我的MySQL学习总结

目录

[第一章 背景知识 4](#_Toc492397630)

[1、 数据库 4](#_Toc492397631)

[2、 SQL 4](#_Toc492397632)

[3、 MySQL 5](#_Toc492397633)

[第二章 创建和使用数据库 5](#_Toc492397634)

[1、 创建MySQL数据库 5](#_Toc492397635)

[2、 使用MySQL命令行工具 7](#_Toc492397636)

[3、 MySQL数据类型 7](#_Toc492397637)

[4、 创建表 8](#_Toc492397638)

[5、 操作和修改 8](#_Toc492397639)

[第三章 查询 9](#_Toc492397640)

[1、 查询机制 9](#_Toc492397641)

[2、 查询语句 10](#_Toc492397642)

[3、 select 10](#_Toc492397643)

[4、 from 11](#_Toc492397644)

[5、 where 12](#_Toc492397645)

[6、 group by和having 12](#_Toc492397646)

[7、 order by 12](#_Toc492397647)

[第四章 过滤 13](#_Toc492397648)

[1、 条件评估，以及条件构建 13](#_Toc492397649)

[2、 条件类型 13](#_Toc492397650)

[3、 null关键字 14](#_Toc492397651)

[第五章 多表查询 14](#_Toc492397652)

[1、 什么是连接 15](#_Toc492397653)

[2、 连接3个或者更多的表 15](#_Toc492397654)

[3、 自连接 16](#_Toc492397655)

[4、 相等连接和不等连接 16](#_Toc492397656)

[5、 连接条件和过滤条件 16](#_Toc492397657)

[第六章 使用集合 17](#_Toc492397658)

[1、 集合论 17](#_Toc492397659)

[2、 集合操作符 17](#_Toc492397660)

[3、 集合规则 17](#_Toc492397661)

[第七章 数据生成、转换和操作 18](#_Toc492397662)

[1、 使用字符串数据 18](#_Toc492397663)

[2、 使用数值数据 19](#_Toc492397664)

[3、 使用时间数据 19](#_Toc492397665)

[4、转换函数 19](#_Toc492397666)

[第八章 分组与聚集 19](#_Toc492397667)

[1、 分组的概念 19](#_Toc492397668)

[2、 聚集函数 20](#_Toc492397669)

[3、 产生分组 20](#_Toc492397670)

[4、 分组过滤条件 21](#_Toc492397671)

[第九章 子查询 22](#_Toc492397672)

[1、 什么是子查询 22](#_Toc492397673)

[2、 非关联子查询 22](#_Toc492397674)

[3、 关联子查询 23](#_Toc492397675)

[4、 何时使用子查询 24](#_Toc492397676)

[第十章 连接—进阶 24](#_Toc492397677)

[1、 外连接 24](#_Toc492397678)

[2、 交叉连接 24](#_Toc492397679)

[3、 自然连接 24](#_Toc492397680)

[第十一章 条件逻辑 24](#_Toc492397681)

[1、 条件逻辑 24](#_Toc492397682)

[2、 Case表达式 25](#_Toc492397683)

[3、 使用case表达式 25](#_Toc492397684)

[第十二章 事务 25](#_Toc492397685)

[1、 多用户数据库 25](#_Toc492397686)

[2、 什么是事务 26](#_Toc492397687)

[第十三章 索引和约束 26](#_Toc492397688)

[1、 索引 26](#_Toc492397689)

[2、 约束 26](#_Toc492397690)

[第十四章 视图 27](#_Toc492397691)

[1、 介绍 27](#_Toc492397692)

[2、 为什么使用视图 27](#_Toc492397693)

[3、 可更新的视图（使用update或者 insert语句更新） 27](#_Toc492397694)

[第十五章 元数据 27](#_Toc492397695)

[1、 定义 27](#_Toc492397696)

[2、 使用元数据 28](#_Toc492397697)

1. 背景知识
2. 数据库

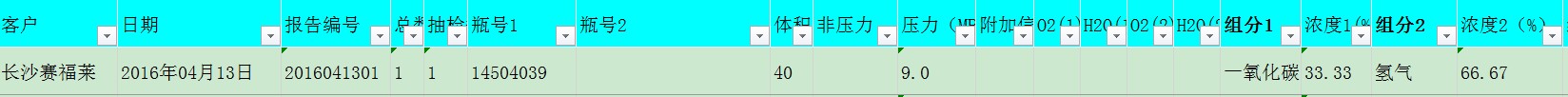
什么是数据库？数据库就是数据信息的集合。例如电话本也是数据库，其中每一个电话号码就是数据条目。

数据库分很多种，大体分为关系型数据库和非关系型数据库。

非关系型数据库，包括层次和网状型数据库。

一般来说，现在用的最广泛的是关系型数据库，它利用信息的冗余来表示数据之间的关系。

例如下图



关系型数据库可以将这个条目分为3个表（table）

客户订单信息表：客户名，订单日期，订单数量，瓶号

组分表：组分1及其浓度，组分2及其浓度，瓶号

包装物信息表：体积，压力，瓶号

其中，“瓶号”这一信息是冗余的，也是可以通过它来找到唯一的对应表，并将他们连接起的关键信息。

1. SQL

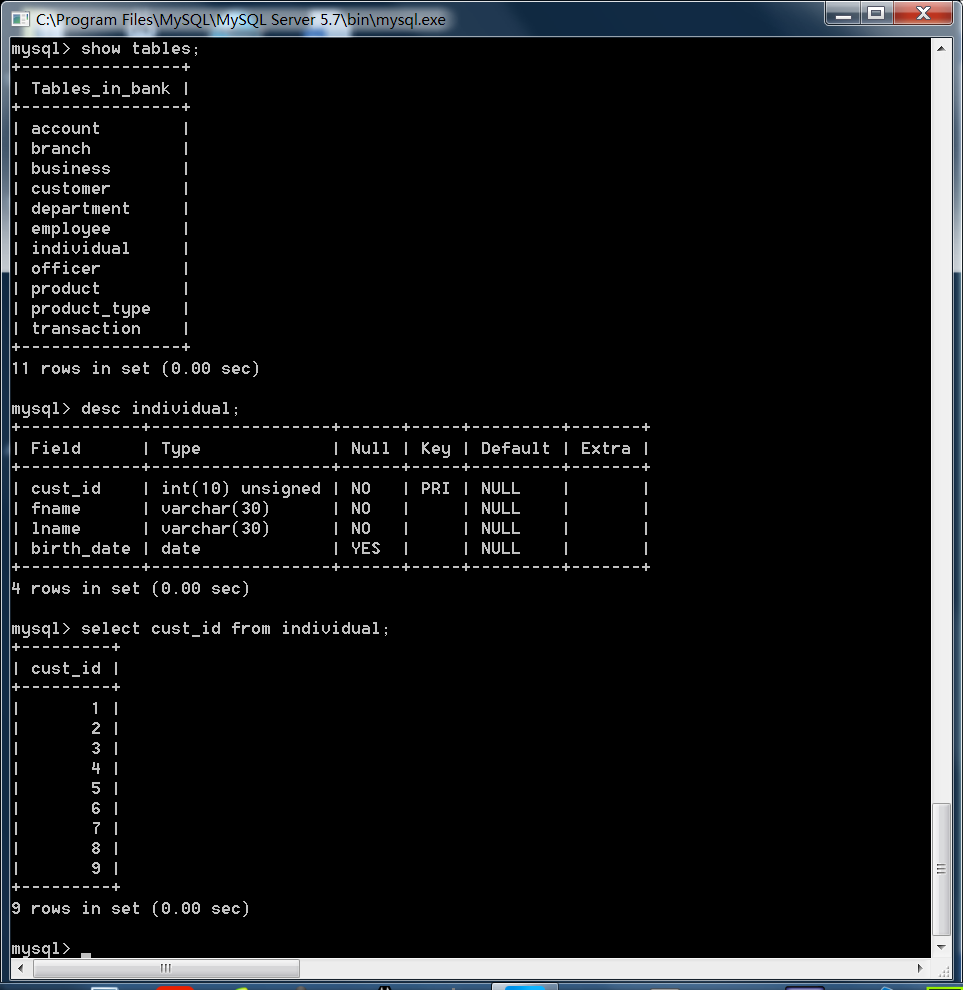
就是sequel语言。

其中分为几个独立模块：SQL方案（schema）语句，用于定义其中的数据结构，是关于数据的数据；SQL数据语句，用于操作SQL方案语句所定义的数据结构（增删改查）；SQL事务语句，用于开始、结束或者回滚事务。

SQL是“非过程化语言”。也就是说，只有输入和输出，过程由数据库管理员或者性能调优专家设置好的“优化器”来解决。

特别地，平时用的最多的是SQL数据语句，分别是select、insert、update和delete。

以下，用MySQL作一个示范



1. MySQL

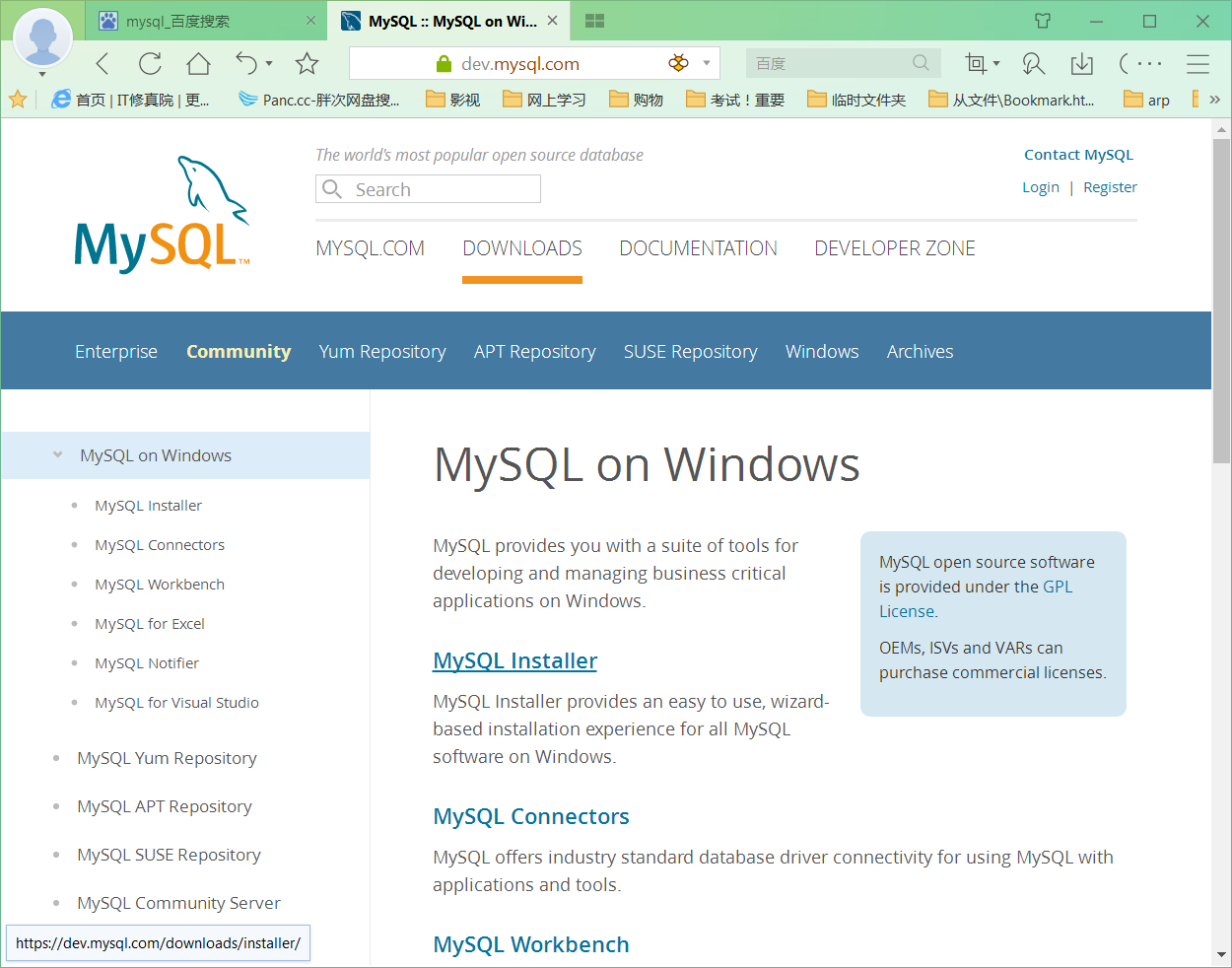
数据库并不是一味选择最强大的软件就好，而是应该因时因地制宜。如果只是保存电话信息，一个电话本就好，如果保存客户信息，10万条以内，excel完全可以胜任，但是如果要保存百万以及更大的数据量，最好使用商业软件Oracle。

而MySQL是开源的，首先比较便宜（免费使用），而且安全性也不错，比较适合小公司使用。当然，由于MySQL是开源的，技术强如阿里巴巴之类的公司，可以将其定制得比Oracle更加强大。

