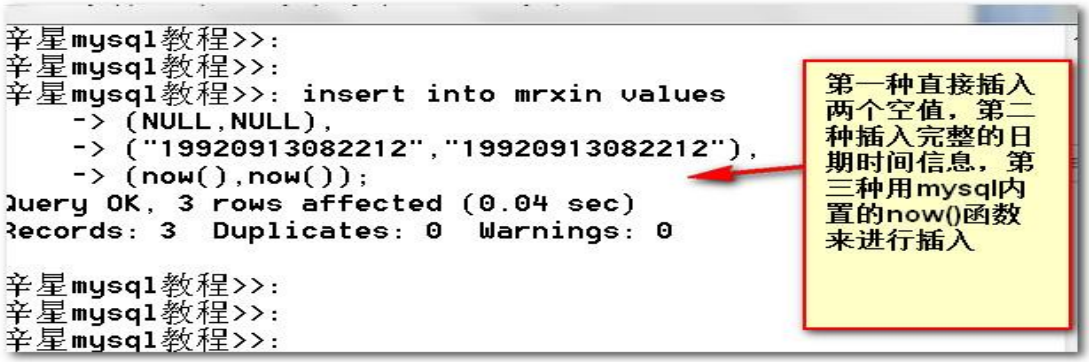
5.如果我们向 timestamp 插入空值，那么会插入当前时间，如果我们向 datetime 插入空值，那么会插入一个空值。

**********实战演练**********

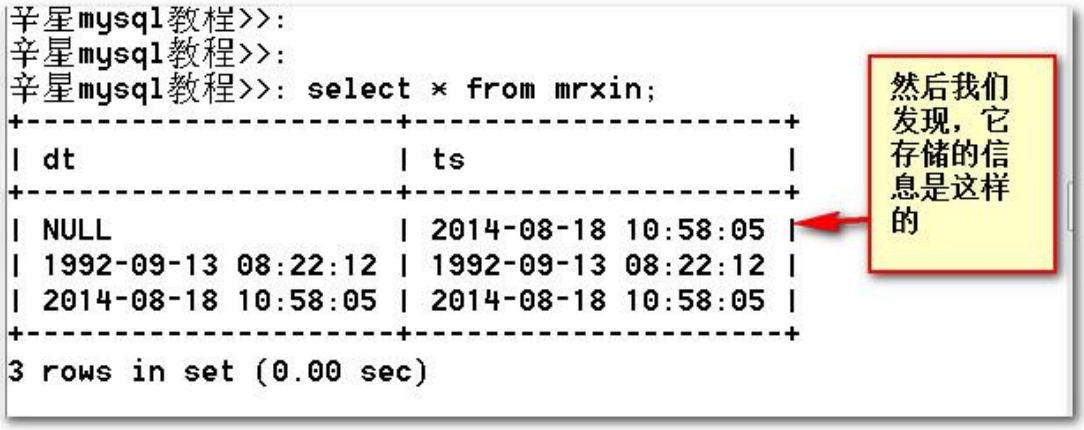
1.首先创建一个表格：



2.然后插入几组值：



3.然后我们看看都存了些什么东西：

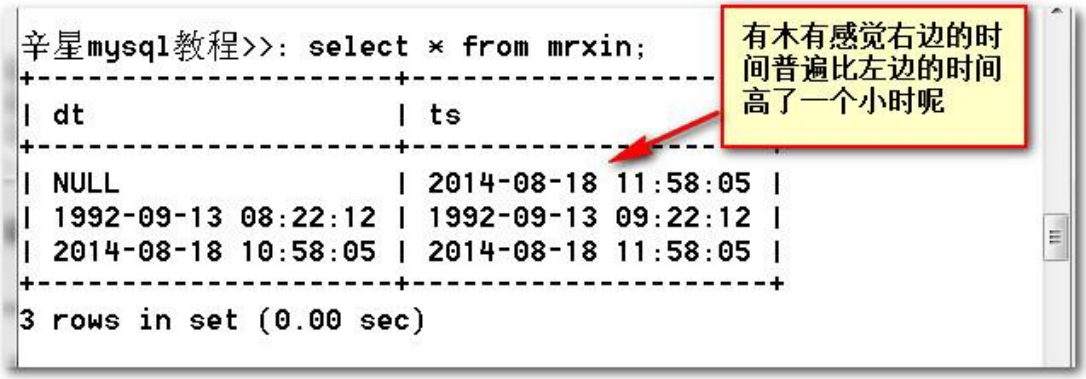


4.我们发现给 datetime 存一个 NULL，就是保存一个 NULL，给 timestamp 存一个 NULL，就是存当前时间。而且两者都支持多种风格的数据解析的，我们下面大家有兴趣再解释。

5.大家需要注意下我这里的 timestamp 和 datetime 存储的都是一样的时间，再确认下，(⊙o⊙)…

6.大家需要记住 timestamp 是和时区有关系的，这里我更改一下时区，比如我改成东京的所在的时区→东九区：

7.现在我们再来看一下相关数据：



**********优雅的风格 ********程序员的审美********

1.前面说过 mysql 支持 n 种风格的解析，不过前面都是主要讲应用，很少讲一些优美有趣的部分。

2.下面我们新建一个表把，它只有一列，咱们就用 datetime：

3.咱们以自己喜欢的风格插入几条数据，下面是我比较喜欢的风格：

4.咱们发现上述风格就比较舒服一些了，咱们可以把咱们要输入的信息写成一坨，也可以写成两坨，而且还可以以更加优雅的风格去书写。

5.程序员因为整天奋斗在代码堆里，我们必须发现其中的乐趣才行，否则会越来越累，越来越烦，最后干脆就干不下去了。

**********一点扯淡**********

1.说到了优雅，就不能不提一门语言：python。

2.我用过 c、c++、Java、php、shell、Pascal、Javascript 等一些语言，发现语法最优雅的还是 python，大家不妨去研究下，用 python 写代码感觉是一件比较舒服的事。