1. 创建和使用数据库
2. 创建MySQL数据库

首先，去MySQL官网下载客户端。

<https://dev.mysql.com/downloads/windows/> 如图

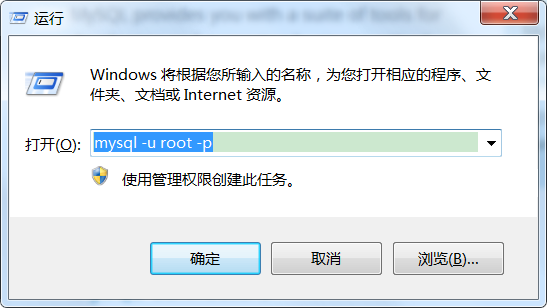


这个是windows系统的，选择其中 MySQL Installer

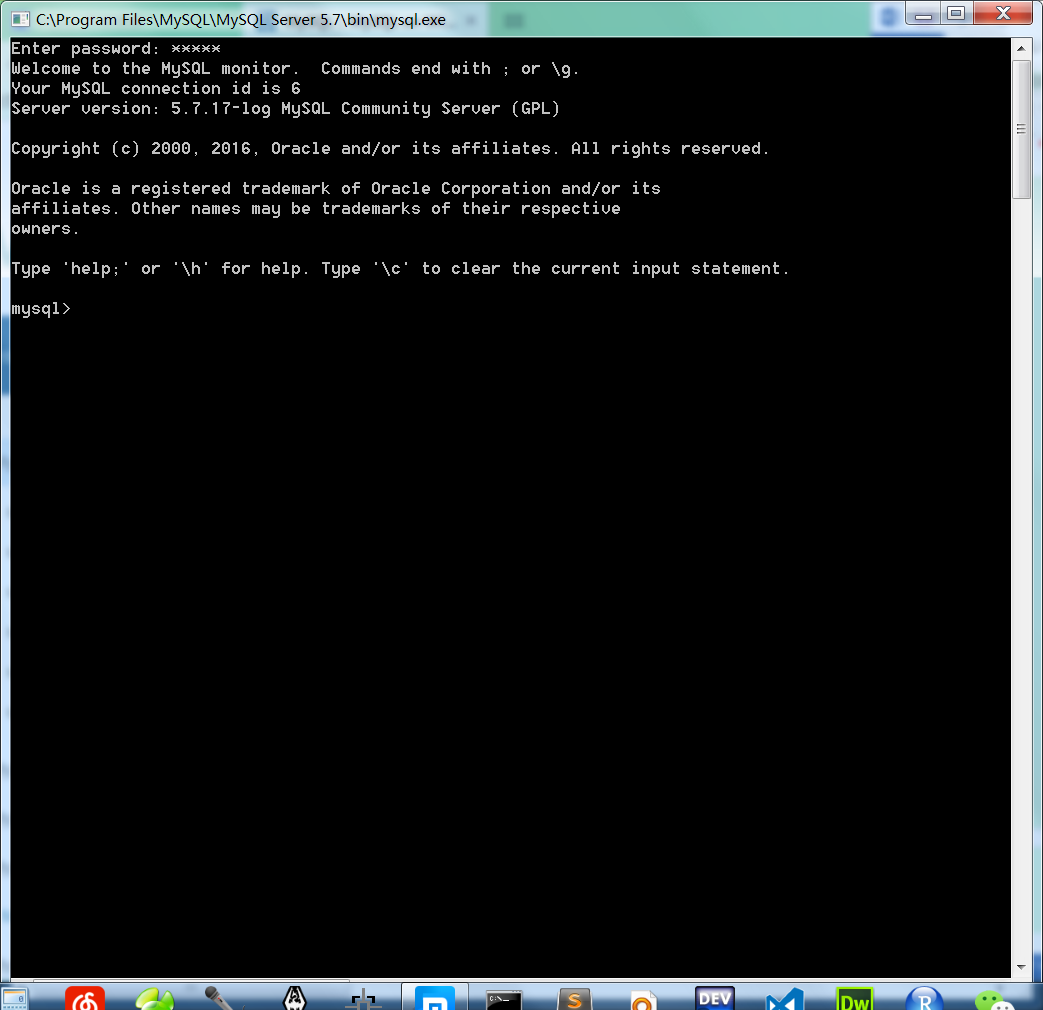
安装。（安装的最后阶段，会提示你输入root账户的密码，现在暂定为32147）

特别注意的是，安装完成之后需要在 我的电脑—高级设置—环境设置 中的 path中加入MySQL的路径，由于本次选择的是安装版，所以系统自动增加了。

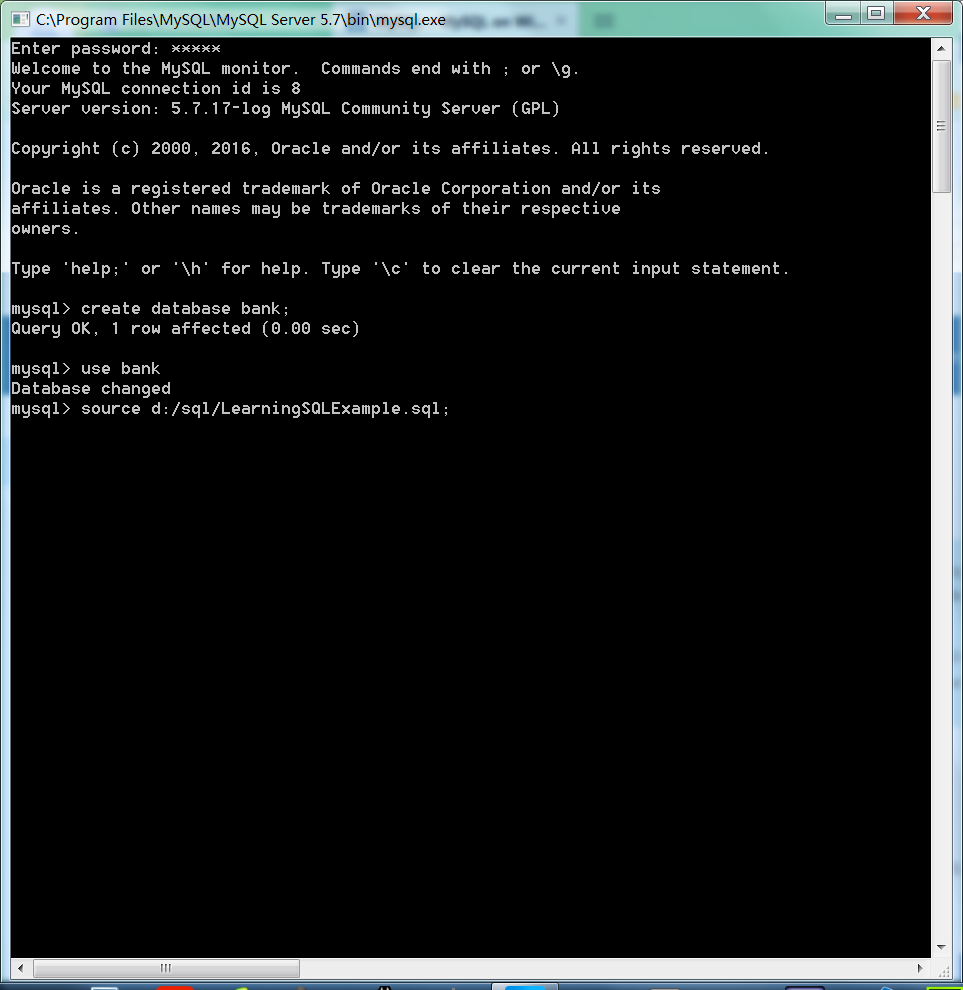
安装完成之后，如图输入命令



然后输入root账号的密码 32147 显示登陆成功，如图

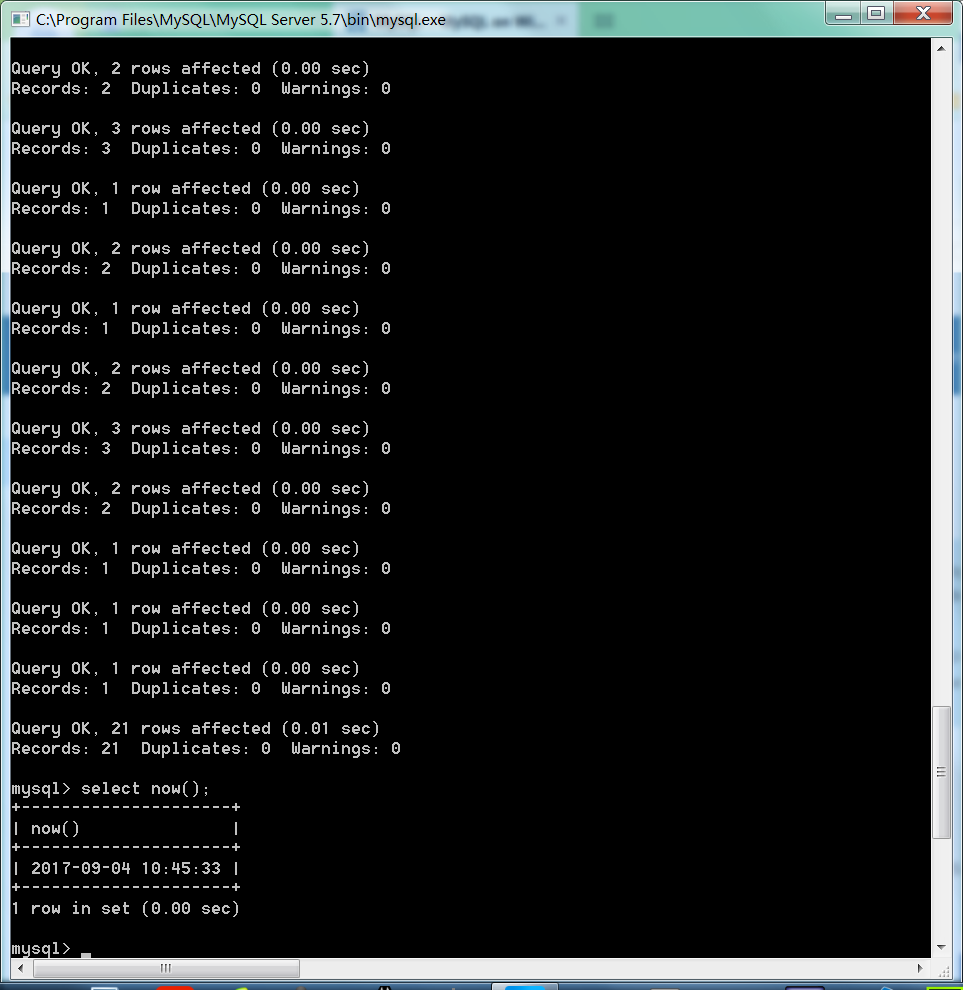


再按图输入命令，用作示例的数据库建立完成。（附件）



1. 使用MySQL命令行工具

MySQL有很多UI工具，但是最多时候，使用还是在命令行模式中。特别地，在使用一些内置函数时，MySQL并不需要指定数据表。例如下图。



1. MySQL数据类型

字符型：varchar（xx）是可以自动变长适应的，char（xx）则不是。

文本数据：如果需要储存的数据大于varchar的上限64Kb，就要使用文本类型，mediumtext或者longtext。

数值型：整数，浮点数等等，特别地，unsigned表示正整数等等（被unsigned修饰的不能放入负数）。

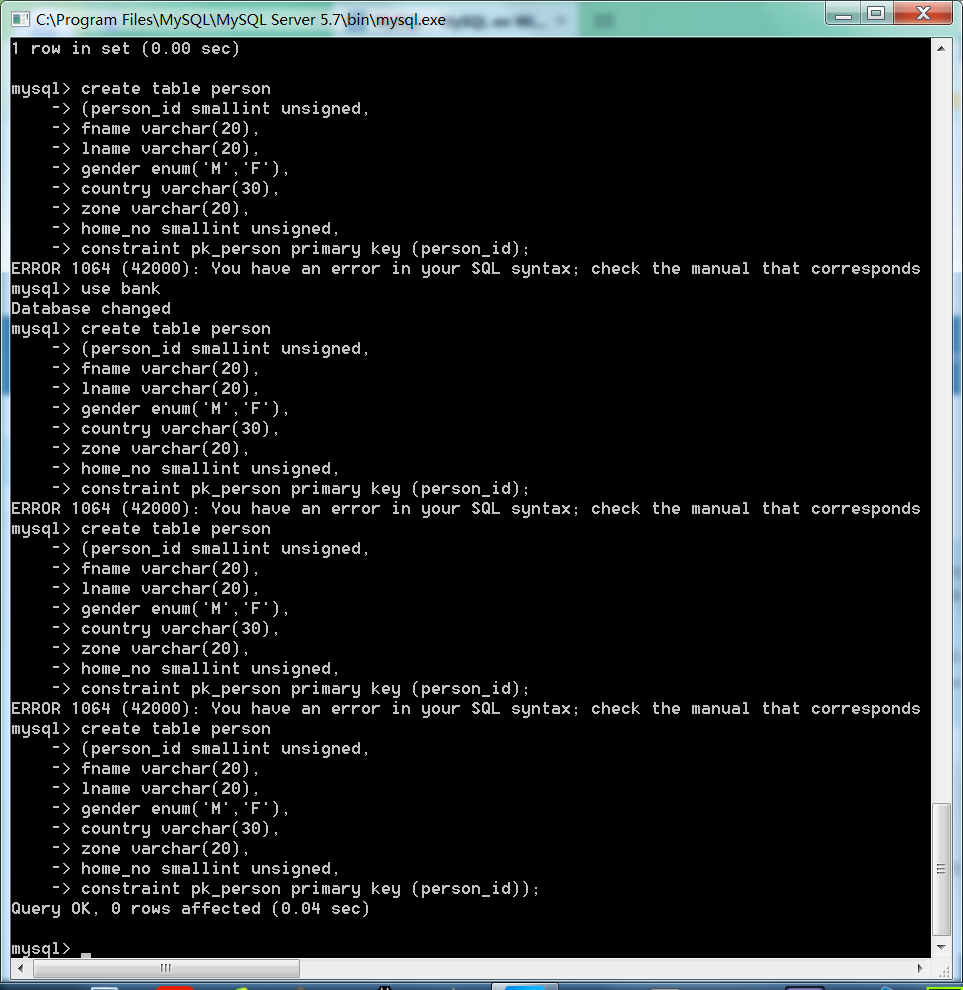
时间数据：date，time，timestamp等等。

使用的时候按照需要选择合适的数据类型。

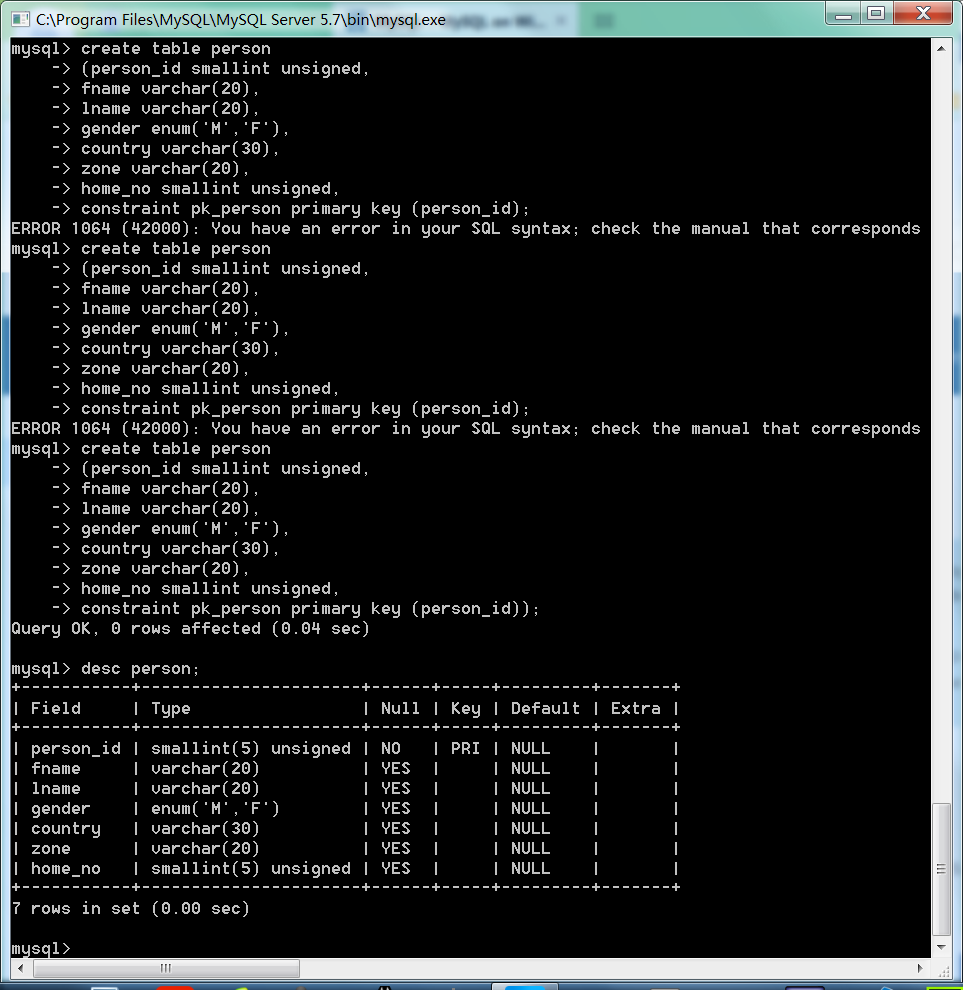
1. 创建表

创建表之前，最好设计一下表的样式和分类。一个良好的表，项目一定是规范化，易懂而且每一列都不是复合的对象。例如现在创建一个人物信息表，其中包含身份证号，姓名，性别和地址。

其中地址应该可以拆分为国家、地区、门牌号，所以我们可以如下图一样创建一个表：



我们可以使用desc命令查看这个表，例如

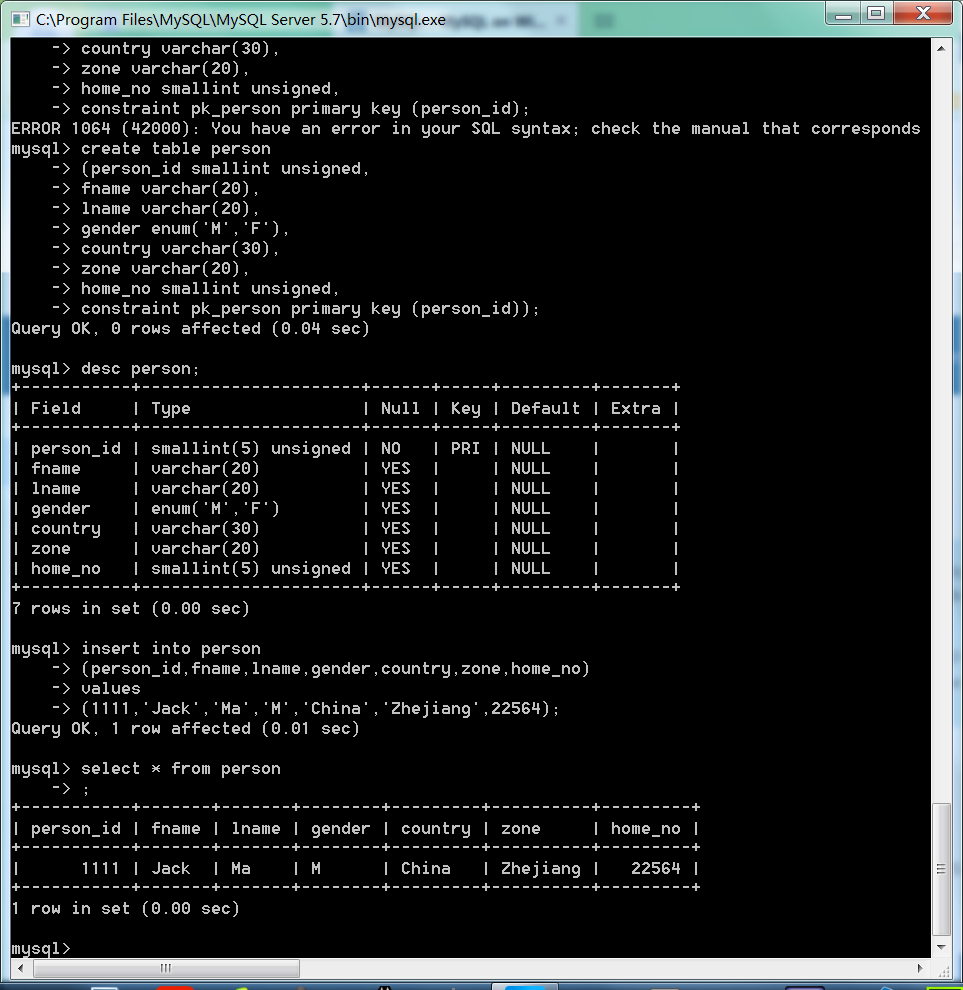


1. 操作和修改

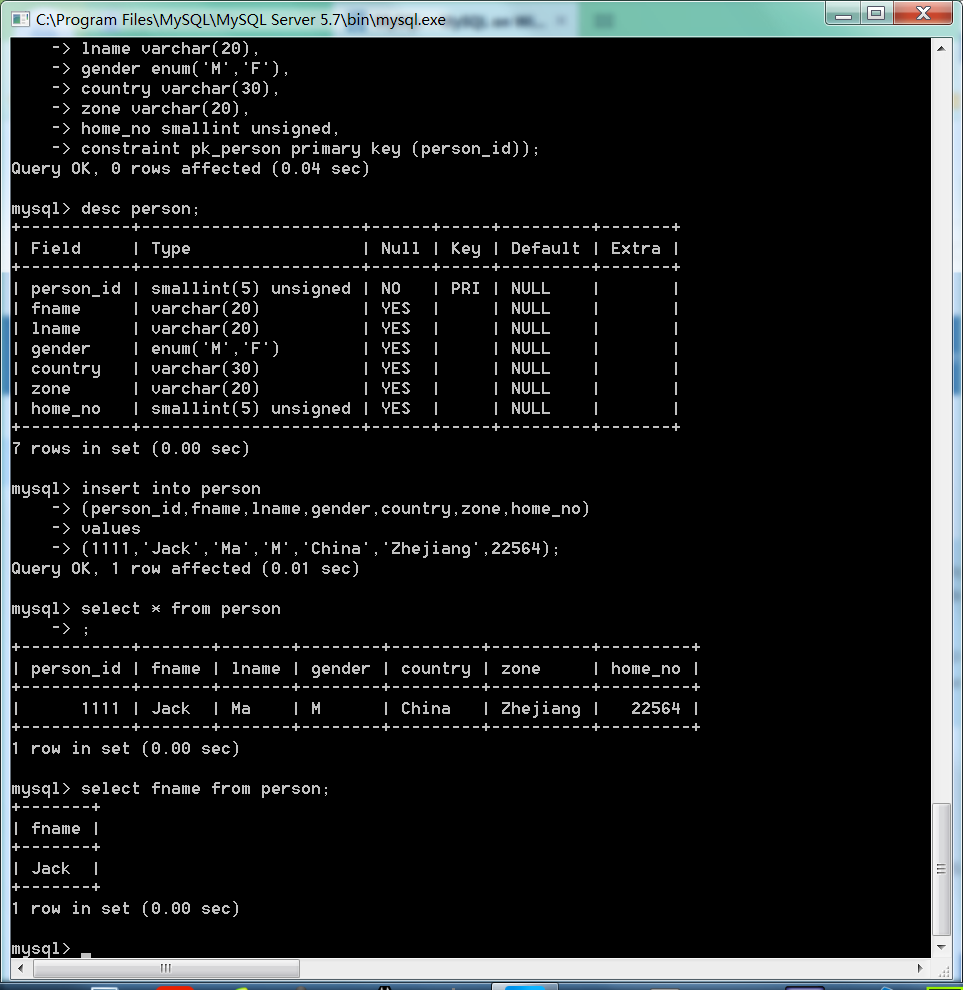
操作和修改，就是增删改查。

例如可以如下图一样插入以及查询数据

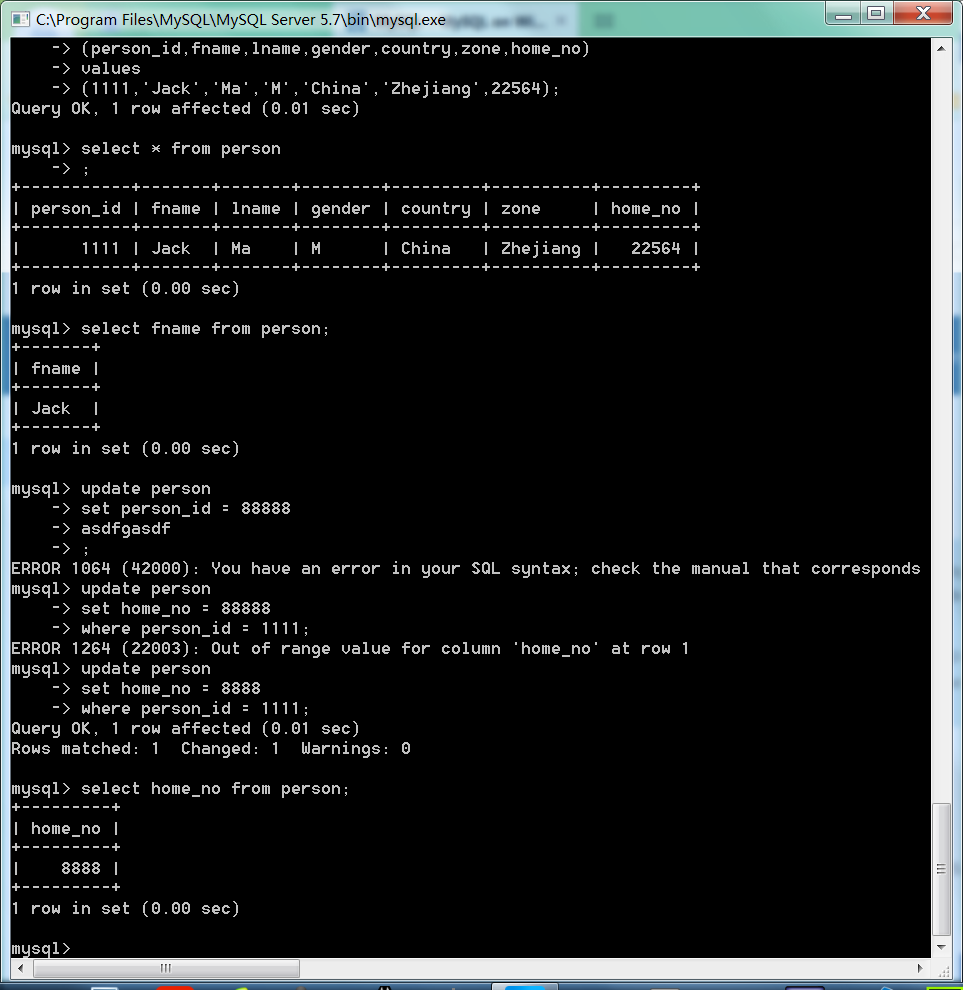
特别地，插入语句时，使用的语法是（列名，列名…）values（数据，数据…）



查询



更新



1. 查询
2. 查询机制

查询之前，服务器会进行3个检查

用户是否有权限执行语句

用户是否有权限访问数据

语法正确性

如果查询的时候发生某些异常，请按照上面的检查思路自检。

1. 查询语句

Select：确定结果集中应该包含哪些列

From：指明要提取数据的表，以及这些表是如何连接的

Where：过滤不需要的数据（剩下需要的数据）

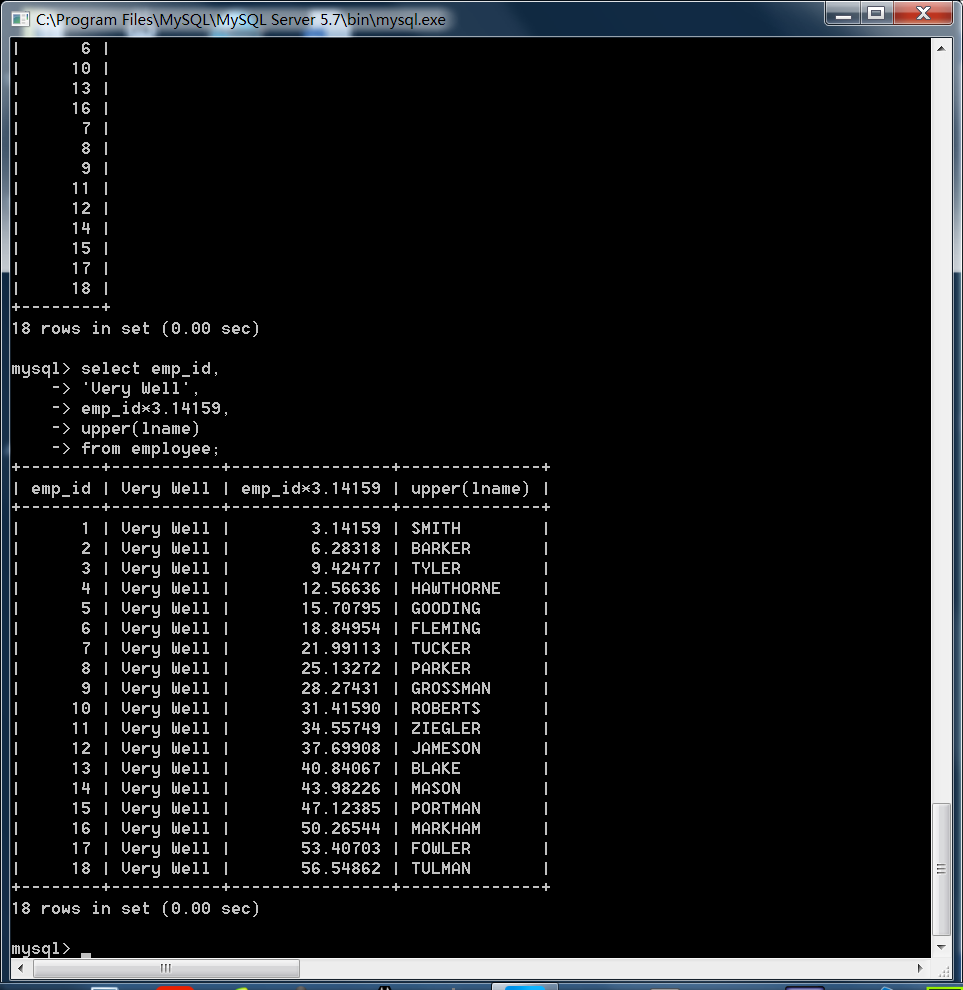
Group by：对具有相同列值的行进行分组

Having：过滤不需要的组（剩下需要的组）

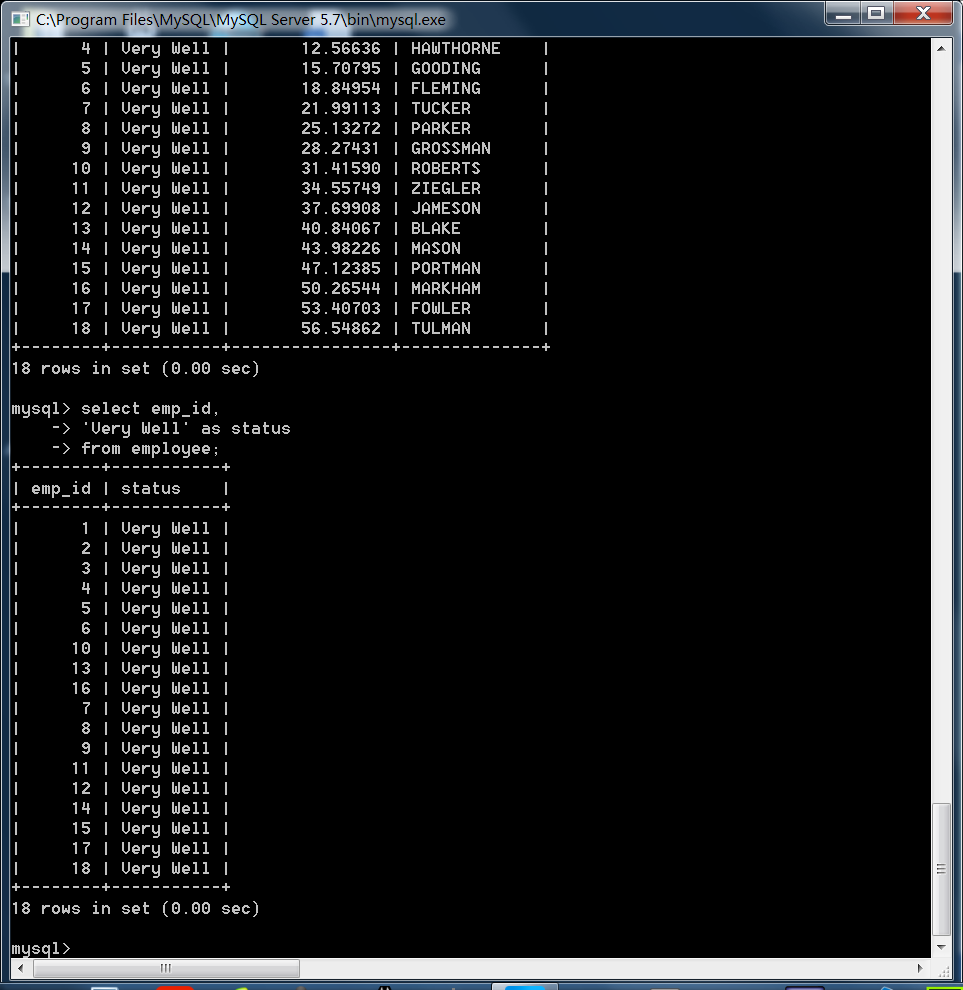
Order by：按一个或者多个列，对最后结果进行排序

1. select

select语句是被最后评估和执行的。同时，使用select不仅仅能选择存在的列，也能“产生”需要的列，例如下图



如果有需要，也可以将列名改为自己想要的，如下图



特别地，在select后面加上distinct关键字，可以去除该列的重复项。（但是请尽量少用，distinct关键字使用时，服务器会先对数据排序，加重服务器负担。）