第五节：复合类型

**********复合类型**********

1.mysql 还支持两种复合数据类型：enum 和 set，它们扩展了 sql 规范。

2.虽然这两种类型在技术上都是字符串类型去实现，但是可以被视为不同的数据类型。

3.enum 类型只允许从一个集合中取得一个值，而 set 类型允许从一个集合中取得任意多个值。

**********enum**********

1.有过 c 语言或者 Java 语言基础的应该都知道有一个类型叫做“枚举类型”，但是我们这里的 enum 和枚举类型并不完全一致，但也有不少相似点。

2.enum 类型字段可以从集合中取得一个值或者使用 null 值，除此之外的任何输入都会使得 mysql 在这个字段中插入一个空字符串。

3.如果插入值的大小写与集合中值的大小写不匹配，mysql 会自动使用插入值的大小写转换为集合中大小写一致的值。

**********语法格式**********

1.比如我们指定一个字段为 enum 类型，并且取值仅限于 m 和 f，那么字段应该是【gender enum('m','f') 】即可。

2.插入的时候直接使用'm'或者'f'即可。

**********内部机制**********

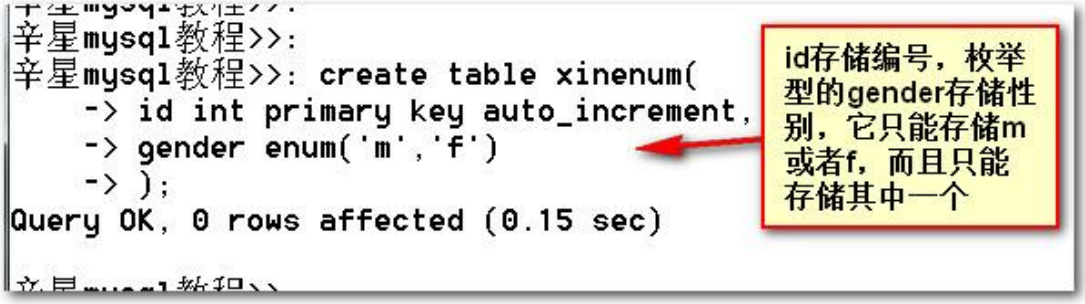
1.这里只是简单的说下它的内部机制，并不是它的全部实现。

2.enum 类型在系统内部可以存储为数字，并且从 1 开始用数字作为索引。

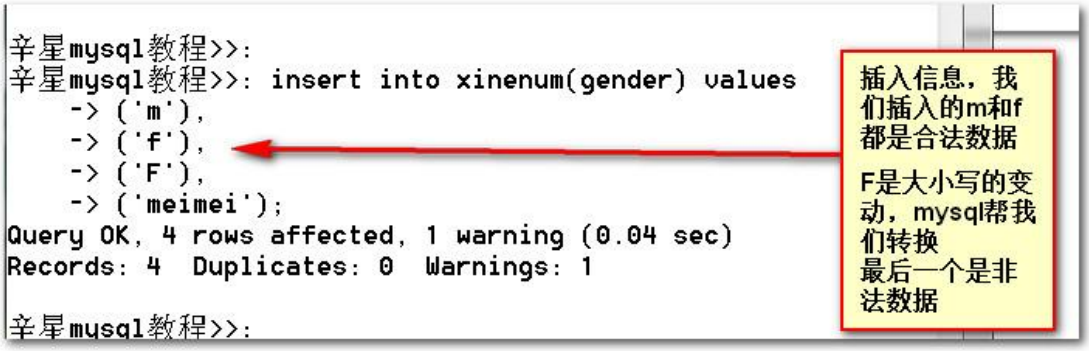
3.一个 enum 类型最多可以包含 65535 个元素，其中一个元素被 mysql 保留，它用于存储错误信息，这个错误值用空字符串来表示，对应的索引为 0.

**********实战环节**********

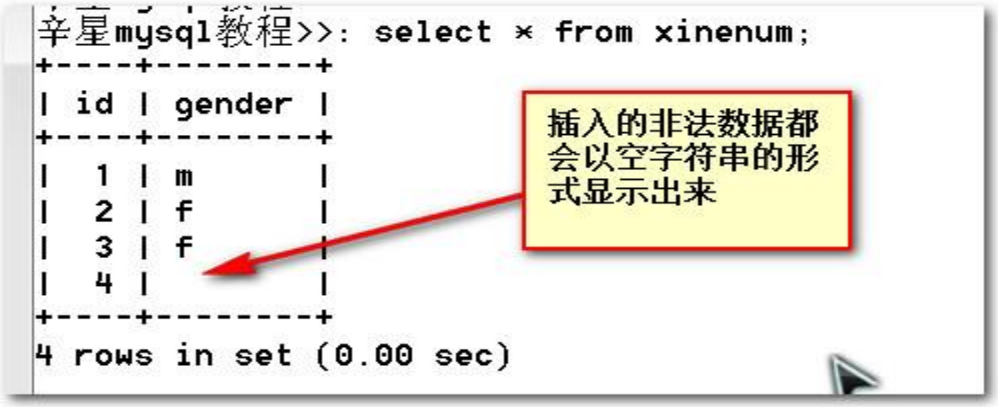
1.首先我们建一个表，我们用 id 表示成员编号，用 gender 表示其性别：



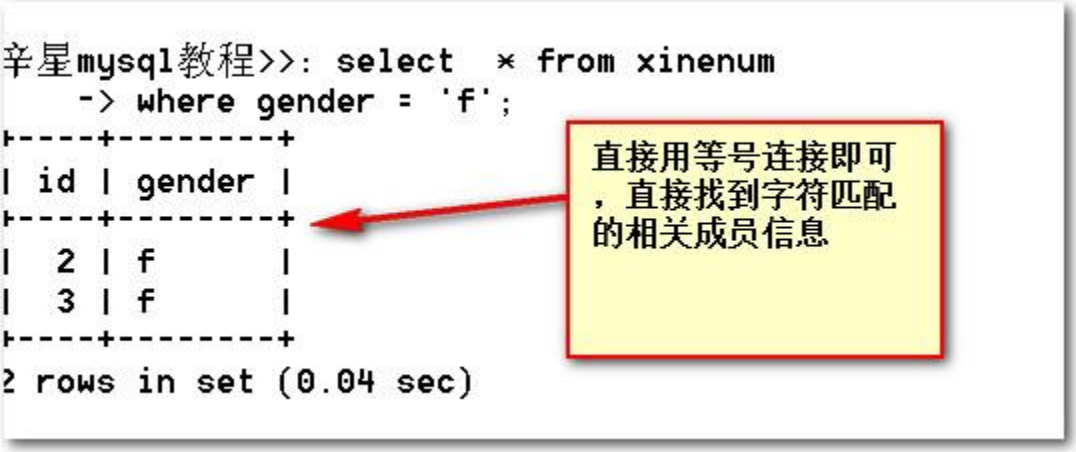
2.下面我们向里面输入一些数据：



3.然后我们看一下效果：



4.然后我们从里面检索一下信息，比如我们这里检索性别为 f 的所有成员的信息：



**********set 类型**********

1.set 即“集合”，它可以从预定义的集合中取得任意数量的值，而不是只能插入一个值。

2.并且与 enum 类型相同的是，如果我们试图在 set 类型字段中插入一个没有预定义的值，都会让 mysql 插入一个空字符串。

3.如果我们插入一个既有合法数据又有非法数据的元素的记录，那么 mysql 会保留合法的数据，除去非法的数据。

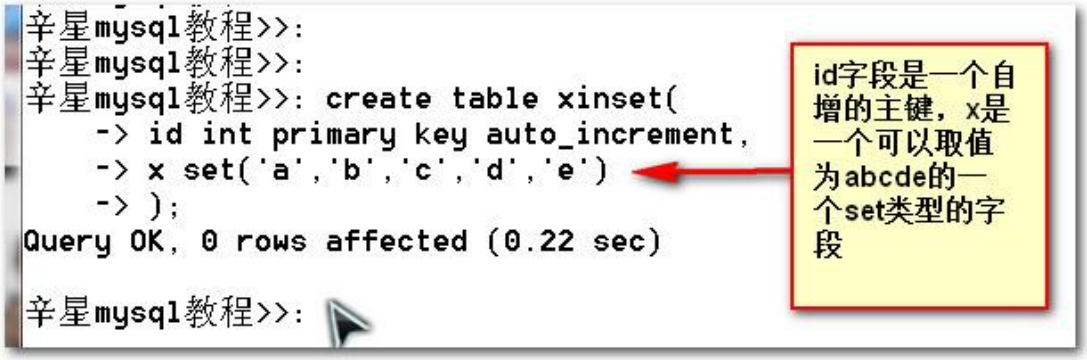
**********语法格式**********

1.set 语法格式和 enum 是很相似的：【name set('a','b')】这样子。

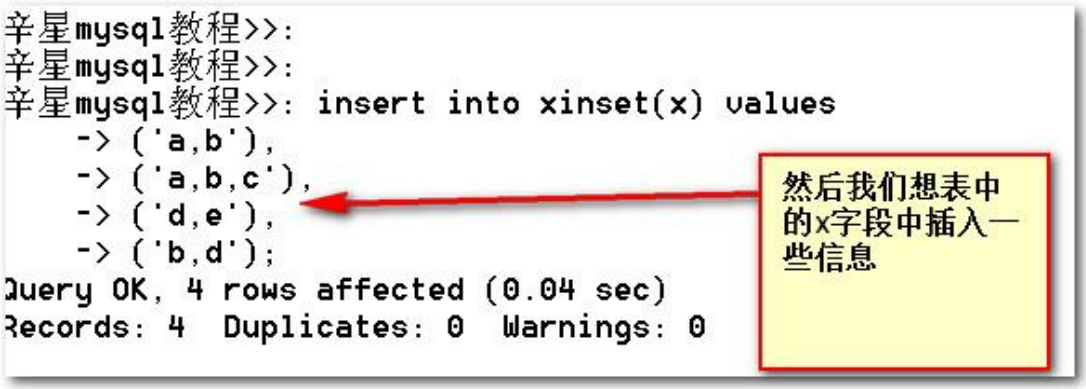
2.至于它的插入和查询大家还是看我的实战演示把，如果插入和查询都明白的话，对于删除和修改就是小 kiss 了。