1. from

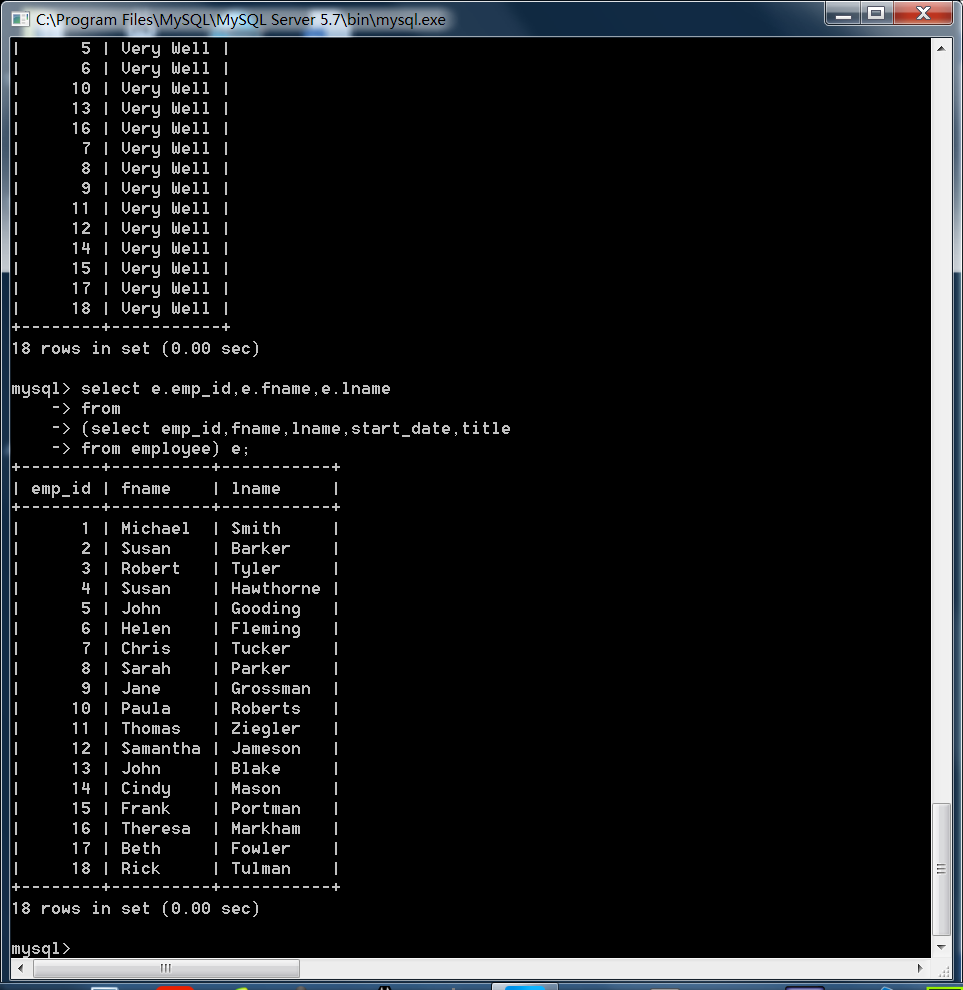
from中对表的定义很宽泛，可以是

使用create table创建的永久表

子查询所返回的临时表

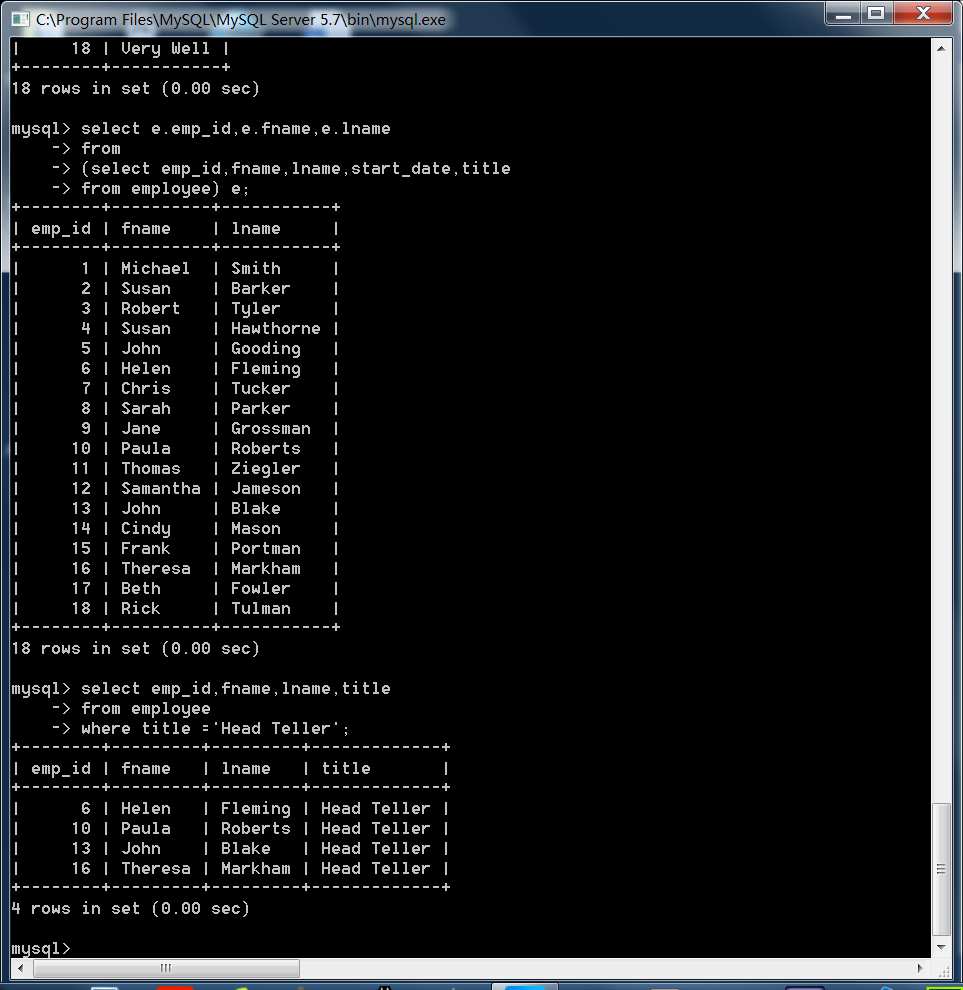
使用create view子句所创建的视图

其中，如果你觉得表的名字太麻烦或者需要用特定的标识来区分，可以给表起别名（和列的情况一样）。如下图



1. where

where子句用来过滤剩下需要的行，例如where title = ‘Head Teller’，则是结果中只有符合Head Teller的数据，如图

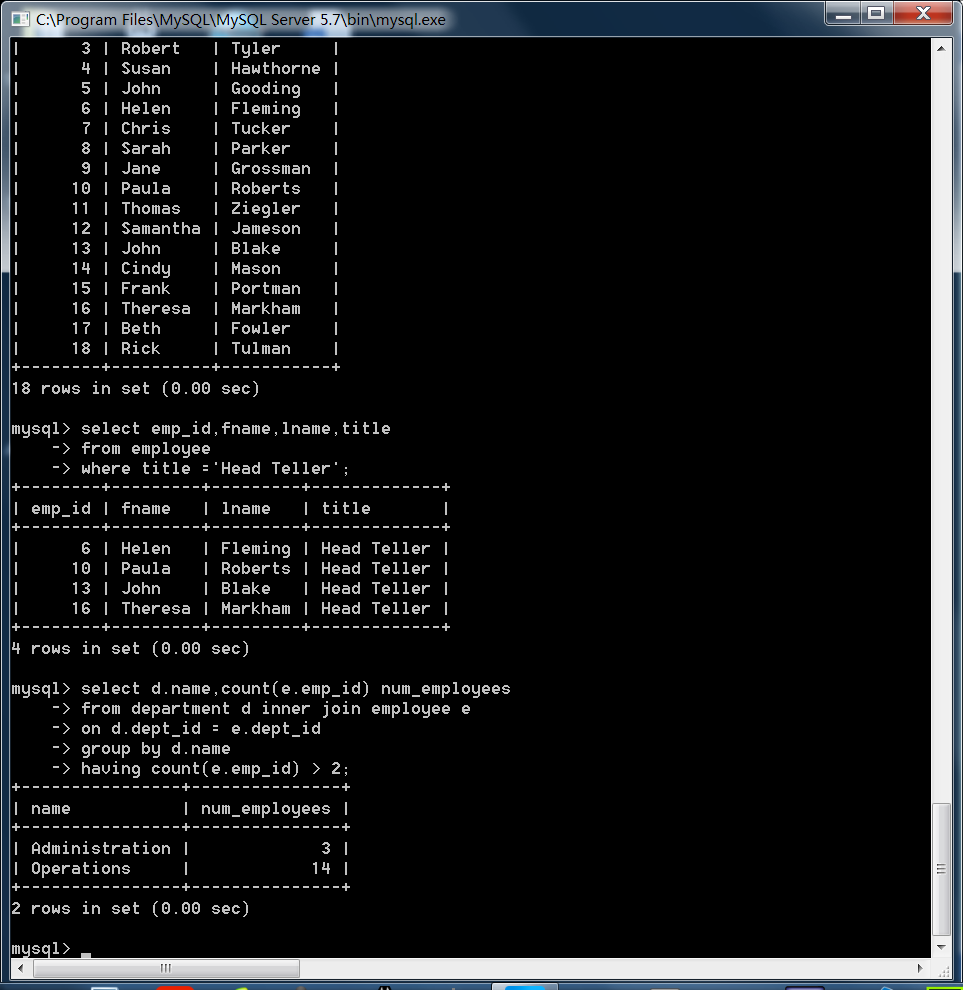


Where中条件如果有多个，则可用and或者or来连接。例如

Where title = ‘Head Teller’ and emp\_id > 5，可用（）来分别表示。

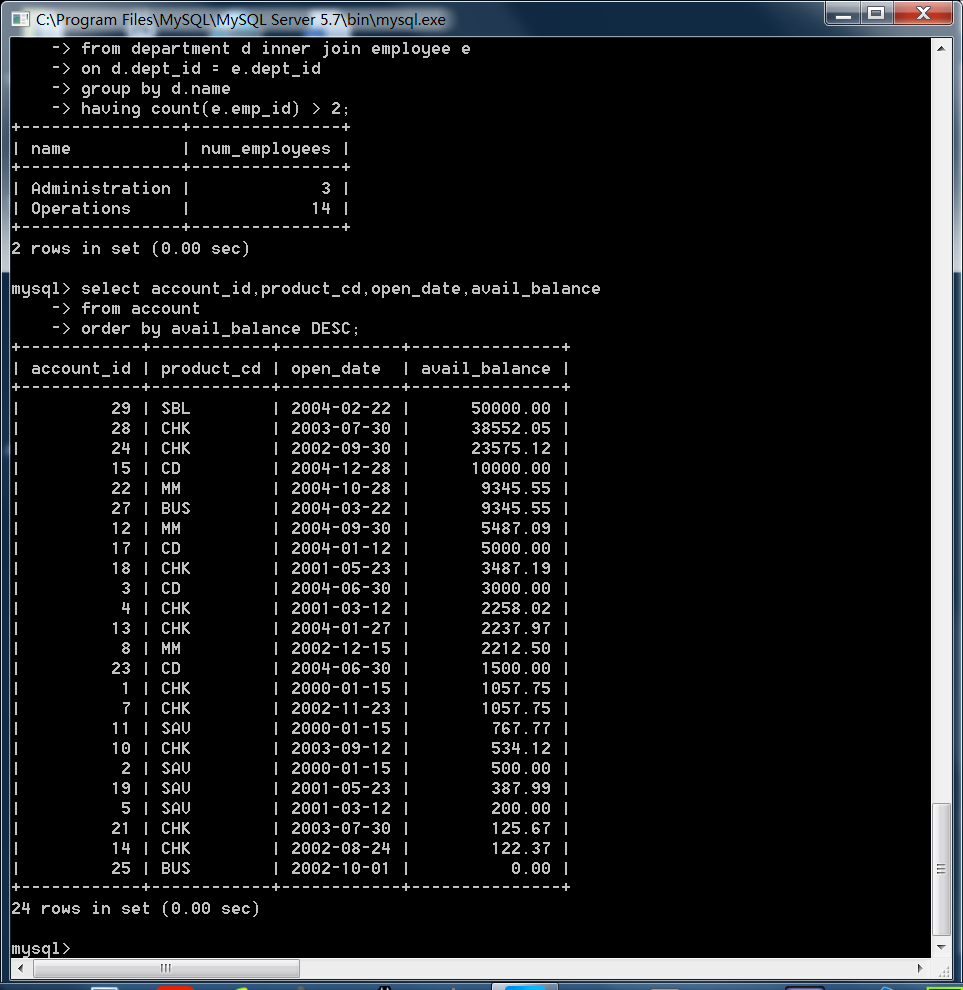
1. group by和having

实际使用效果如图。



1. order by

order by语句排序方式多种多样，可以升序排序（关键词asc 默认），降序排序（关键字desc），根据表达式排序，根据数字占位符排序（即第几列）等等。下面举一个例子（根据avail\_balance降序排序）



1. 过滤
2. 条件评估，以及条件构建

正如上面where语句，多个条件过滤时，可以使用and，or，not，（），等等操作符，同时，条件可以是

数字

表或者视图中的列

字符串

内建函数

子查询

表达式列表

比较操作符

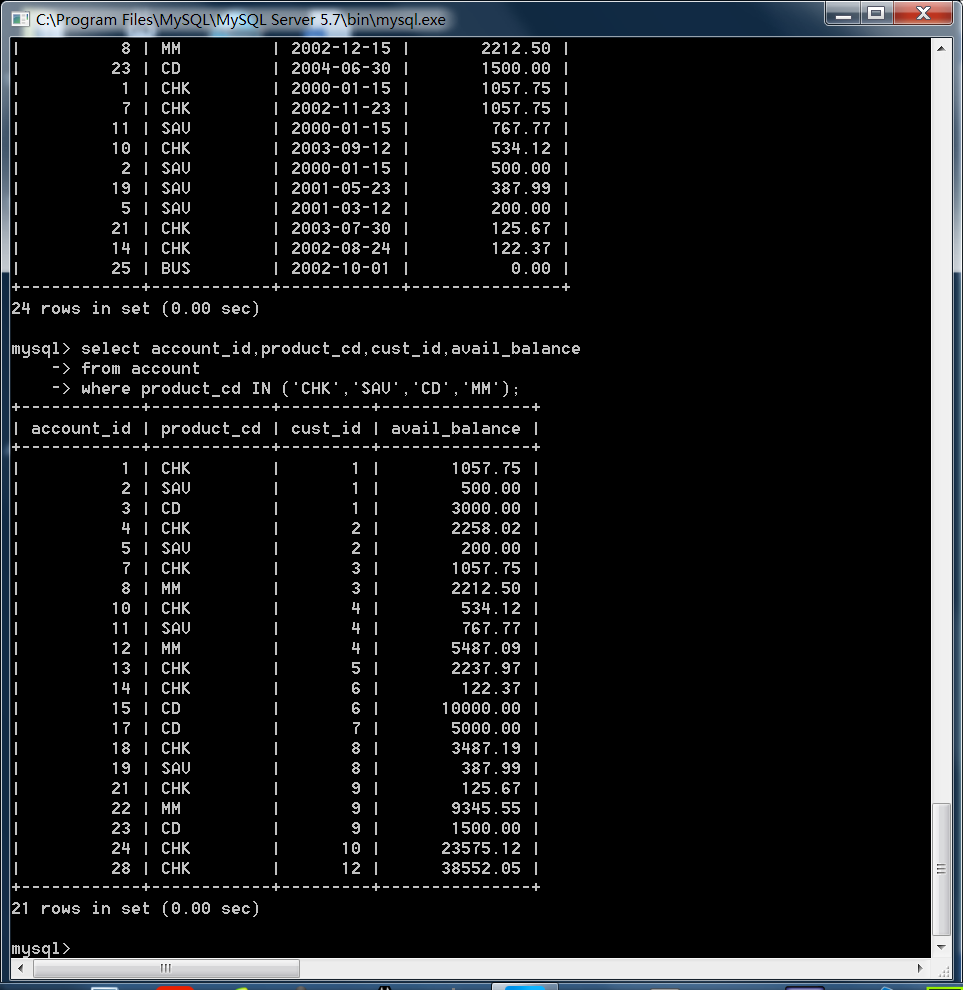
算术操作符

1. 条件类型

条件类型可以是相等条件，符号为“=”，可以是不等条件“<>”。

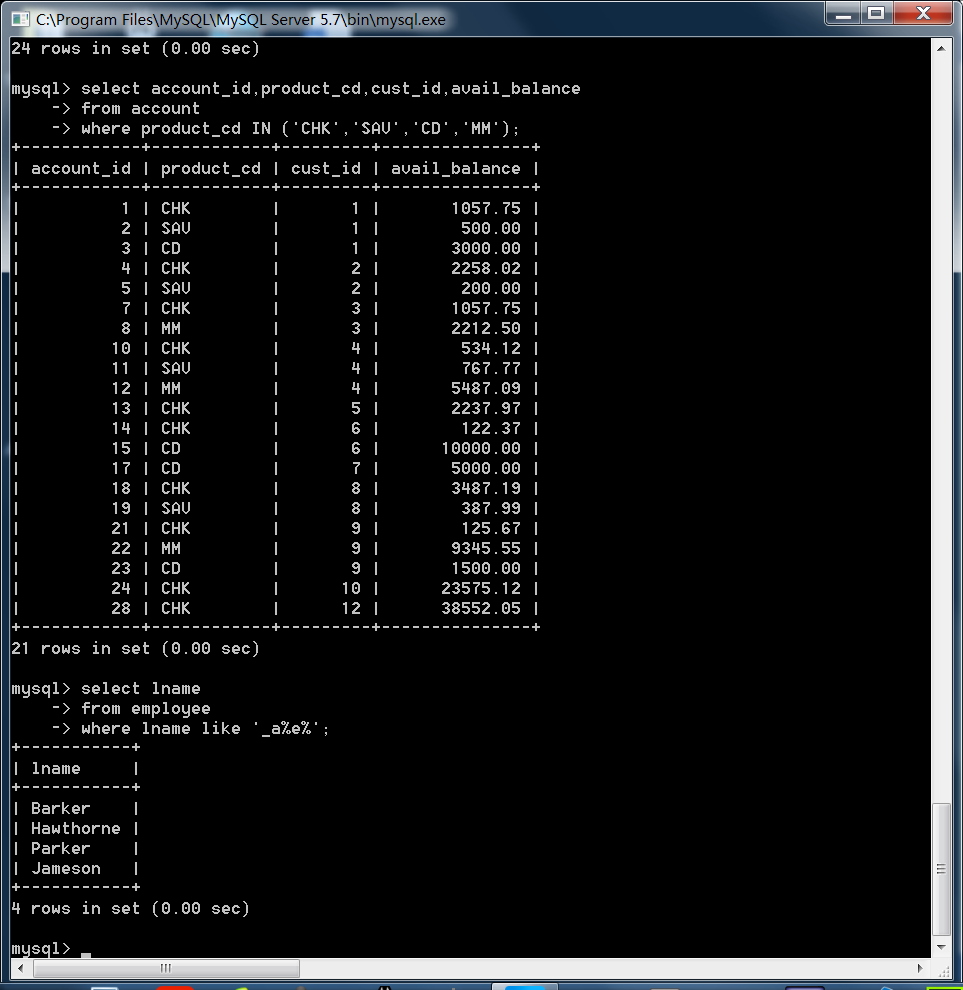
范围条件：例如emp\_id>2，between操作符（必须说明的是，使用时应该是between 下限 and 上限，而且上下限是闭合的），字符串范围。

成员条件：使用in关键字。例如下图



当然，也可以使用 not in 关键字……同时，where里可以嵌套子查询。

匹配条件：使用关键字“like”，字符串，字符串集合，字符匹配，同时，可以使用通配符（“\_”代表一个字符，“%”代表任意数目的字符，包括0）以及正则表达式。例如下图



1. null关键字

表示一个随机的值，null的值不确定，可以用XX is not null来判断是否被赋值。

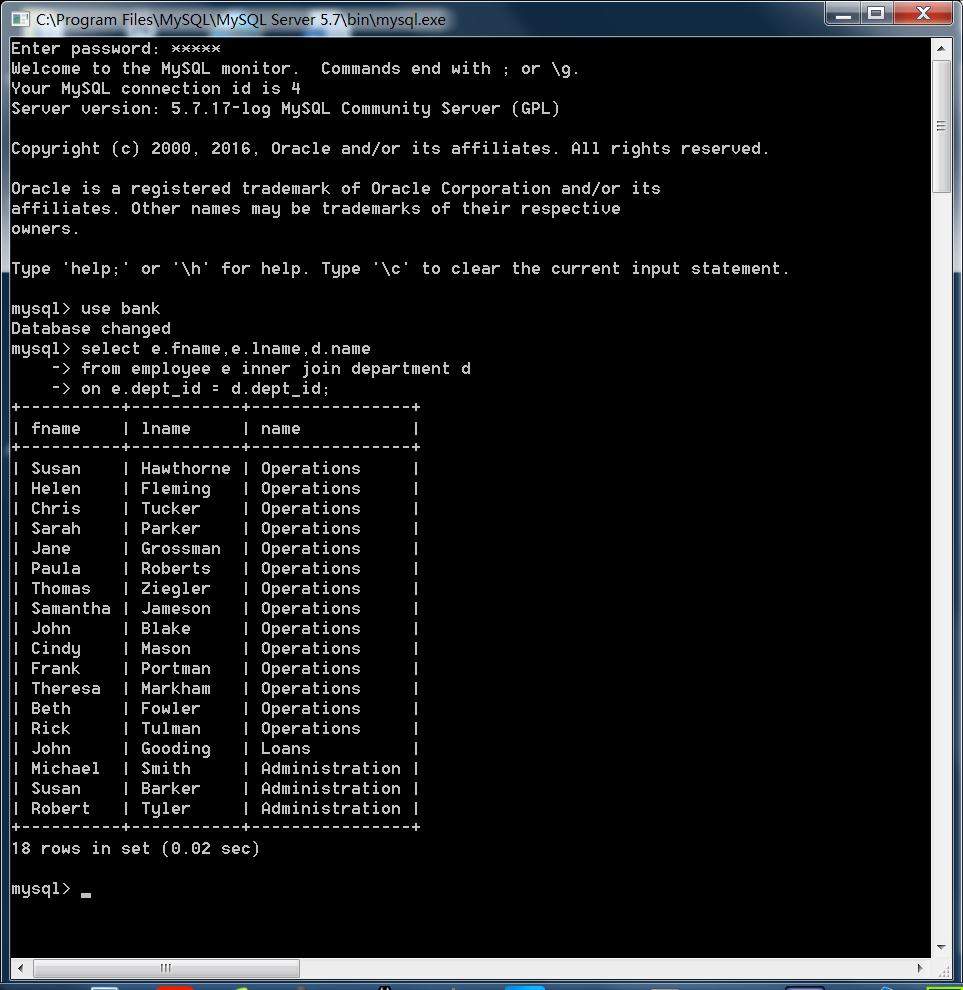
1. 多表查询
2. 什么是连接

有时候，需要在一个表或者视图中显示实际上分别为两个表的项目的内容，就需要连接，而连接的依据（桥梁）就是“冗余”，各个表之间关联的信息。

连接的方式有几种

笛卡尔积：相当于矩阵相乘，会产生“重复项”，关键字join。

内连接：关键字inner join xx on yy，不会产生“重复项”，其中，on可以用using关键字代替，只不过很少人用。例如



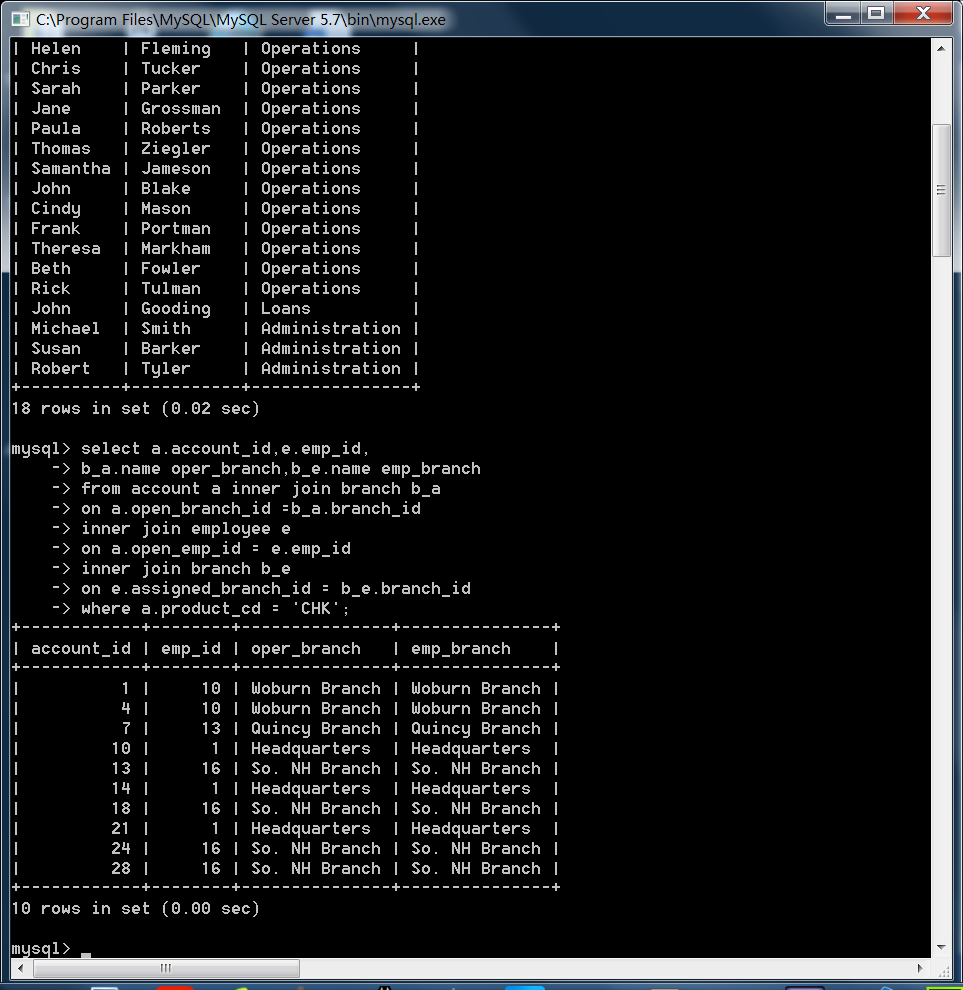
Inner join … on语法符合SQL92标准，是现在主流数据库都可以通用的语法，当然，也是可以使用一些旧版语法，但是与他们相比，inner join … on语法更加清晰和易懂。

1. 连接3个或者更多的表

同等级的多表连接：其实就是多次使用连接语句，而且连接的顺序并不影响结果，因为SQL会自动选择最佳的执行顺序。

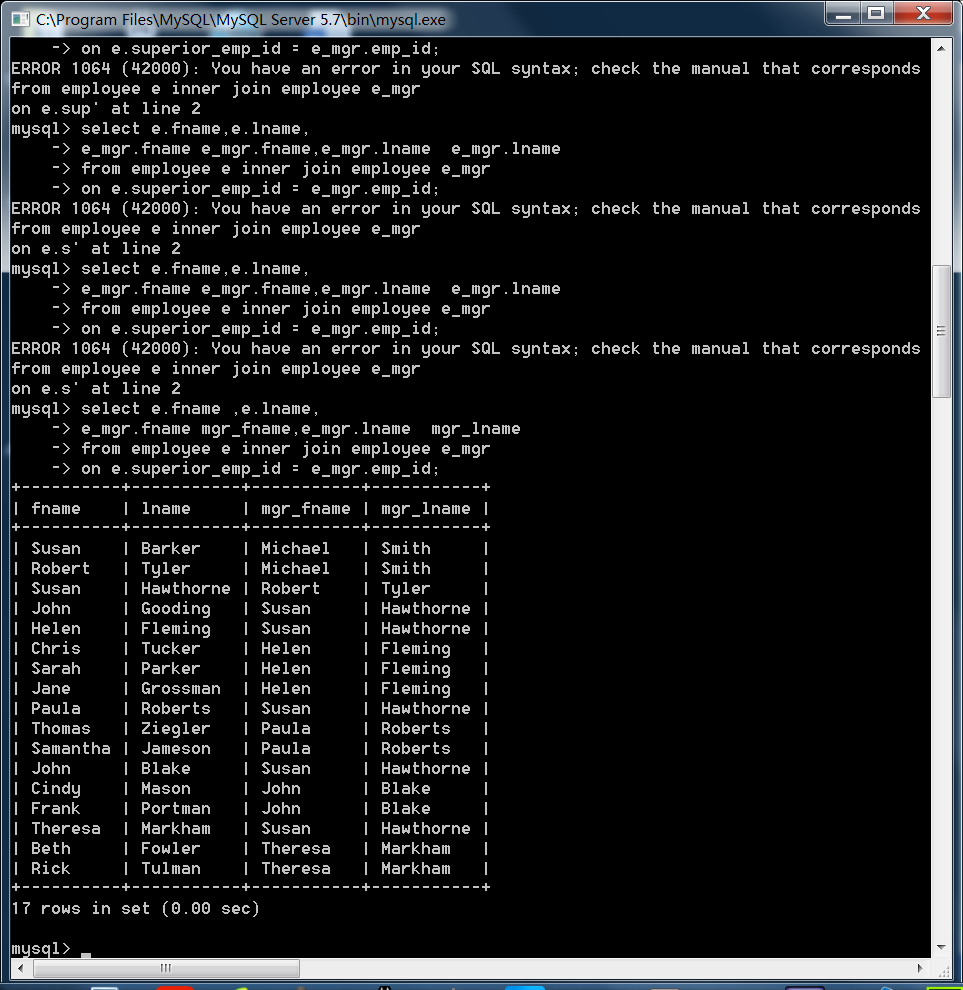
将子查询结果作为查询表：没啥好说的，就是两个表连接之后出来的一个表，再与另外的表连接……

连续使用同一个表：这种情况之下，需要为这个表定义不同的名称，以免混淆。例如



1. 自连接

自己连接自己，某些特殊用途。例如下图，在employee表包含一个指向自身的外键，主管id（superior\_emp\_id），就可以利用它来列出每个雇员的姓名同时还有主管的姓名。

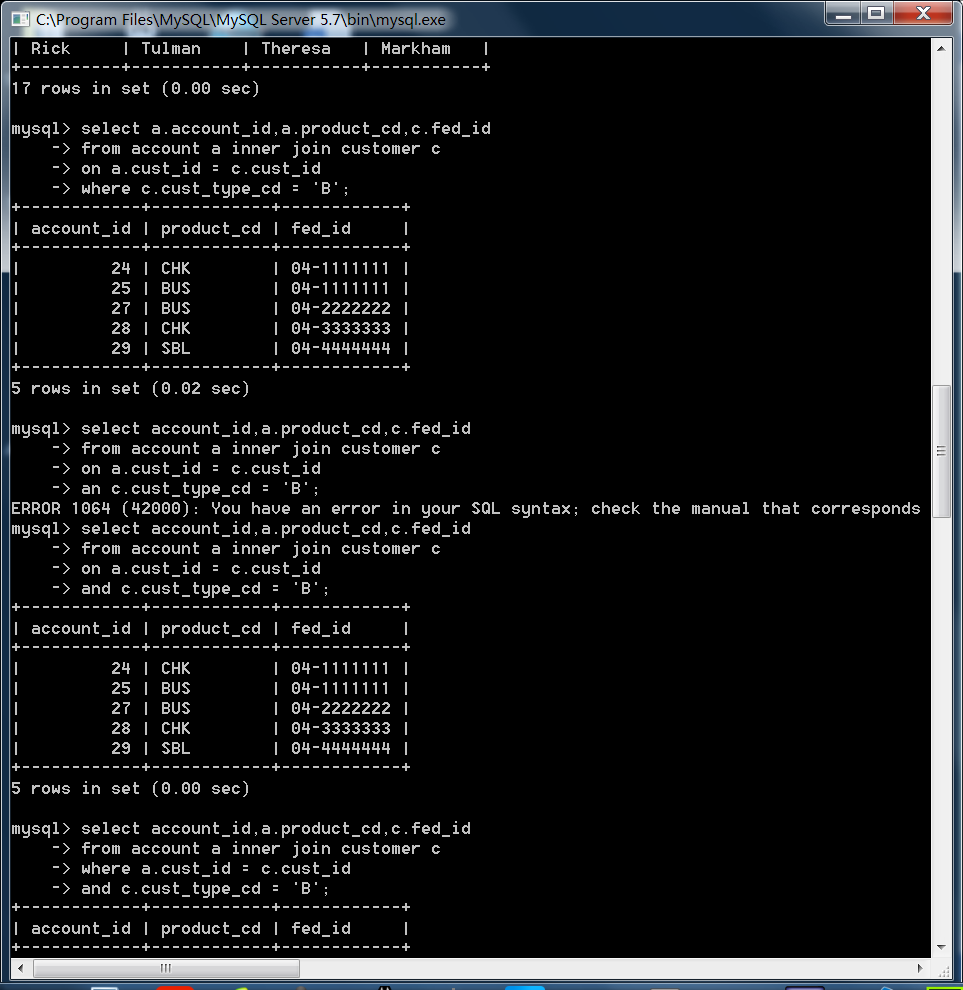


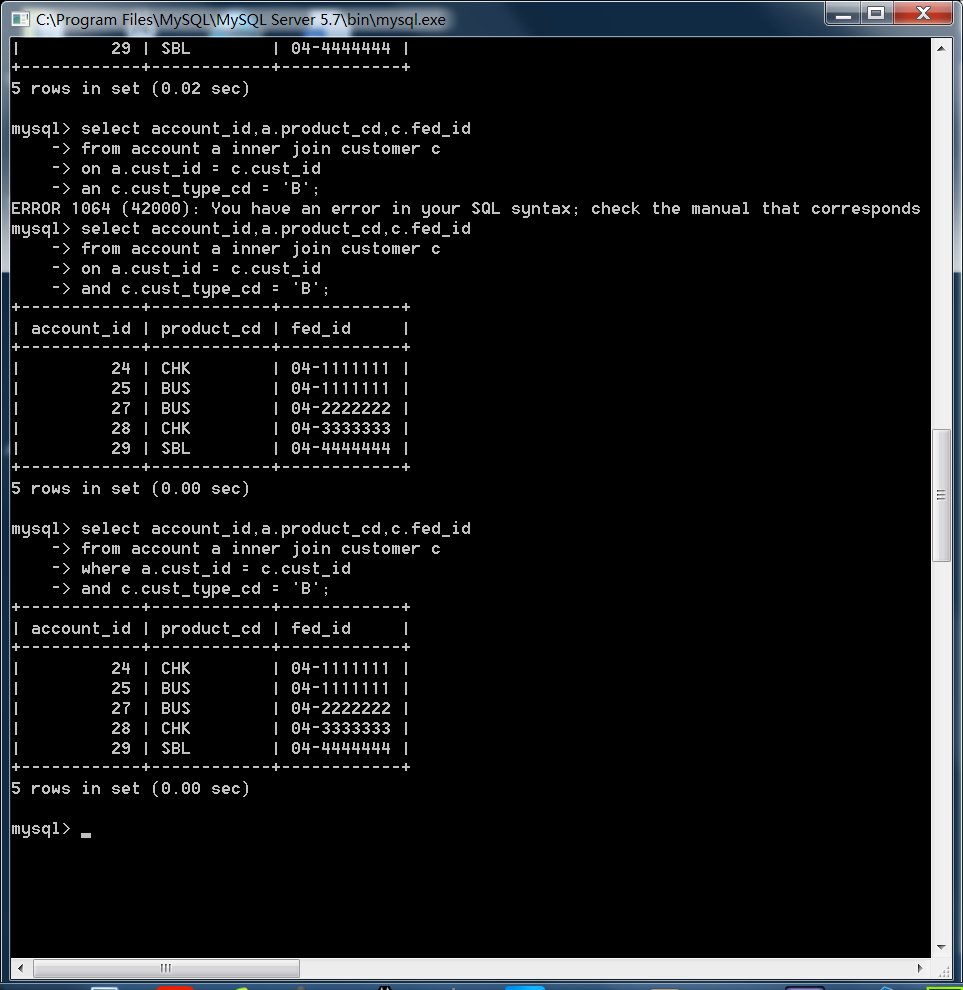
1. 相等连接和不等连接

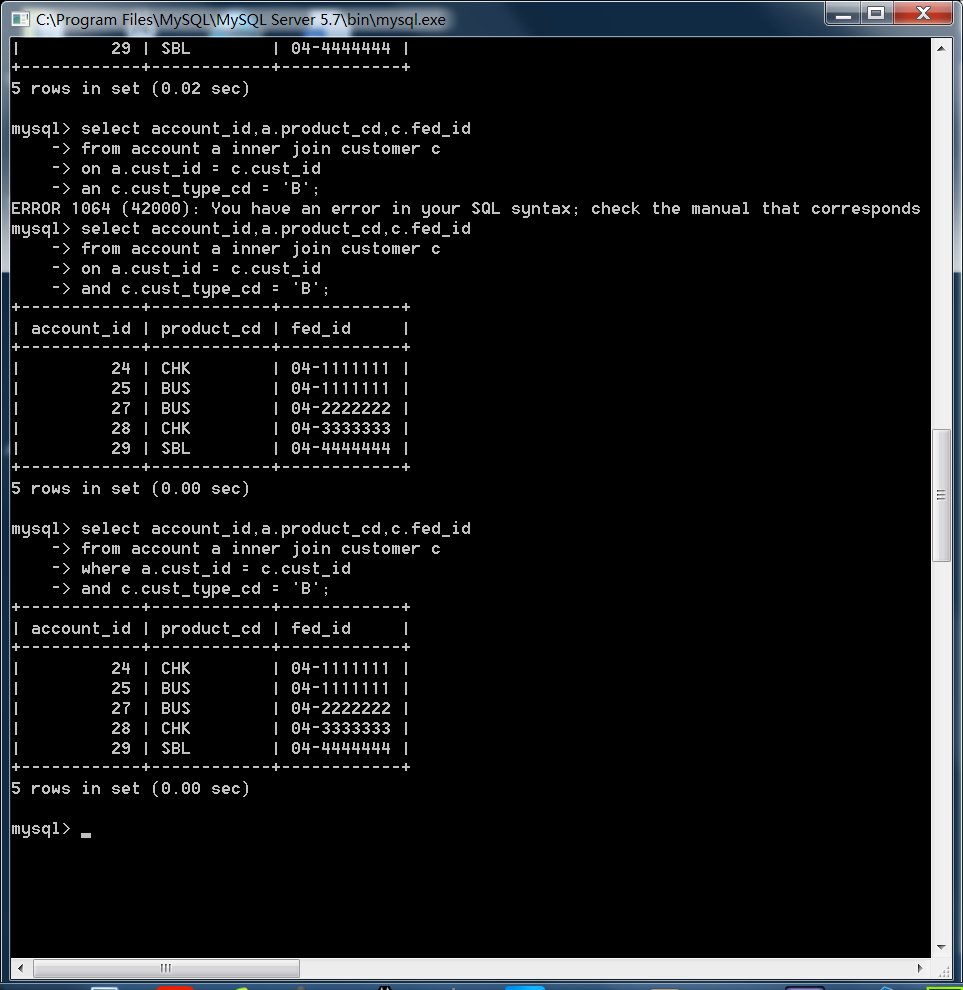
就是在连接的时候，on子句中，使用>= ,<=,=,>,<等等符号作为条件。

1. 连接条件和过滤条件

连接条件的关键字是on，过滤条件是where，由于SQL中，语句的位置是很灵活的（因为服务器会自己选择最佳的执行路线），有时候两者位置倒转，结果也一样，但是，为了逻辑清晰和提高可读性，最好还是按照SQL92标准来写。例如下图







1. 使用集合
2. 集合论

集合……初中生基础咯……交集，并集和差集。

1. 集合操作符

交集：intersect，在Oracle和SQL server里面可以使用。

差集：except，在Oracle数据库中，可以使用非ANSI标准的minus操作符代替，MySQL暂时不支持。

并集：union

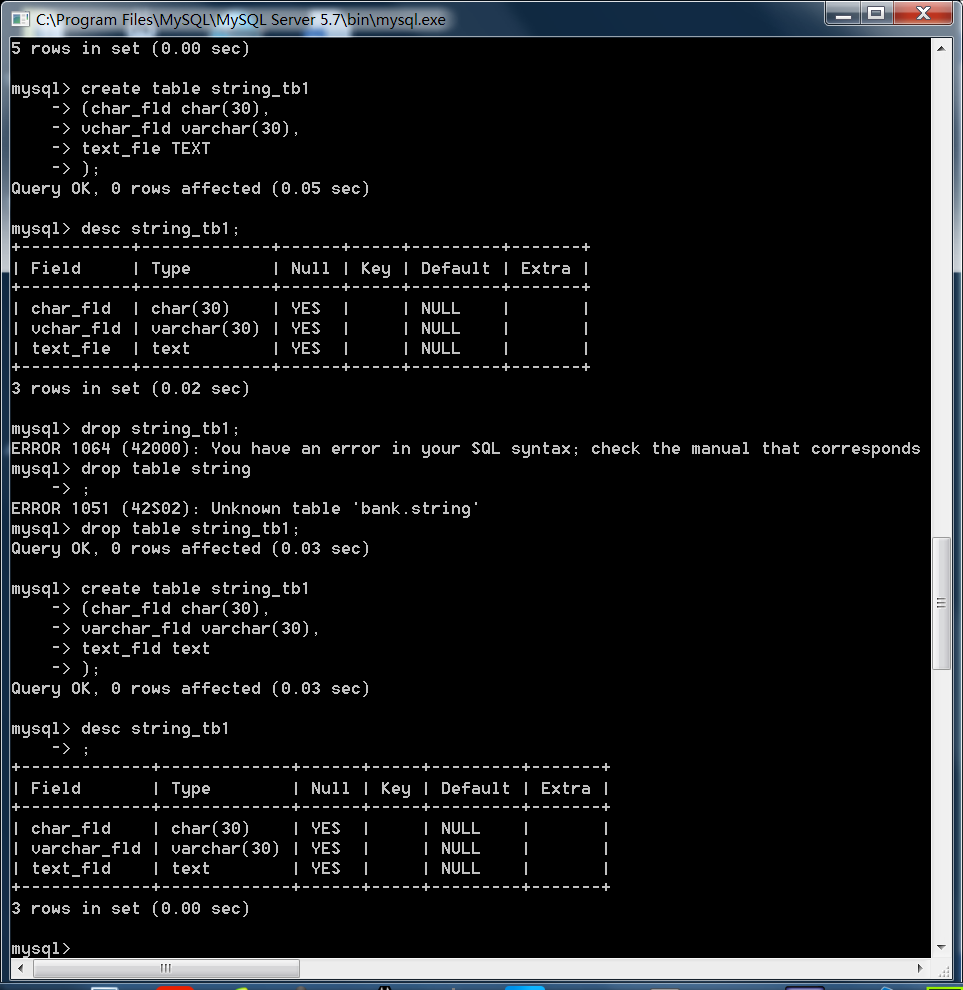
其中，MySQL只支持union（暂时），操作符为union all，并不删除重复项。

1. 集合规则

初中内容咯……两个以上集合需要指定操作符优先级，用（）来指定，当然，现在MySQL只能用union all，不指定优先级似乎不会有问题。

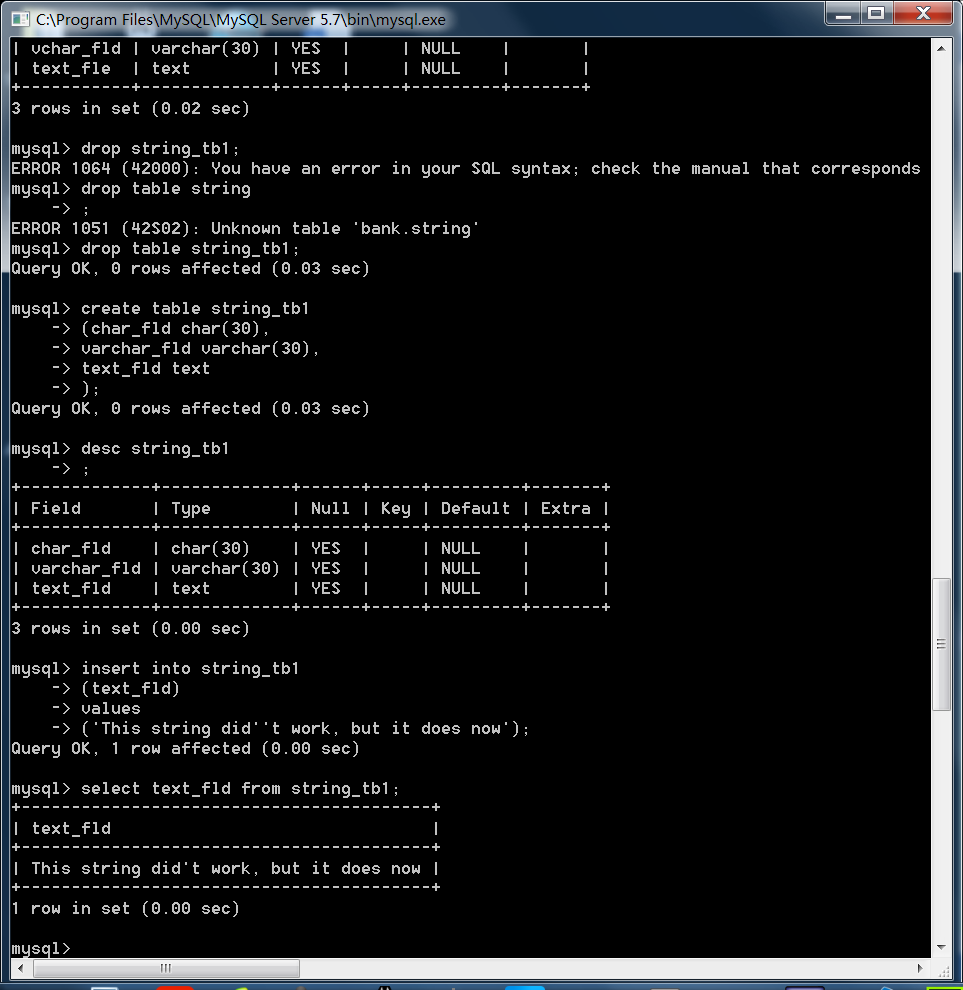
1. 数据生成、转换和操作
2. 使用字符串数据

本次用作演示的表，先按照下图来创建



当然可以使用常规的insert，update等命令来进行数据的生成和修改，但是需要注意的是，当加入的字符串长度大于一开始设置的长度时，MySQL会报错并且给你发一张好人卡。