**********set 实战**********

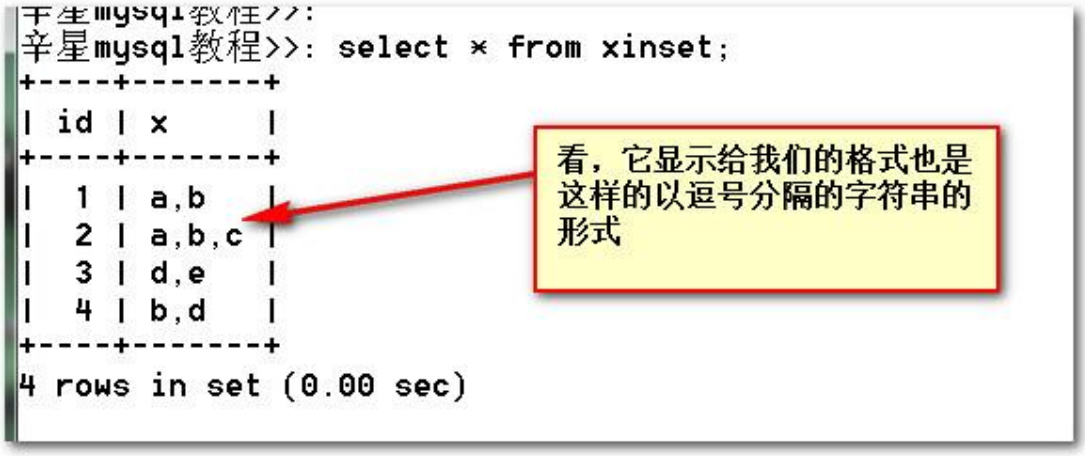
1.我们首先创建一个表，然后在创建该表的时候指定该 set 可以取哪些值：



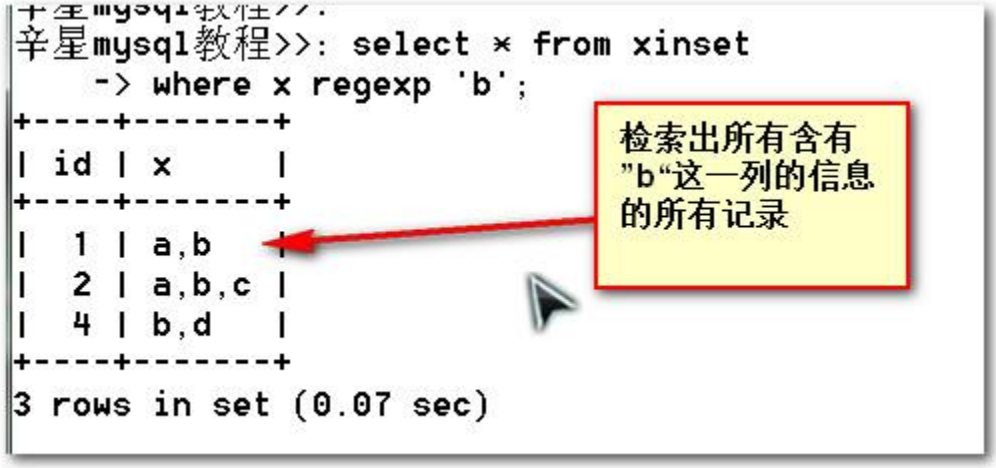
2.然后我们向里面插入一些信息：



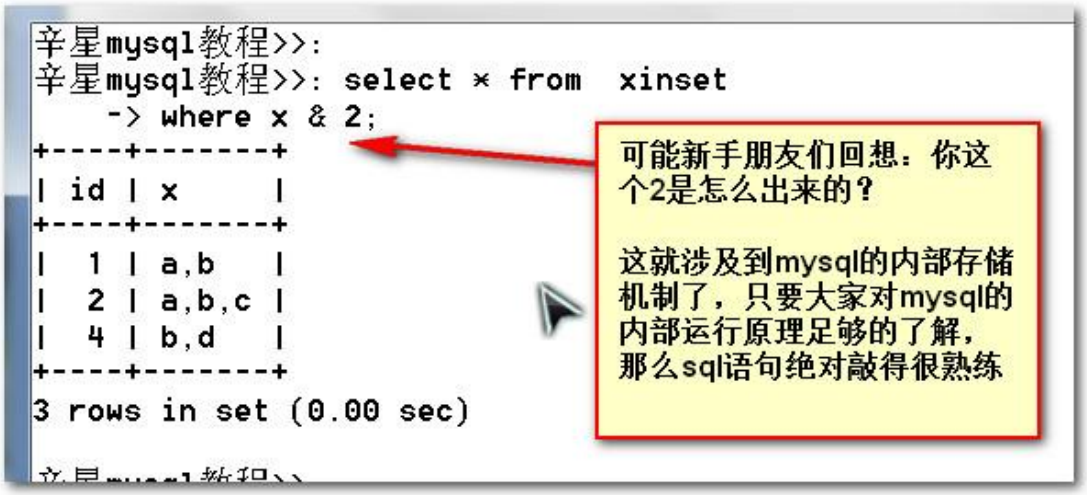
3.然后我们检索一下我们插入的信息：



4.那么比如我们要是想要检索特定的信息呢？比如我们想检索 x 列中含有 b 这一项的所有信息，可以用 like 操作符或者 regexp 操作符，这里我使用了 regexp 操作符：



5.当然在介绍一种方式，这种方式我下面再给大家解释一下原因，不过特别方便：



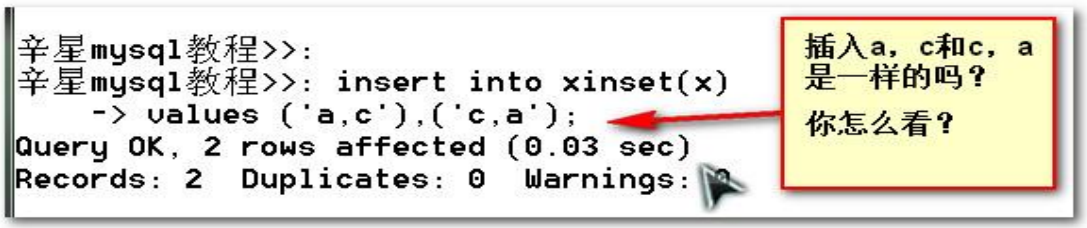
6.对于5这一条，既然我已经说开了，就会说清楚，绝对不说一半，绝对不会让大家花两分钟看完，迷惑一生。