另外，当字符串里面含有单引号的时候，需要在其前面再加一个单引号作为转义字符，例如下图



同时，如果要加入特殊字符，就要视乎当时使用的字符集来用char（number）来调用。

操作字符串的内置函数有

Length（xx）：返回xx的长度。

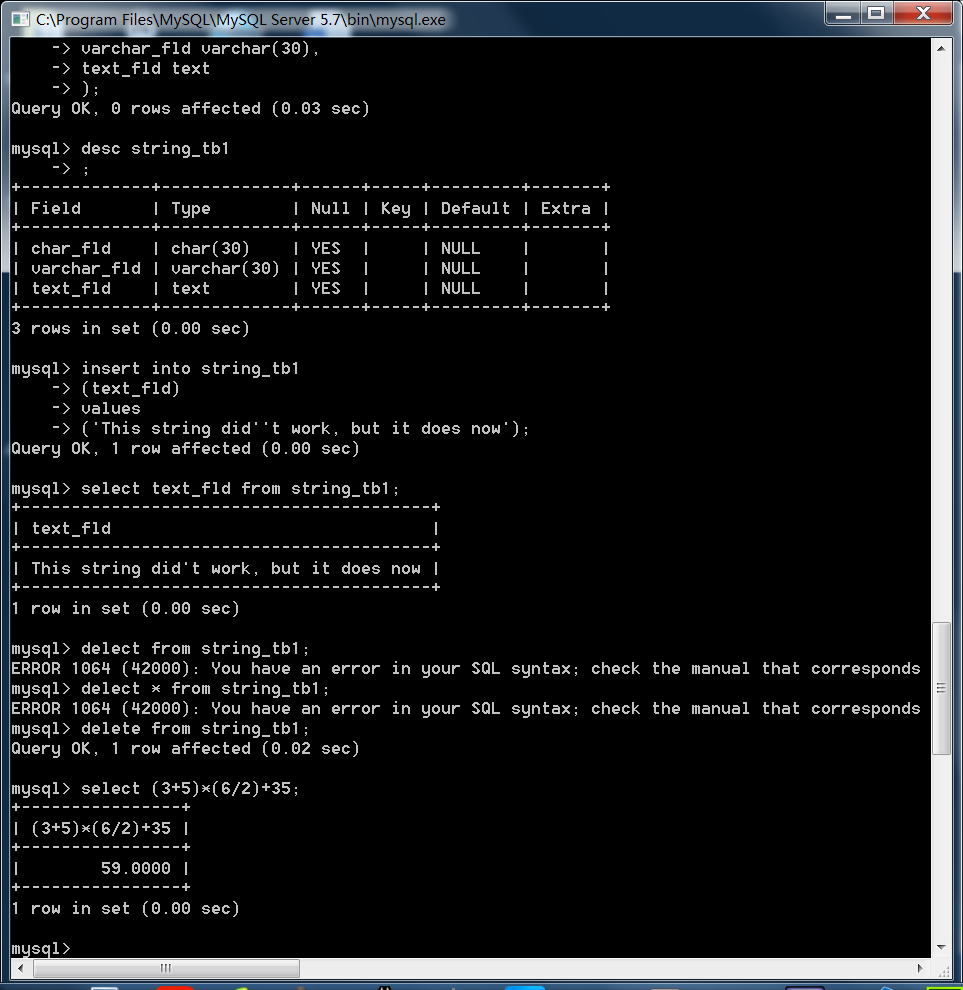
Locate（‘A‘，xx）：返回xx中第一个A的位置。

Strcmp（‘aa‘，’bb‘）：对aa和bb进行比较 ，大小写不敏感。

Concat（）：根据字符串片段构建新的字符串。

1. 使用数值数据

可以使用select语句把MySQL当作计算器来用，例如



也可以使用内置的函数来进行一些数学运算，例如select cos（x）等等。

然后有一些控制精度的内置函数

Ceil（）向上取整

Floor（）向下取整

Round（）四舍五入

特别地，Truncate（72.0909,2）=72.09

Sign（）取该数字的符号1,0或者-1

Abs（）取绝对值

1. 使用时间数据

使用时间数据之前，请确认系统本身的时区。

字符串转换到日期：select cast（’2017-7-7 15:20‘ as datetime）🡪2017-07-07 15:20

Select cast（’2017-7-7‘ as date） 🡪2017-07-07

Select cast(‘110:10:20’ as time) 🡪110:10:20

另外，可以使用str\_to\_date（）来转换，例如

Str\_to\_date（’september 17,2017’ , ‘%m %d, %y’） 🡪2017-09-17

关于时间的函数有

Date\_add (xx , interval yy) 🡪在xx的基础上增加yy（可以是天，小时，月等等）

Select last\_day(xx)🡪返回xx日期里面本月最后一天

Select dayname(xx)🡪返回xx日期是周几

Select extract(year/month/day form xx)🡪提取xx中的年/月/日

Select datediff(xx,yy)🡪返回xx和yy之间有多少天

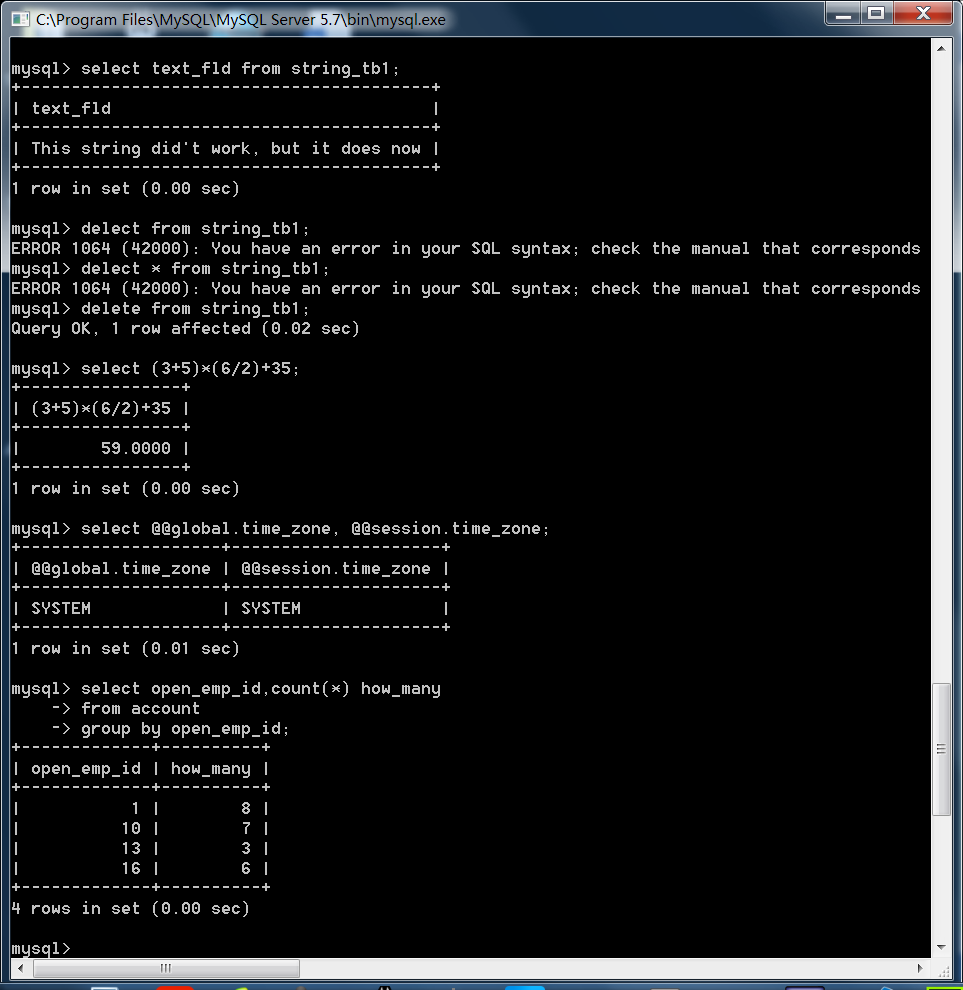
4、转换函数

Select cast（xx as 类型）

1. 分组与聚集
2. 分组的概念

分组是为了使用户在更高的粒度上面与数据进行交互。

以下例子是统计emp\_id的职员每人开的账户数



1. 聚集函数

聚集函数指的是对某个分组的所有行执行的特定操作。

Max（）🡪返回集合中的最大值

Min（）🡪返回集合中的最小值

Avg（）🡪返回集合中的平均值

Sum（）🡪返回集合中所有元素的和

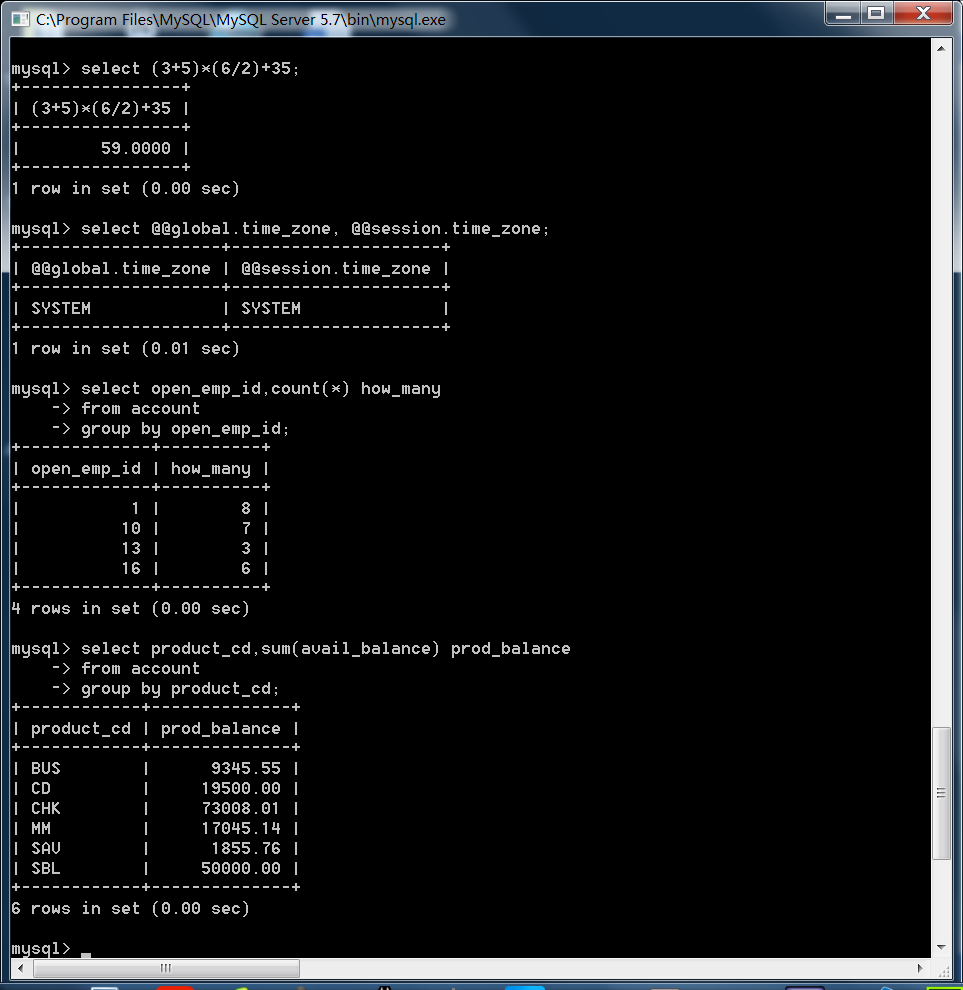
Count（）🡪返回集合值的个数

在使用聚集函数时，（）内也可以说表达式

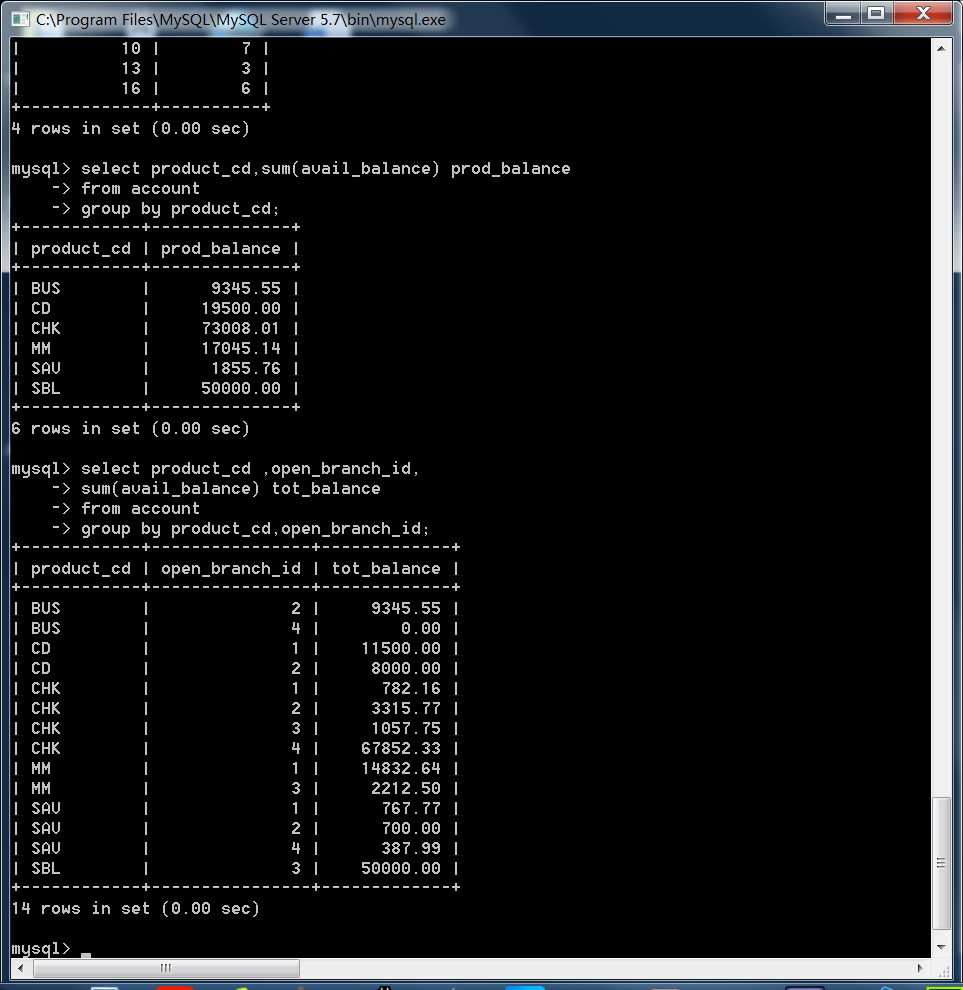
Count（\*）与count（val）的区别在于val会忽略其中的null

1. 产生分组

对单列进行分组，例如

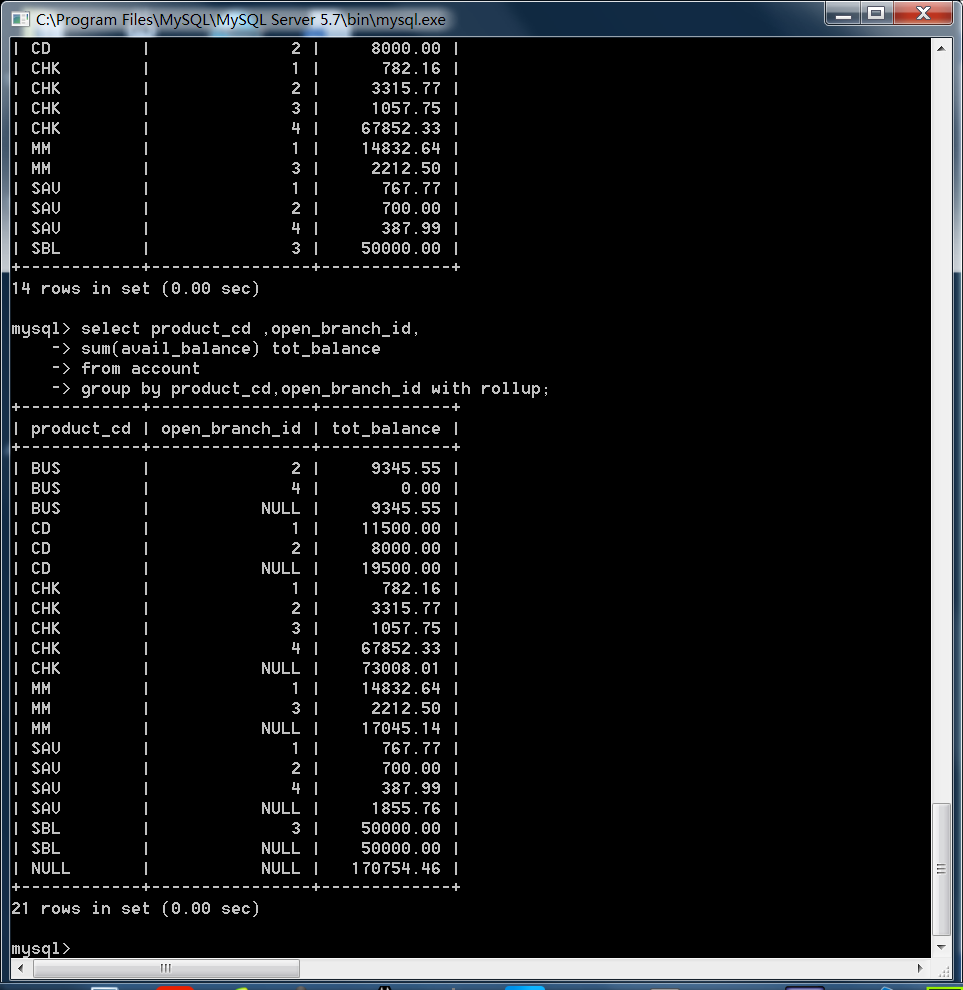


对多列进行分组，例如下图表示每个branch\_id的各个账户的合计balance



当然，分组时， 分组条件也可以是表达式。

在group by语句后面加上with rollup，可以产生合计数例如下图，其中的null行就是每个product的合计数



1. 分组过滤条件

Where语句要在group by语句之前使用，并且where语句不能包含聚集函数。

Having则要在group by语句之后使用，它在分组后才能产生作用。

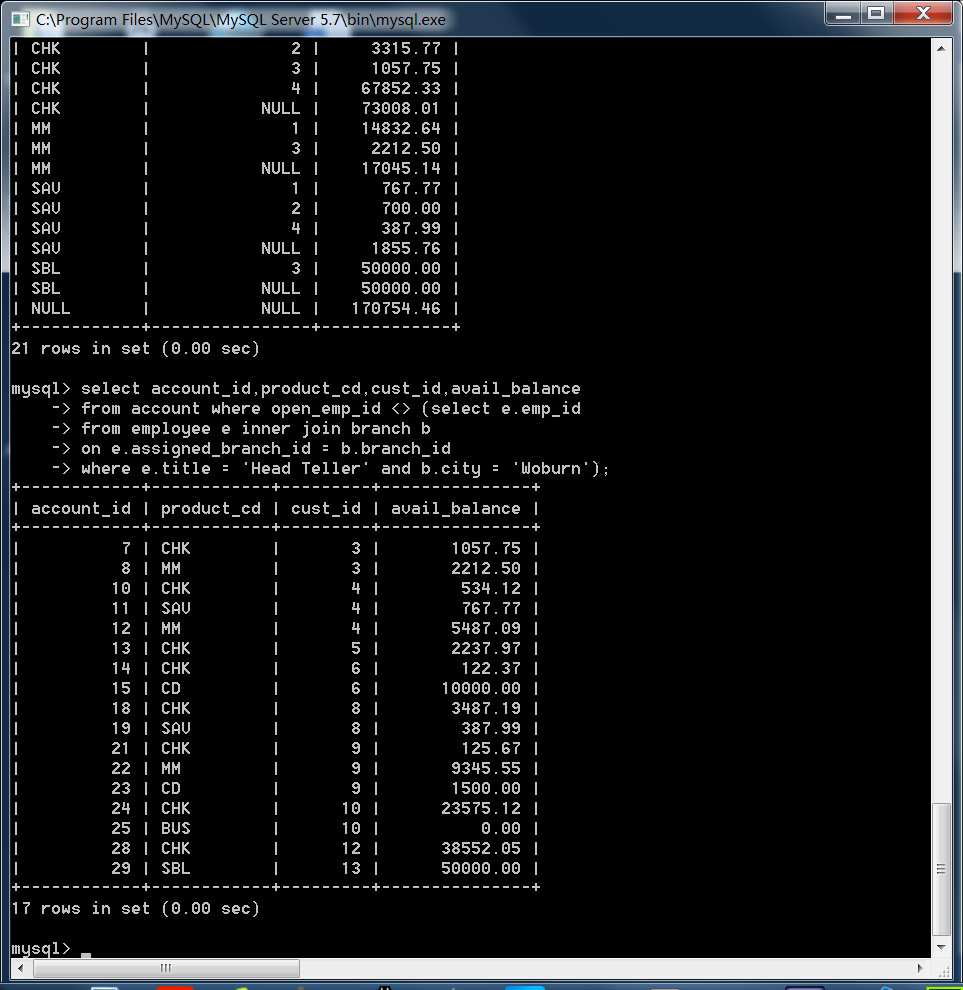
1. 子查询
2. 什么是子查询

子查询是一个SQL语句内的查询。一般返回的结果集可以是：单行单列、单列多行、多列多行。

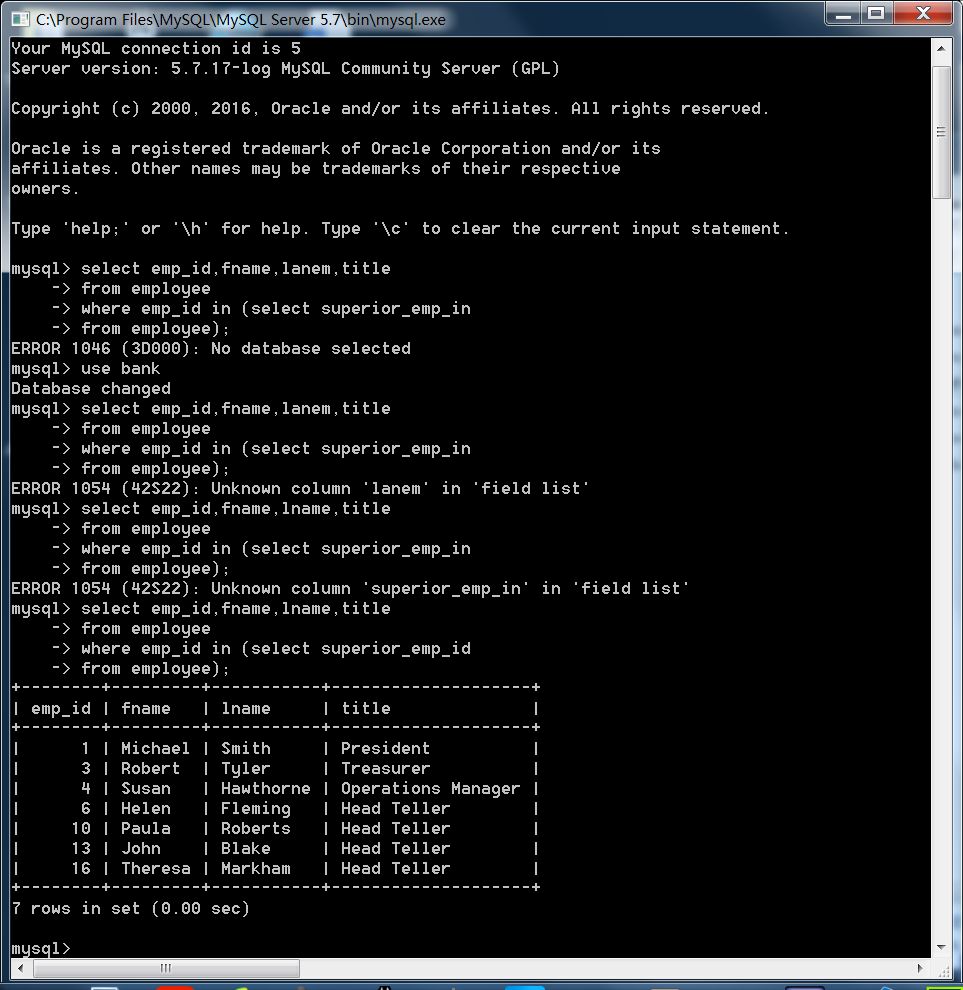
子查询是一个临时表，存在于内存中，用完后会被丢弃。

1. 非关联子查询

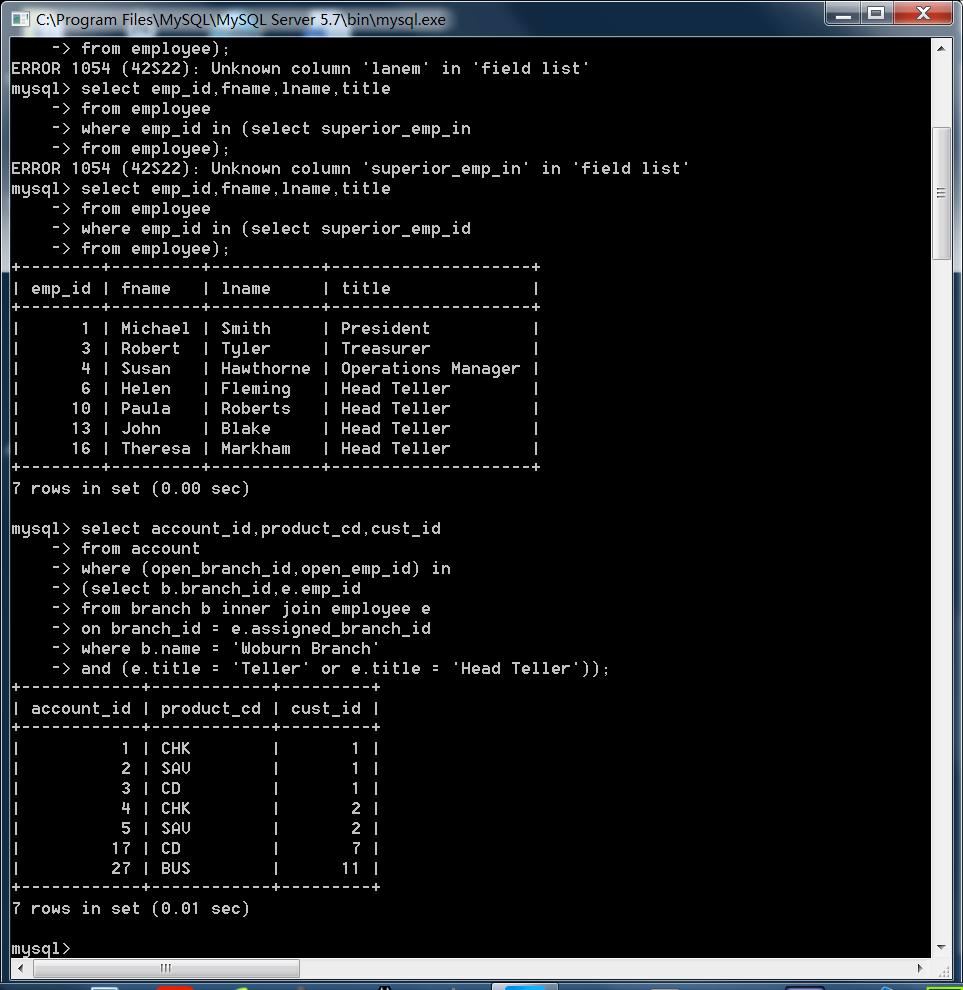
这些子查询是可以完全独立的，不依赖于宿主SQL语句的任何资源。其中，返回单列单行的非关联子查询被称为标量子查询，经常用于等式条件（相等，大于小于，不等）运算符的一边。例如



多行单列子查询，不能在等式条件运算符中，但是可以用在in和not in或者any和<> all运算符中。例如下图

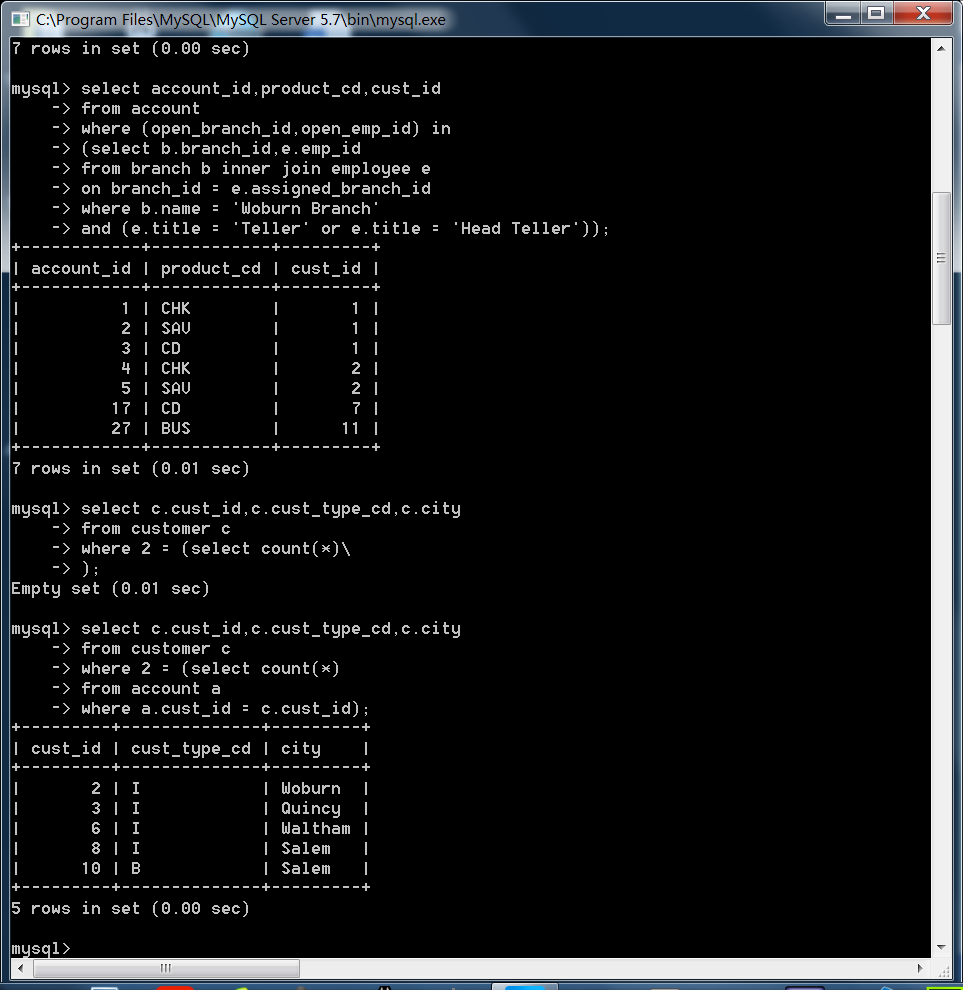
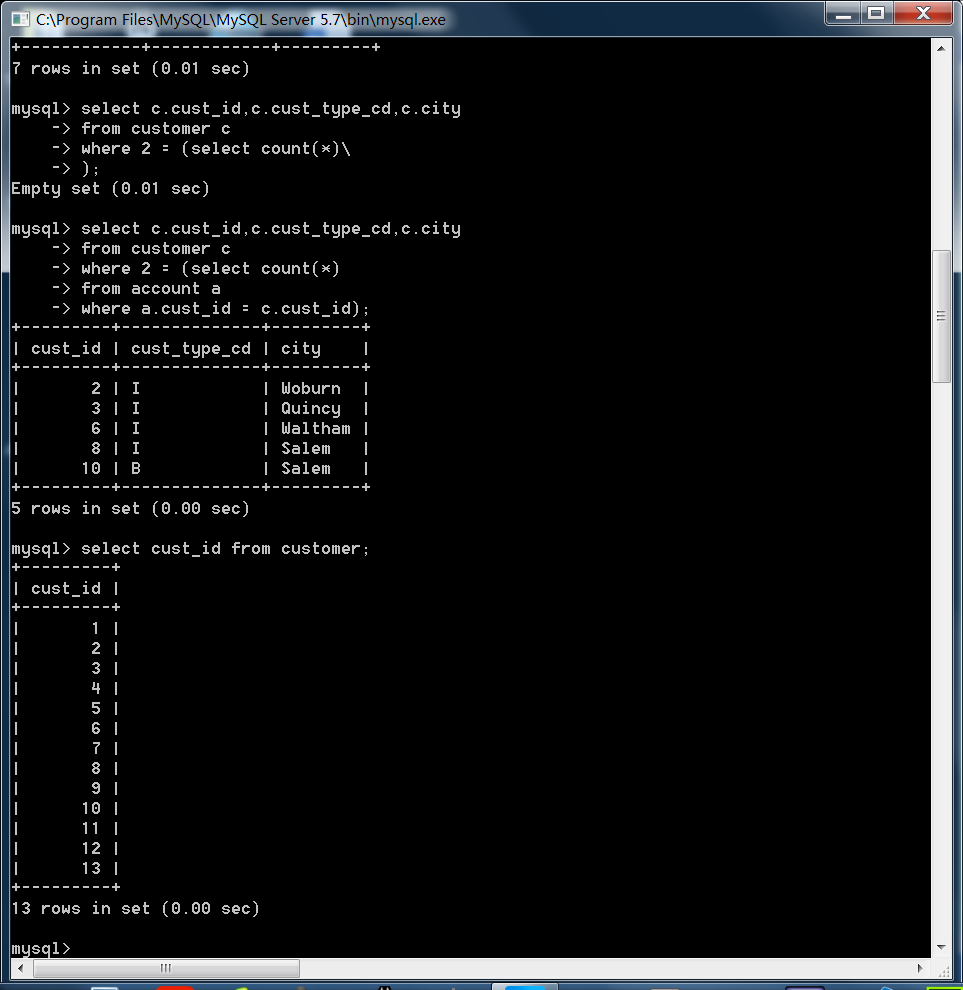


多列子查询，如下图



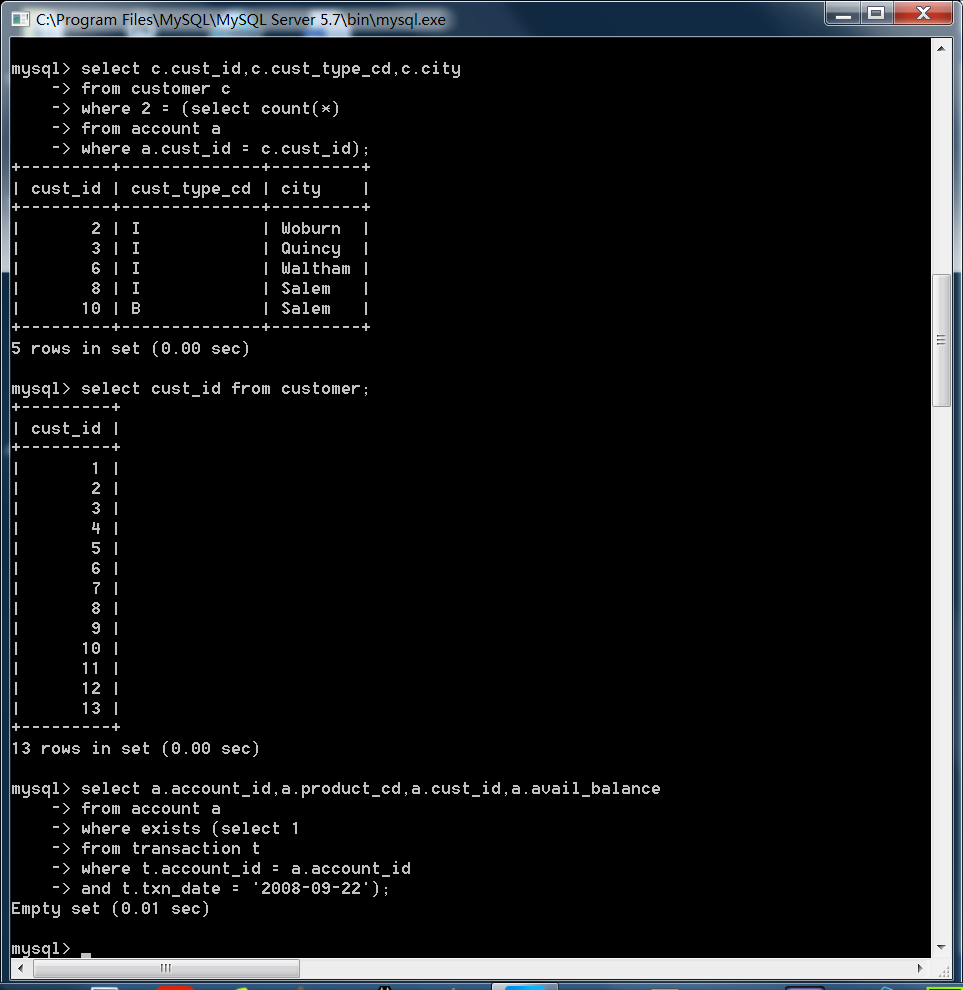
1. 关联子查询

关联子查询不是在语句执行之前一次执行完毕，而是为每一个候选行执行一次。例如



在这个查询里面，子查询会先根据 c.cust\_id 这个条件筛选出13个条目，对每个条目进行select count(\*)子查询。关联子查询还可以用其他条件，例如范围条件等等。

Exists 运算符：是关联子查询中最常用的运算符，它只关心是否存在而不关心存在多少数量。一般用法如下图，子查询中select后面一般是1或者\*，表示至少一行



关联子查询也可以用在update，set，delete中，但是需要注意，在MySQL中delete语句中使用关联子查询时，不能使用表别名。

1. 何时使用子查询
2. 连接—进阶
3. 外连接

如果表a中有一些项在表b中没有对应的，在内连接inner join 的时候，就会被去掉。如果想显示那些项，则需要外连接。

外连接有

Left outer join

Right outer join

并且他们可以同时使用多个。

特别地，自外连接就是一个表连接自己，这个在之前有演示过类似。例如这个语句

Employee e outer join employee g

On ….