**********检索特定值**********

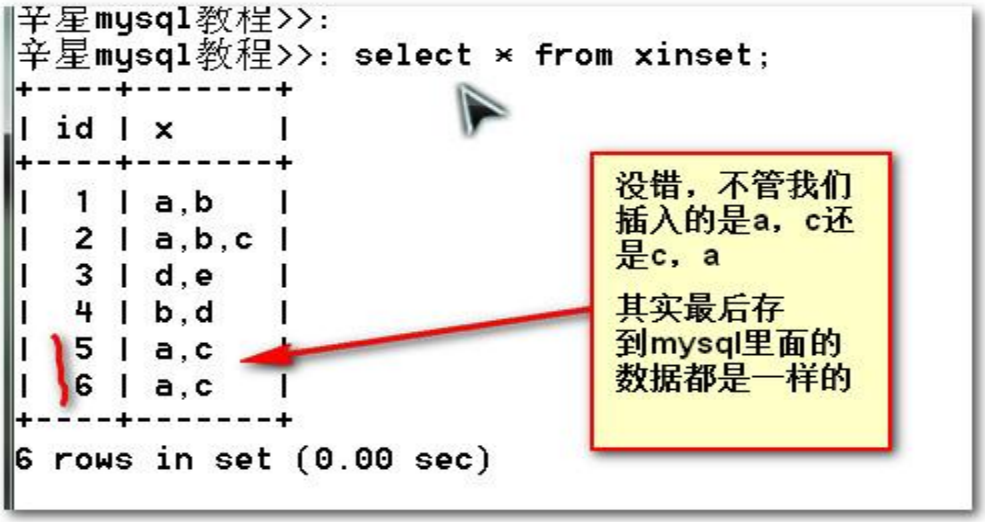
1.其实我们还支持一种检索方式，那就是使用=来连接，它表示相等，比如我们就像要x字段中存的是a和c的值的记录。

2.此时我们需要首先考虑一个问题，就是“a，c”和“c，a”是否相等？大家可以想一下，按理来说，应该是相等才对，那么是否相等呢？咱们实例操作一下。

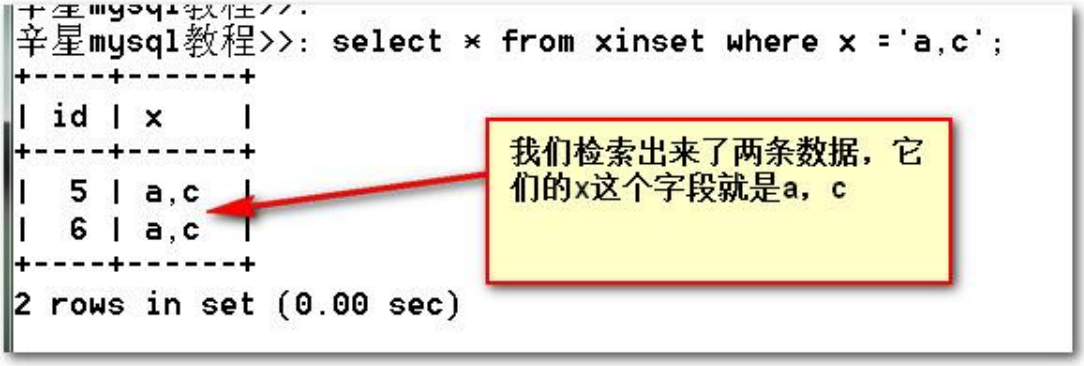
3.我们首先向里面插入这样的数据：



4.然后我们发现我们插入的数据是这样子的：



5.既然这一点明确了，那我们就把这两条信息检索出来吧：



**********内部机制**********

1.一个 set 类型最多可以包含 64 项元素。

2.在 set 元素中值被存储为一个分离的“位”序列，这些“位”表示与它相对应的元素。

3.“位”是创建有序元素集合的一种简单有效的方式，并且它还去除了重复的元素，所以 set 类型中不可能包含两个相同的元素。

**********set 的内部实现**********

1.一个 set 类型最多可以包含 64 个元素，而且它们的取值都是一个整数，这个整数的二进制码表名该集合的哪些成员为真。

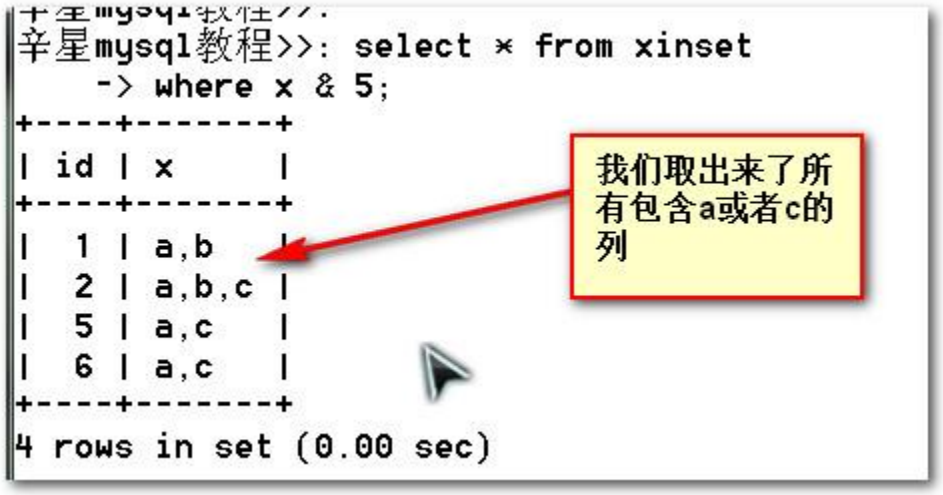
2.就以我们上面的 set('a','b','c','d','e')，那么它们对应的值的列表为：

集合元素	二进制表示	十进制表示
a	00001	1
b	00010	2
c	00100	4
d	01000	8
e	10000	16

3.比如我们想表示一个集合中含有 a 元素和 c 元素，那么它的二进制表示是：00101，那么它的十进制表示就是 5.

4.因此，我们分析这么一个表达式 x & 5 ，它这里就是把 x 的标志位与 00101 比较一下，如果两者都为 1，则该位为 1，只要有一个为 0，则该位为 0，因此，它的结果会导致只要 x 中含有 a 或者 c，那么就会被筛选出来。

5.因此，经过上述分析，我们可以得到下面的结果：



6.大家应该明白我上面的【where x & 2】表示什么意思了吧，它是把所有带有 2 的都拿出来。

第六节：补充说明和 null

**********英语单词*****null **********

1.我一开始看到这个单词的时候，感觉汉语发音是“牛”，后来问了问别人怎么读，有人读作“囊”。

2.后来我越发困惑，去百度发音了查了下，发现读音应该读作“耨”，发音和“no”是很相似的，也就是/nʌl/，也有音标是这么写的[nʌl]，好像我上学的时候音标是用两个斜线标记的，但是我看到不少书也用中括号标记。

3.null 的含义本身就很神秘，本意是“空的，元素为零”的意思，计算机编程中通常用它表示空值、无结果、空集合。

**********编程语言**********null**********

1.C 语言中有个大名鼎鼎的 null 指针，就是说无类型指针，且值为 0，成为悬空指针。

2.PHP 在 php4 版本以上也引入了 NULL，NULL 类型唯一的值就是 NULL，它表示一个变量没有值，有可能这个变量被赋值为 NULL，或者未被赋值，或者被 unset 掉了。

3.Java 中也支持 null 这个关键字，它可以用来声明一个空引用，如果我们引用了它，就会抛出一个空指针异常，哈哈，是不是很多 Java 程序员对“指针”都特别头疼？

**********数据库**********null**********

1.咱们数据库中也支持 null，它表示未知的数据。

2.如果您非要死扣定义，那么它可以细分为三种情况。

3.第一种，数据存在，但是不知道它的值。

4.第二种，不知道数据是否存在，mysql 自动填充为 NULL。

5.第三种，数据就是不存在，我们给一个字段赋值为 NULL。

6.其实死扣定义没啥意思，总之，NULL 表示“不知道确定值”。

**********null**********空字符串**********

1.空字符串表示信息存在，是一个确定信息，就是一个‘’，而null表示不知道是多少。

2.因此判断空字符串和null是否相等，会有如下结果：

3.我们判断一个字段是否为空，可以用 is null，比如我们判断：

4.对于一个字段，咱们也可以用is null来判断其是否为空。

**********存储**********null**********

1.还记得我们建表的时候有一个选项把：not null 表示不允许为null，而默认是可以为null的。

2.我们可以插入一个null值，比如【insert into xx(id,name) values (222,NULL);】就插入了一个空值。

3.查询的时候要用is null，不要用=null，因为=null会返回一个null，如果读者不太明白，可以看下面专栏。

**********is null **********= null**********

1.对于 is null 和= null，我怕大家晕倒，特别说明一下。

2.大家看一下效果就明白了：

3.我们用同样的 null 去判断，那么会发现，null is null 得到的是一个布尔值，但是 mysql 是没有布尔类型的，因此它会自动转化为 0 或者 1 来显示。