这样的连接可以排除null的干扰。

1. 交叉连接

于笛卡尔积类似。

语句 cross join

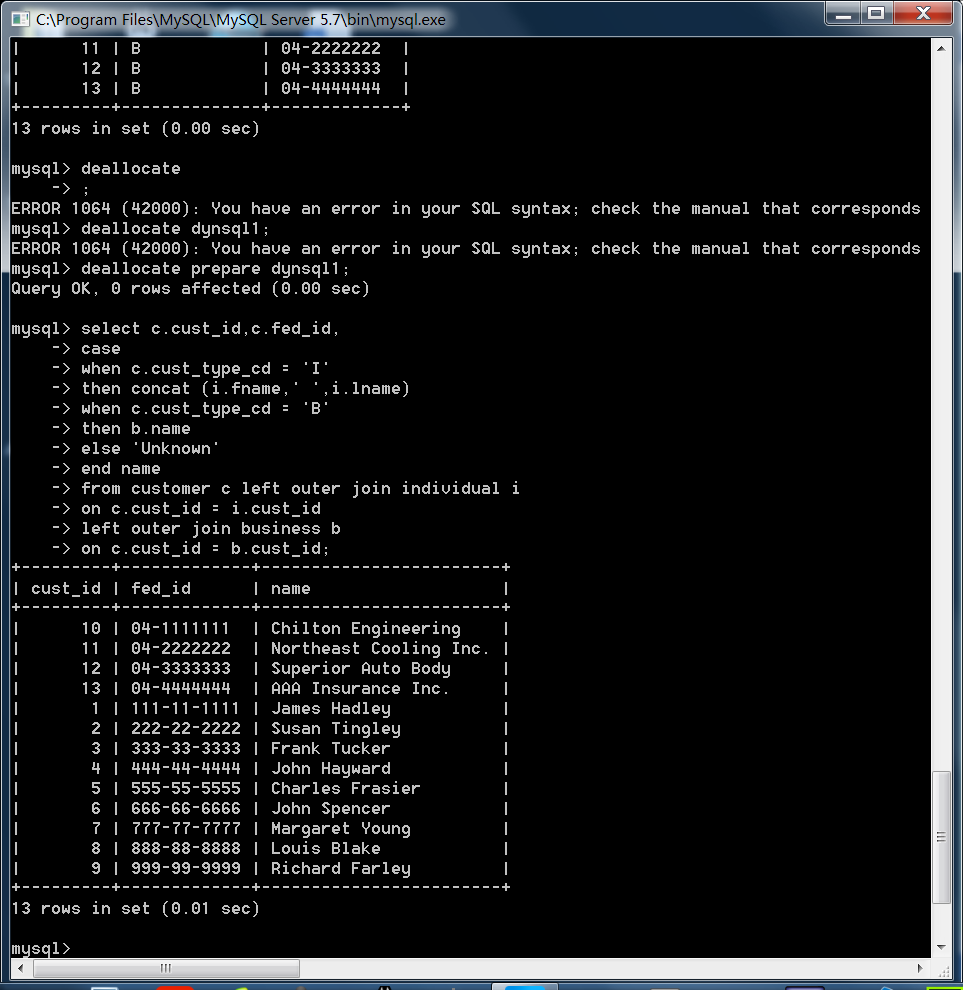
1. 自然连接

由服务器去选择最好的连接方法，少用，耗费资源。语句为nature join。

1. 条件逻辑
2. 条件逻辑

条件逻辑相当于编程语言中的switch。

下图是一个条件逻辑的例子，其中case表达式返回name列的数据，数据是由individual账号的fname+lname和business账号的name组成，选择条件是c.cust\_type\_cd



1. Case表达式

Case

When c1 then e1

When c2 then e2

…

…

[else ed]

End (as xx)

1. 使用case表达式

很简单。

当然，使用case表达式后面可以使用group by等等语句。

1. 事务
2. 多用户数据库

数据库并不总是只有一个用户在使用，如果多个用户同时使用数据库，则有可能他们同时访问或者修改同一个条目，这时候，就需要事务机制来保证数据的合法和安全性。

保证数据库的可用性一般都是通过锁这一技术。分别有读锁和写锁。锁的粒度有表锁，页锁和行锁，MySQL三者都有，取决于你使用什么样的数据库引擎。

1. 什么是事务

错误并不能完全避免，例如停电等等物理性质的，人为操作等等主观性质的错误，要保证数据的合理和规范，就需要事务。

事务有原子性：要么一个都不执行，要么全部执行以及持久性。

MySQL的启动事务机制是：单个SQL语句会被独立于其他语句自动提交，启动一个事务之前需要先提交一个命令。

在结束事务是，为了持久化数据变化，必须显式地借宿事务，使用commit命令。如果需要撤销，则使用rollback。

为了工作的方便，有时候需要设置一些保存点，使用savepoint my\_savepoint语句。然后可以使用rollback to my\_savepoint语句来回滚到保存点。

1. 索引和约束
2. 索引

使用索引，可以更加高效地查询和修改。

其语法是

Alter table xx

Add index aaaa （name）；

可以使用show命令查看所有的索引。

添加唯一索引，可以让服务器自动检索该语句是不是已经存在于本表。语法是

Alter table xx

Add unique aa （name）

也可以创建多列索引

索引的类型有：B数索引和位图索引。

每个索引其实是一个表，请适当使用（仅当出现清晰的需求的时候才添加索引）。

1. 约束

主键约束：标志一列或者多列，并保证其值在表内的唯一性。

外键约束：一列或者多列的值必须在另一个表中存在，级联更新的时候使用。

唯一约束：限制一列或者多列的值，保证其在表内的唯一性。

创建约束的语法如（创建外键约束）

Constraint fk\_product\_type\_cd foreign key (product\_type\_cd)

References product\_type (product\_type\_cd);

删除约束使用alter drop语法

增加则是alter add

特别地，在MySQL中，约束都会生成索引。

级联约束

和外键的存在有关。如果更新的时候，数据在外键上没有对应的，则服务器报错。要进行修改，则要按照以下步骤：

首先删除外键

然后添加新的外键时，使用on update cascade语句

接下来如果更新或者插入时，就算现在外键中没有对应的值，也可以同步更新。

如果要删除级联更新，则使用on delete cascade语句。

1. 视图
2. 介绍

视图是一个数据查询机制。不涉及数据储存，不会占用很大的空间，而且可以用于隐藏数据或者更好地展示需要的数据。

一旦视图被创建，用户就可以把它当作一个表来用。

具体用法和 表 几乎一样。（只不过是将table换为view）

1. 为什么使用视图

隐藏数据，或者展示一部分数据

数据聚合，暴露出需要的数据的正确形式

为终端用户屏蔽数据的复杂性

1. 可更新的视图（使用update或者 insert语句更新）

满足以下条件的视图是可更新的

没有使用聚合函数，例如max（）

没有使用group by或者having子句

Select或者from子句不存在子查询，摒弃where子句中任何子查询都不应用from子句的表

视图没有使用union、union all和distinct操作符

From子句包含不止一个表或者可更新视图

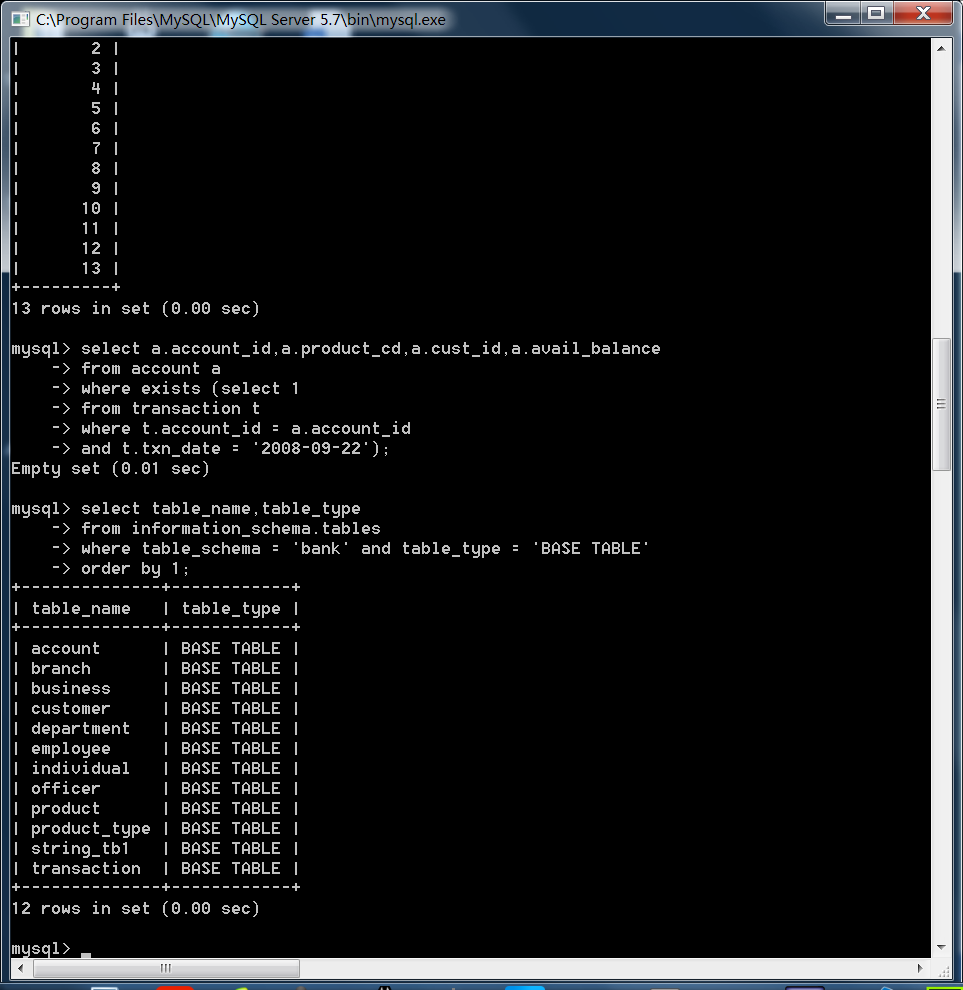
如果有不止一个表或者视图，from子句只能使用内连接

而且，单个语句只能修改一个基础表。

1. 元数据
2. 定义

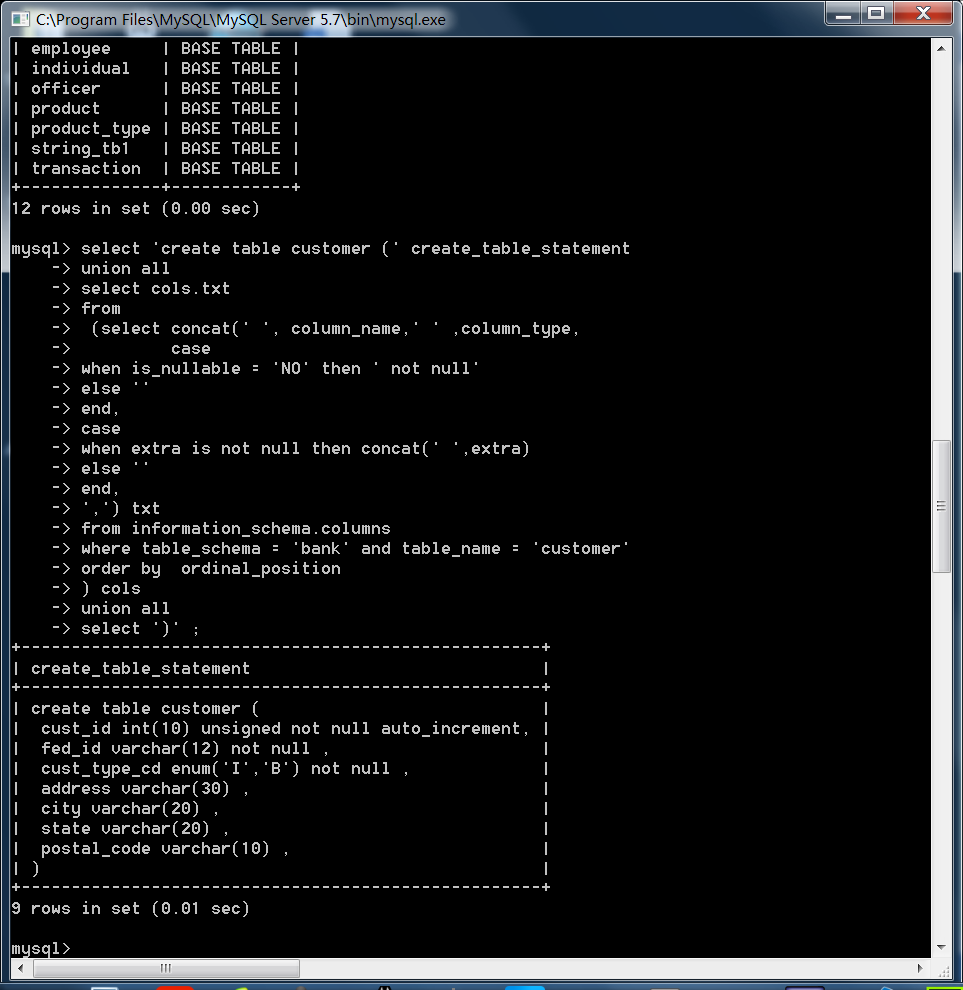
元数据就是关于数据的数据，被称为数据字典或者系统目录。

Information\_schema数据库，其中所有可用对象是视图，可以提取其中的元数据，例如tables，views，columns等等。举例用法如下

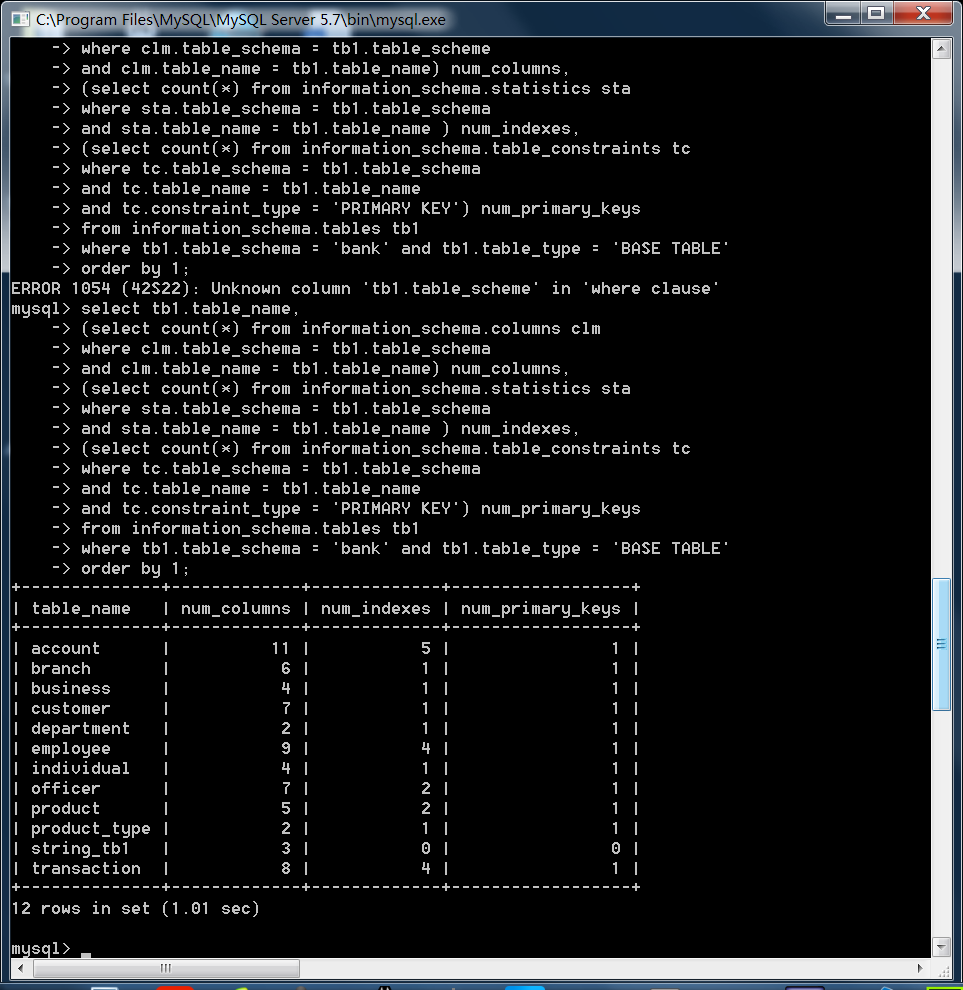


1. 使用元数据

可以使用元数据来生成脚本，很神奇的功能。例如下图

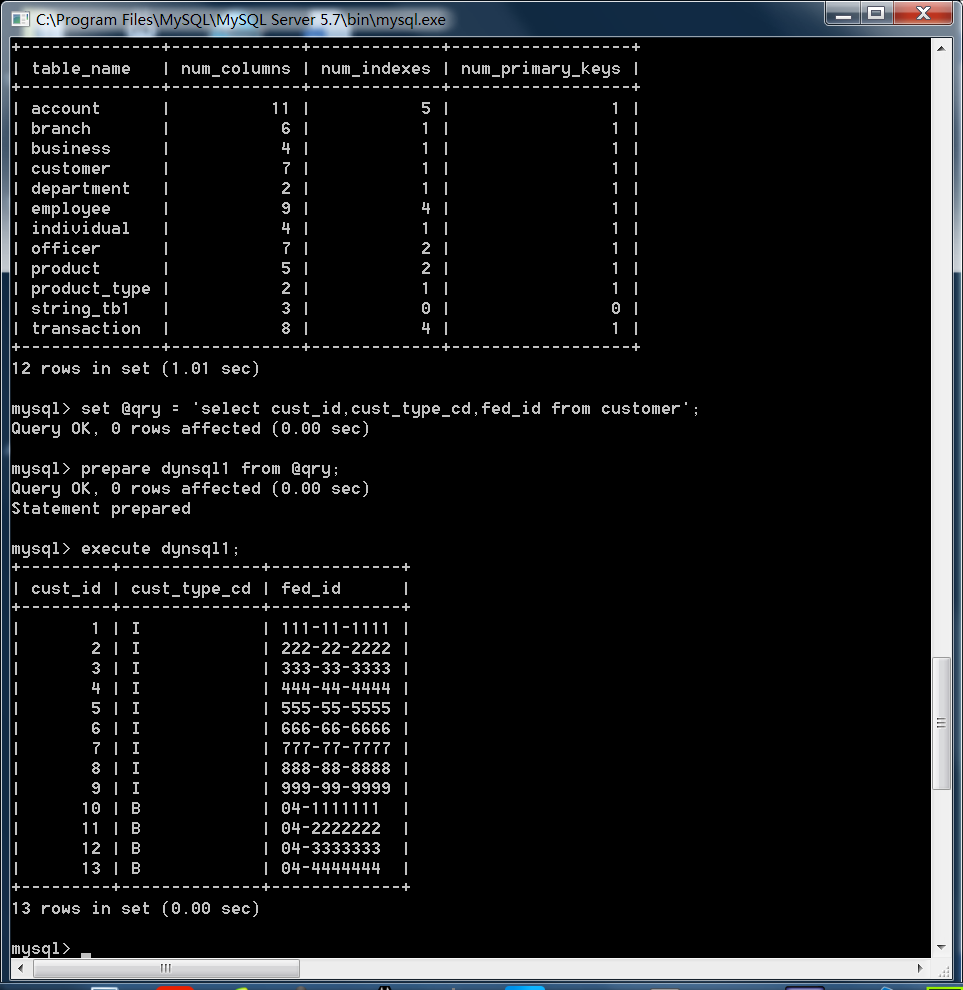


可以用来部署验证。例如下面的查询返回bank schema中每个表的列数、索引数和主键约束数。



生成动态SQL

MySQL为动态SQL执行提供了prepare、execute、deallocate语句，使用execute之前必须要prepare，例如



其中，set语句的时候，SQL语句以字符串的形式，然后prepare的时候，将该字符串赋予dynsql1变量，最后执行。

在execute语句完成之后，必须要使用deallocate语句释放资源。

通过information\_schema来获取需要的表名等等，就可以利用上面的语句生成动态SQL语句。