4.而 null = null 返回的也是一个 null，因为本来就是两个不确定的数据，比较一下，结果还是不知道。

5.就像我不知道两个陌生人的名字，你过来问我说他们是不是叫一样的名字，我只能很遗憾的说：我也布吉岛。

********** 内部机制**********null**********

1.请允许我抄一段 mysql 官方的解释：null columns require additional space in the row to record whether their values are NULL。for MyISAM tables,each NULL column takes one bit extra， rounded up to the nearest byte。

2.什么意思呢，也就是说，我们存储一个 null 值是需要额外的空间的，而我们存储空值，是不需要额外的空间的。

3.这也就导致了为什么我们说 not null 的效率比 null 高一些，因为 mysql 在进行比较的时候，null 会参与字段的比较，因此他会影响效率。

4.我们后面会讲到索引，B 数索引是不会存储 null 值的。

5.不过这一本书不会涉及索引，因为我把它分到了 mysql 优化那部分。

6.不过索引并不难，大家可以百度下，看个十几篇博客，基本就掌握的差不多了。

**********补充说明**********

1.不同的数据库的数据库类型有可能有差别。

2.比如有些数据库中的 float 都是不指定小数点位数的，而且有些数据库中的 decimal 也不是定点数，而是浮点数等等。

3.希望大家在学习一个新的数据库的时候一定要去了解下它的数据类型，不要全凭借已有的数据库的知识去认识一门新的数据库系统。

**********小广告**********

1.如果您感觉辛星的教程还可以的话，不妨去我的博客空间转转啦：blog.csdn.net/xinguimeng。

2.辛星，期待您的关注。

3.私人 QQ：1808347923.

4.私密邮箱：xinguimeng@163.com 。

第六部分：杂项内容

第一节：字符集

**********ASCII**********

1.目前计算机中应用的最广泛的字符集及其编码，是 American Standard Code for Information Interchange，即“美国标准信息交换码”。

2.它被国际标准化组织 ISO 定为国际标准，称为 ISO 646 标准，适用于所有拉丁文字母。

**********UNICODE**********

1.这种编码是将世界上所有的符号都纳入其中，用来解决乱码问题，这就是 Unicode。

2.Unicode 当然是一个很大的集合，它的规模大约可以容纳一百多万个符号。

3.Unicode 只是一个符号集，它只是规定了符号的二进制码，并没有规定这个二进制码该如何存储。

4.于是出现了 N 多种存储方式，也就是说有许多种不同的二进制格式，它们都是 Unicode。

5.其中 utf-8 是其中的一种相当优秀的实现方式。

**********Unicode 编码**********

1.我们计算机上通常所说的 Unicode 编码指的是 UCS-2 编码方式，即直接用两个字节存入字符的 Unicode 码。

2.它通常还分为小尾存储和大尾存储，如果没写，默认为小尾存储。

**********UTF-8**********

1.UTF-8 最大的一个特点，就是他是一种变长的编码格式。

2.它可以使用 1 到 4 个字节表示一个符号，根据不同的符号来变化字节长度。

3.因为本书并不致力于如何使用 UTF-8 编码根据相应的字节码得到对应的字符，因此，这部分内容暂时跳过。

**********ANSI 编码格式**********

1.它不是某一种编码，而是指本地编码。

2.ANSI 是默认的编码方式，对于英文文件是 ASCII 编码，对于简体中文通常是 GBK。

3.现在的操作系统都支持 Unicode。

**********中文编码**********

1.早期的计算机智能处理英语，使用 7 位的 ASCII 编码。

2.为了处理汉字，程序员设计了用于简体中文的 GB2312 和用于繁体中文的 BIG5.

3.GB2312(1980 年)一共收录了 7445 个字符，包括 6763 个汉字和 682 个其他符号。

4.1995 年的汉字扩展规范 GBK1.0 收录了 21886 个符号，它分为汉字区和图形符号区，汉字区包括 21003 个字符。

5.从 ASCII 到 GB2312 再到 GBK，这些编码方法是向下兼容的，即同一个字符在这些方案中的编码总是相同的，但是后面的标准支持更多的字符。

6.2000 年的 GB18030 是取代 GBK 的正式国家标准，该标准收录了 27484 个汉字，同时还收录了藏文、蒙文、维吾尔文等主要的少数民族文字。

**********GB18030**********

1.GB18030 扩展至 27484 个汉字。

2.GB18030 的编码采用单字节、双字节和四字节方案。

**********总之**********

1.BIG5 支持繁体中文，GB2312 支持简体中文，它们的历史都比较早。

2.BIG5 和 GB2312 是 GBK 的子集。

3.GBK 是 GB18030 的子集。

**********经验**********

1.GBK 为汉字量身打造，存储汉字会占用更小的空间。

2.UTF8 是国际编码，它的通用性比较好。

**********mysql 乱码问题原因**********

1.第一，Server 本身设定问题。

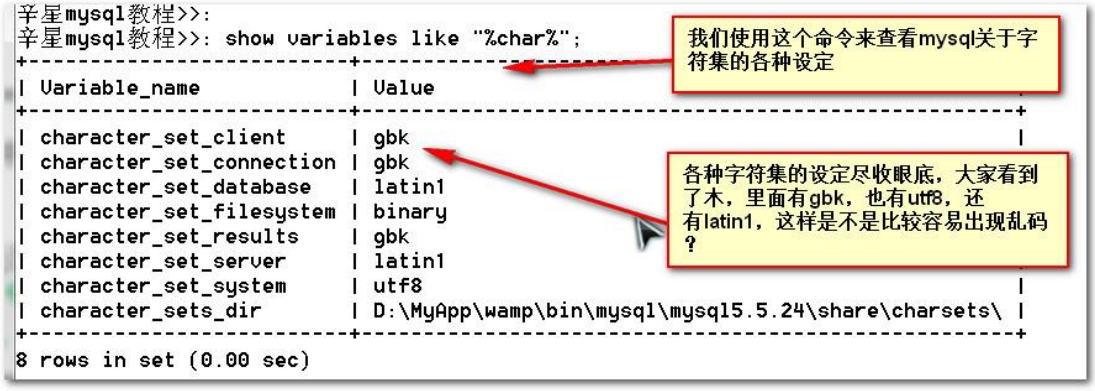
2.第二，table 的语系设定 问题(包含 character 与 collation)

3.第三，客户端程式的连线语系设定问题。

**********查看设定**********

1.咱们可以用 show variables like "%char%"来查看。

2.如下截图：



3.当然上面给出了八条信息，这里我为大家重点解释其中的四条：

character_set_server：设置服务器使用的字符集。

character_set_client:设置客户端发送查询使用的字符集。

character_set_connection:设置服务器需要将收到的查询串转换成的字符集。

character_set_results:设置服务器要将结果数据转换到的字符集。

**********一条特殊语句**********

1.这个语句就是【set names xxx】这里的 xxx 是字符集名。

2.比如 set names utf8 的效果等价于同时设定如下：

set character_set_client = 'utf8';

set character_set_connection = 'utf8';

set character_set_results='utf8';

**********校对集**********

1.校对集可以理解为某个字符集的排序规则。

2.一个字符集至少有一个校对集，一个校对集也肯定会属于一个字符集。

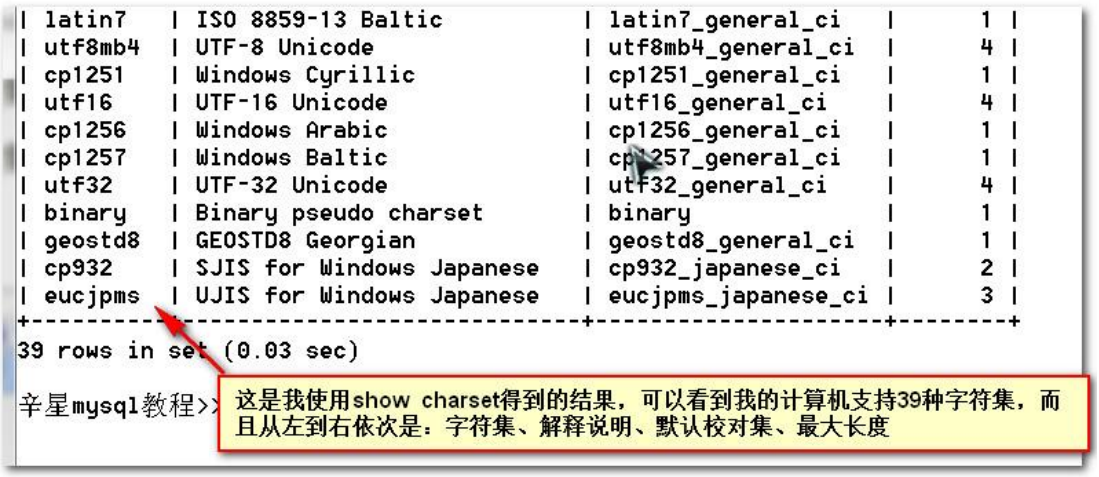
3.一个字符集都有一个默认校对集，因此很多时候我们指定字符集即可，可以不指定其校对集。

**********查看字集**********

1.查看字符集：show charset;

2.查看校对集：show collation;

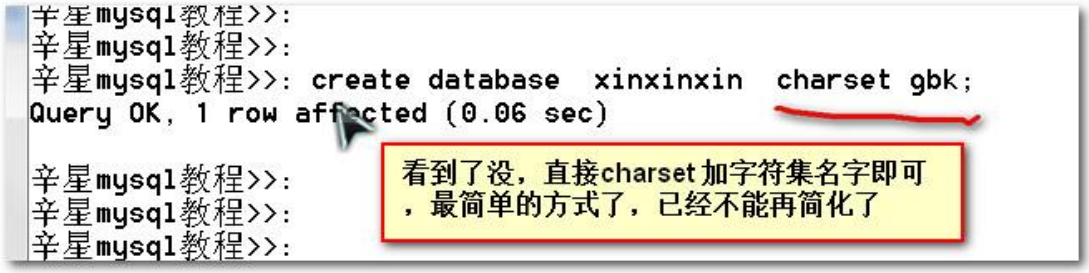
3.下面是我的 windows 机器上使用 show charset 得到的结果：



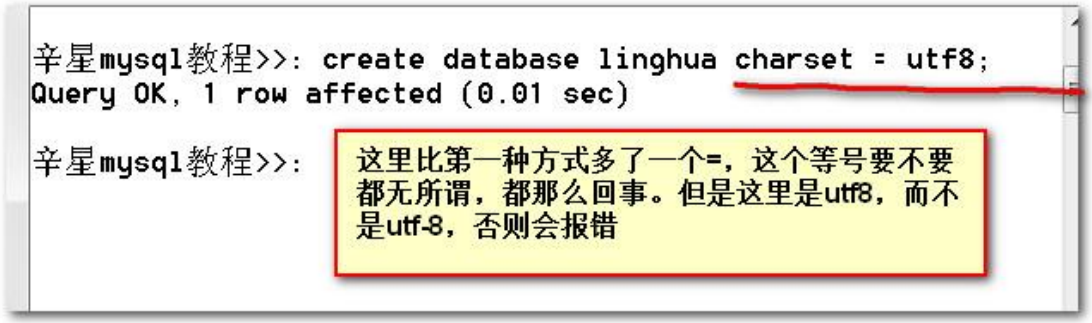
**********指定数据库的时候指定字符集**********

1.创建数据库的时候指定字符集，不过这个它支持好几种格式，我说几种比较好用的，这里的好用就是书写简单。

2.第一种就是最简单的了，它直接使用 charset 后面跟着字符集的名字即可：

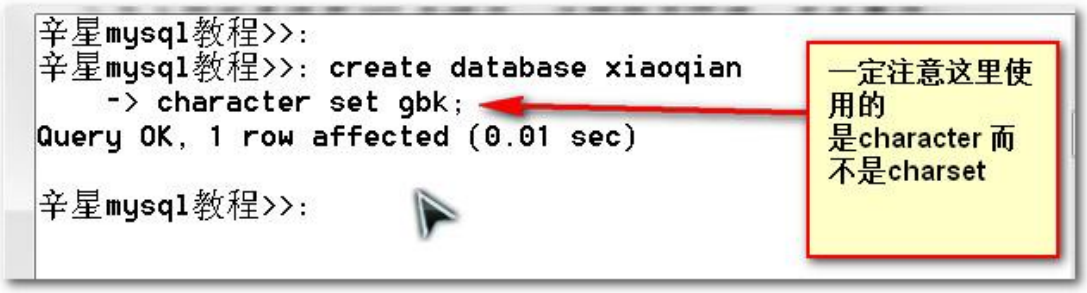


3.第二种就是稍微麻烦一点点啦，它还得再加一个=，看我操作奥：



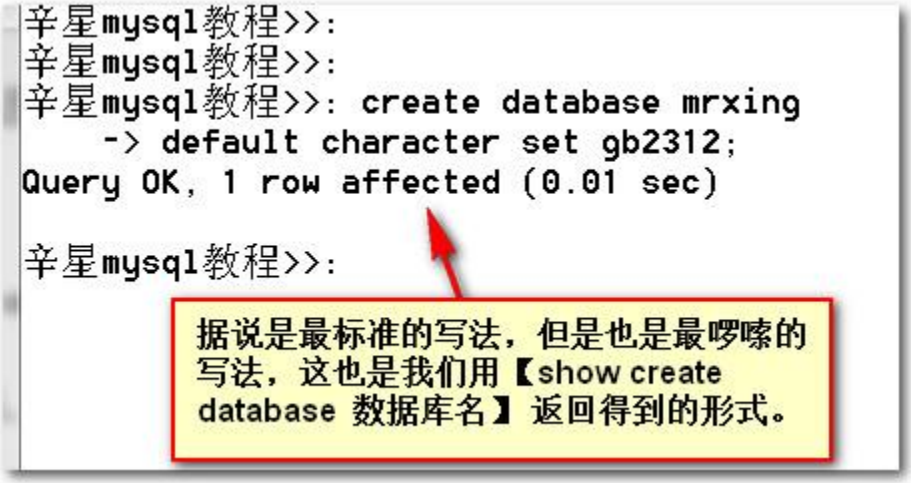
4.这里说明一下，这里必须用 utf8 而不是 utf-8，否则会报错的。

5.第三种就是使用 set 关键字，这种稍显啰嗦，看我操作：



6.注意第三种的是 character set gbk 而不是 charset set gbk；

7.第四种也是我看过的最啰嗦的，不过没办法啦，其实这是最标准的写法：



8.好吧，说了这么多，我只是想说：由于 mysql 强大的兼容性，读者不论使用哪一个，都能正常通过，但是不要记混。

**********建表的时候指定字符集**********

1.我决定建表这一块就说一个，我不想让大家看了头大。

2.通用格式：

create table 表名(列名 1 数据类型 1 列级完整性约束，.....)

charset 字符集名 ；

3.嘎嘎，这是最简单的，没有之一。

**********mysql 的设定**********

1.有时候我们会遇到一些问题，这些问题就是一些规定。有时候问为什么，答案是没有为什么，有的数据库就不这么规定，但是我们就这么规定。

2.mysql 从 4.1 开始就设置了如下四个级别的编码格式。

3.第一级别：服务器级别，如果为服务器指定了一个编码格式，在创建数据库的时候，如果没有特殊说明，都是使用该编码格式。

4.第二级别，数据库级别，如果为数据库指定了一个编码格式，在创建表的时候默默人编码格式就是数据库的编码格式，可以指定某个表的编码格式。

5.第三级别，数据表级别，可以为数据表指定一个编码格式。

6.第四级别，数据列级别，指定这一列数据使用何种编码格式。

**********修改表的编码格式**********

1.对于修改表，我们使用 alter 语句，我并没有讲，如果大家有兴趣，可以去百度下“alter mysql”来看看相应的介绍。

2.修改表的编码格式：alter table 表名 charset 新编码格式。

3.修改一个列的编码格式：

alter table 表名 change 旧列名 新列名 列类型 列级完整性约束。

4.注意这里的列级完整性余数里面必须包含编码格式。

5.所谓的列级完整性约束，就是列修饰符。

第二节：事务初步

**********为什么是初步**********

1.事务这个东西，不花费一段精力是很难彻底吃透的，否则只能是浅尝辄止。

2.其实我最不喜欢的就是浅尝辄止，因为这样吧，也算懂，但是，出了问题又解决不了，和不懂有什么区别？我感觉是：没什么区别。

3.但是我计划把事务放到了第二本中详细讲解，因此在第一本中先大致讲一下最基本的知识，这样到第二本也有一个更好的过度。

4.而且事务本身就是一个比较重要的概念，由于事务的使用比较广泛，不会事务是不行的。

**********为什么要引入事务**********

1.这是一个老掉牙的例子，好像我在学 mysql 的时候，它就有了，但是也是因为经典，我再给大家描述一遍。

2.比如用户 A 给用户 B 转账 100 元，我们可以先在表里先修改账户 A 的钱的数据，然后再修改账户 B 的钱的数据。

3.比如旧数据：账户 A 有 400 元，账户 B 有 200 元。

4.当我们给用户 A 减去 100 元，但是还未修改账户 B 的数据的时候，突然机器宕机了，那么此时的数据：账户 A 有 300 元，账户 B 有 200 元。

5.虽然这种情况并不是太多见，但是，对于商用程序来说，丝毫的错误都可能是致命的，因此，我们必须杜绝这种情况的出现。

6.因此，我们引入了事务的概念，它可以很好的解决这种问题。

7.不过我们还是先了解一些最基本的概念吧，这样会让我们向后的学习更加轻松。

**********事务的特点**********

1.事务有四大特点，简写为：ACID。

2.A 是 atomic，即原子性，组成一个事务的 sql 语句要么全部执行，要么一条也不执行。

3.C 是 consistent，有人翻译为“稳定性”，也有人翻译为“一致性”，就是要么全是旧数据，要么全是新数据，我们认为这两个都可以，即只存在“失败和成功”，不存在“成功了一部分”，即新数据和旧数据同时存在。

4.I 是 isolated，大部分翻译为孤立性，SQL 标准支持四种，我们后面会介绍。

5.D 是 durable，有人翻译为可靠性，也有人翻译为持久性。

**********事务和引擎**********

1.事务的实现是需要依赖于引擎的实现的，MyISAM 是不支持事务的，InnoDB 是支持的。

2.如果我们需要使用事务，我们通常会选择 InnoDB 引擎。

3.我们下一节会简单的介绍一下引擎。

**********事务的流程**********

1.第一步通常就是开始一个事务，我们要么提交一个事务，或者回滚一个事务，这中间的操作会被当做一个整体。

2.如果提交事务，相当于事务成功了，那么数据库会存储完成之后的数据。

3.如果回滚事务，相当于事务失败了，那么数据库就会存储事务开始之前的数据，就像从事务开始到现在的操作没有进行过一样。

**********事务的 SQL 语句**********

1.开始一个事务：我们通常用 start transaction，也可以使用 begin 或者 begin work。

2.提交一个事务(即事务完成)；commit。

3.回滚一个事务(即事务失败)：rollback.

**********实例操作**********

1.首先我们创建一个表并插入两行数据：

2.然后开始事务：

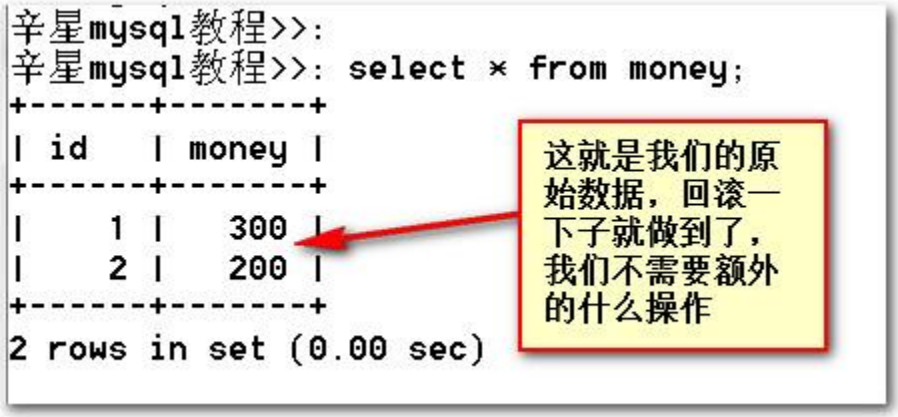
3.然后查看一下数据：

4.然后我们查看一下数据库的数据：

5.然后我们回滚事务，执行：

6.这种情况时有发生的，比如用户 1 和用户 2 本来是要交易的，结果两个人越谈越没得谈，最后买卖取消了，那么，就需要回滚事务来得到原始数据。

7.我们检索下数据库的数据，发现还是原来的用户 1 有 300，用户 2 有 200：

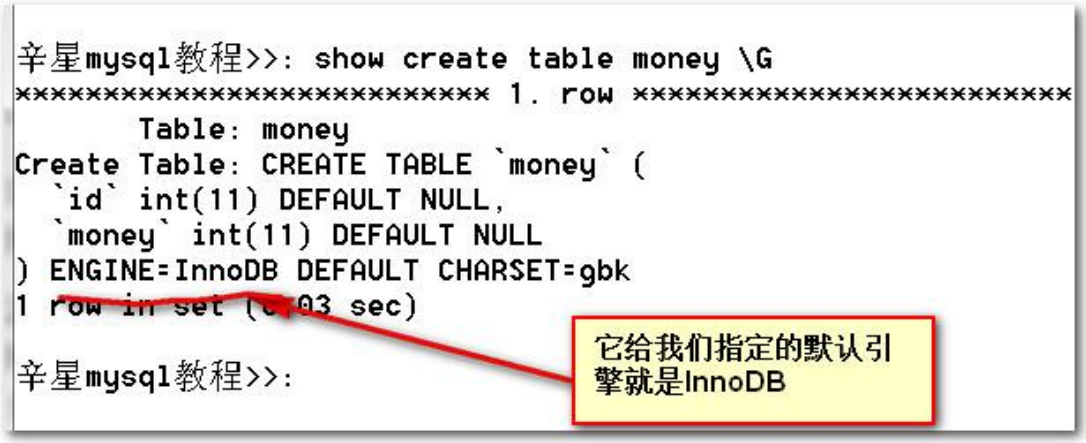


8.以上我给大家演示了一下事务的简单操作。

**********说明**********

1.可能有人说：你不是说只有 InnoDB 引擎支持事务，而 MyISAM 不支持事务码？

2.我们 show 一下：



**********后续知识简介**********

1.后续知识其实还有不少的，比如保存点、孤立级、锁定、自动提交等 n 多问题，我想放到第二本中去介绍。

2.我也逐渐理解了一些东西，那就是一门知识，自己懂不懂和能不能清晰透彻的让别人也懂，是两回事。

3.欢迎各位大神前来指教，比如我的教程的体系设置、知识层面等很多方面，都欢迎不吝指正。

4.当然也可能会产生少量错误，比如我的“2014 年辛星 php 教程夏季版”中就收到了不少读者反映 xxx 页 xxx 行有错别字。

第三节：存储引擎初步

**********说明**********

1.其实我们目前学习的都是最基本的操作，这些 sql 语句在大多数数据库系统中都很相似。

2.这些操作并不能体现 mysql 与其他数据库的差别，因为增删改查大家都差不多，大家都支持。

3.这些知识当我们从一个数据库转向另一个数据库的时候基本不怎么用学习，但是，有些知识，却需要我们去学习，这些知识是 mysql 所独有的。

**********存储引擎**********

1.在 mysql 里，有“存储引擎”这个概念，mysql 的存储引擎已经有 MyISAM、InnoDB、BDB、Memory、Merge、Cluster/NDB 等比较多的种类了。

2.这些存储引擎并无好坏之分，它们都有自己适合的环境。

3.各种引擎在性能、事务、并发控制、参照完整性、缓存、故障恢复、备份等几个方面都不是很一样，因此，很多时候我们必须根据存储引擎来进行讲解。

4.不过这本书是一本入门书，它也不致力于讲解 mysql 内部的实现机制，它只是让大家入门，我们在后续的课程中会讲解 mysql 的内核，敬请您的期待。

5.可以用 show engines 看一下本机支持的 mysql 引擎：


```
辛星mysql教程>>: show engines;
+--------------------+---------+----------------------------------------------
| Engine             | Support | Comment
+--------------------+---------+----------------------------------------------
| FEDERATED          | NO      | Federated MySQL storage engine
| MRG_MYISAM         | YES     | Collection of identical MyISAM tables
| MyISAM             | YES     | MyISAM storage engine
| BLACKHOLE          | YES     | /dev/null storage engine (anything you wr
| CSV                | YES     | CSV storage engine
| MEMORY             | YES     | Hash based, stored in memory, useful for
| ARCHIVE            | YES     | Archive storage engine
| InnoDB             | DEFAULT | Supports transactions, row-level locking
| PERFORMANCE_SCHEMA | YES     | Performance Schema
+--------------------+---------+----------------------------------------------
9 rows in set (0.00 sec)
```

*****MyISAM**********InnoDB*******************

1.这里只是说一下 MyISAM 与 InnoDB，为什么呢，因为新手朋友们用这两个引擎是最多的。

2.下面我做了一个表格来简要说明一下它们的区别：

特性	MyISAM	InnoDB
存储限制	无	64TB
事务安全	不支持	支持
空间使用	低	高
内存使用	低	高
批量插入速度	高	低
锁机制	表锁	行锁
外键	不支持	支持

3.上面的表格并不全，关于缓存、索引等内容我都没有说，因为我感觉我们后面会介绍的很清楚。

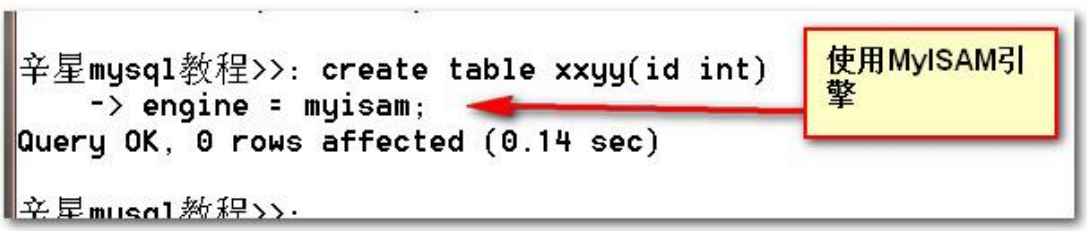
4.如果应用以读和插入操作为主，并且对事务、并发要求不高，选择 MyISAM 是不错的选择。

5.如果对事务的完整性要求较高，在并发条件下要求数据的一致性，有大量的增删改查，支持外键的时候使用 InnoDB 比较合适。

**********使用存储引擎**********

1.我这里默认设置的是 InnoDB 引擎，如果我想用 MyISAM 引擎。

2.我们只需要在建表语句之后加一个 engine=引擎名 即可：



第七部分：话外篇

第一节：我的学习方式

**********我的学习方式**********

1.我承认，我的大学并不算差，至少在985里能拍到前几名，但是我认为我的主要知识来源并不是来自课堂。

2.第一种就是通过视频，我也曾经看过不少教学视频，因为很多操作性强的知识，通过文本或者图片很难描述，但是视频很好很方便。

3.第二种就是通过书籍，很多时候在图书馆一呆就是一下午一晚上，感觉学到了不少。这里我特别推荐看一些大师的书，不是其他人写的不好，而是大师的理解更加深刻，更能一针见血的指出关键所在。还有就是尽量避开那些过时的书把，让它们沉睡在时代的长河中把，因为计算机工业的革新实在太快了。

4.第三种就是通过博客，也是在大约一年前还是两年前，我就特别喜欢在没事的时候，随便搜索一个关键词，比如“辛星csdn”，然后就在博客里面找自己感兴趣的开始阅读。说实话，还是有些博客内容是有一些高精尖的内容的。

5.第四种就是通过实战，我自始至终认为“站在岸上学不会游泳”，因此，要想真正的学会游泳，多在水里练习是非常重要的。

**********有毅力，当自学**********

1.当然，我认为跟着老师学习，跟着培训机构学习，跟着速成班去学习也都很不错。

2.因为我们看一个人知识的结构，发现还是自学的居多，因此，我建议大家多锻炼下自学的能力，这点灰常重要。

3.辛星系列就是为加速自学效率，帮助您扫清自学道路上的一切障碍而打造的，希望您能够喜欢。

4.当然我们也期待您的反馈，只有这样，我们才会进步。

第二节：加入我们

**********行动纲领**********

1.当我在南开读书的时候，就一直特别佩服周总理“为中华崛起而读书”的宏愿，有很多夜晚在总理像前徘徊。

2.我感觉我们这一代作为祖国的接班人，也应该有一颗报国之心，但是，怎奈能力有限，因此想到了在软件工业写这么一套教程，用实际行动来做到“传播编程知识，振兴中华软件”的目的。

3.它的内容非常广泛，涉及 Linux、php、Python、mysql、Java 等等，甚至还会包括批处理、vim、word 等这些编程周边的东西，而且它自始至终不是为了收费才写的教程，但是它不排斥其他人利用它收费。

**********中华编程知识传播联盟*****CPDL**********

1.我也有过建立这么一个俱乐部的想法，但是由于眼下时间太紧了，还没有时间去筹办这件事。

2.这个俱乐部建立之后，会对所有有中国籍的人免费开放，而且它绝对不收会员费，而且来去自由。如果它有盈利，它会给参与该项目的会员发劳务费的。用一句话来说，就是“稳赚不赔”。

3.该俱乐部的主要任务是维护一系列涉及编程方方面面的免费的、可共享的、更新速度较快、全面的教程。

4.它会有定期的内部公益活动，会通知会员，具体活动还未想好。

5.该俱乐部尚未建立，如果您想成为其创始人之一，不妨赶快给我发邮件奥：xinguimeng@163.com 。

**********辛星系列**********

1.辛星系列的书目就是在这个背景下成立起来的，它是我一个人独立完成的，它会一直更新并且绝对让所有人都免费下载。

2.但是毕竟我能力有限，一来知识有限，二来精力有限，我希望有更多的人加入进来。

**********您的品牌**********

1.当然，不可否认，“辛星”这个品牌的名气也不是很响，但是，我希望我们可以一起成长，打造我们的品牌。

2.不可否认，每个人技术水平有高有低，能力有大有小，但是，我们都可以打造我们的品牌，为编程知识的传播贡献一份自己的力量。

3.当然，我也可以以一个大哥哥的身份帮你一下，我也希望更多咱们国人的品牌出现，让编程初学者少走弯路。