1. 了解SQL
   1. 数据库基础
2. 数据库（Database）：保存有组织的数据的容器（通常是一个文件或一组文件），是通过DBMS创建和操纵的容器。
3. 表（table）：某种特定类型数据的结构化清单。表名具有唯一性。
4. 模式（schema）：关于数据库和表的布局及特性的信息。
5. 列（column）：表中的一个字段。所有表都是由一个或多个列组成的。
6. 分解数据：正确地数据分解为多个列极为重要。这样才有可能利用特定的列对数据进行排序和过滤。
7. 数据类型（datatype）：所容许的数据的类型。每个表列都有相应的数据类型，它限制或容许该列存储的数据。
8. 行（row）：表中的一个记录。
9. 主键（primary key）：一列（或一组列），其值能够唯一区分表中的每个行。（应该总是定义主键，主键列不允许NULL值）
10. 主键的好习惯：不更新主键列中的值；不重用主键列的值；不在主键列中使用可能会更改的值。
    1. 什么是SQL
11. SQL（Structured Query Language）：结构化查询语言。
12. MySQL简介

2.1 什么是MySQL

A．MySQL是一种DBMS，即它是一种数据库软件。种类：一类基于共享文件系统的DBMS，另一类基于客户机-服务器的DBMS。

B．客户机-服务器软件：服务器部分是负责所有数据访问和处理的一个软件，它运行在称为数据库服务器的计算机上。客户机是与用户打交道的软件（所有请求或更改来自运行客户机软件的计算机）。

C．服务器软件为MySQL DBMS，客户机可以是MySQL提供的工具，脚本语言，web开发语言，程序设计语言。

2.2 MySQL工具（已经集成到MySQL Workbench）

A．MySQL命令实用程序

B．MySQL Administrator：用于简化MySQL服务器的管理。

C．MySQL Query Browser：用来编写和执行MySQL命令。

1. 使用MySQL

3.1 连接

A．如何连接和登录到MySQL，为了连接到MySQL，需要一下信息：主机名（计算机名）——如果连接到本地MySQL服务器，为localhost；端口（使用默认端口3306之外的端口）；一个合法的用户名；用户口令（如果需要的话）。

3.2 选择数据库

A．在执行任何数据库操作前，需要选择一个数据。为此，可使用USE关键字。

B．关键字（key word）:作为MySQL语言组成部分的一个保留字。

C．（USE crashcource;）必须先使用USE打开数据库，才能读取其中的数据。

3.3 了解数据库和表

A．（SHOW DATABASES;）数据库、表、列、用户、权限等信息被存储在数据库和表中。

B．不过，内部的表一般不直接访问。（可用SHOW命令来显示这些信息，MySQL从内部表中提取这些信息。）SHOW TABLES; SHOW COLUMNS FROM customers;

C．什么是自动变量？某些表列需要唯一值。在每个行添加到表中时，MySQL可以自动地为每个行分配下一个可用编号，不用在添加一行时手动分配唯一值。

D．DESCRIBE语句：MySQL支持用 DESCRIBE作为SHOW COLUMNS FROM的一种快捷方式。

1. 检索数据

4.1 SELECT语句

A．SELECT语句的用途是从一个或多个表中检索信息。

4.2 检索单个列

A．SELECT prod\_name FROM products;

B．未排序数据：如果没有明确排序查询结果，则返回的数据的顺序没有特殊意义。

C．结束SQL语句：多条SQL语句必须以分号分隔。如果使用的是MySQL命令行，必须加上分号来结束SQL语句。

D．SQL语句和大小写：SQL语句不区分大小写，但是一般对所有SQL关键字使用大写，而对所有列和表名使用小写。

E．使用空格：在处理SQL语句时，其中所有空格都被忽略。

4.3 检索多个列

A．当心逗号：在选择多个列时，一定要在列名之间加上逗号，但最后一个列名后不加。SELECT prod\_id, prod\_name, prod\_price FROM products;

B．数据表示：SQL语句一般返回原始的，无格式的数据。数据的格式化是一个表示问题，而不是一个检索问题。因此，表示（对齐和显示上面的价格值，用货币符号和逗号表示其金额）一般在现实该数据的应用程序中规定。

4.4 检索所有列

A．SELECT \* FROM products;

B．使用通配符：一般，除非你确定需要表中的每个列，否则最好不要使用\*通配符。检索不需要的列通常会降低检索和应用程序的性能。

C．检索未知列：由于不明确指定列名，所以能检索出名字未知的列。

4.5 检索不同行

A．SELECT vend\_id FROM products;

B．SELECT DISTINCT vend\_id FROM products; 关键字DISTINCT指示MySQL只返回不同的值（唯一）。

C．DISTINCT关键字应用于所有列而不仅是前置它的列。

4.6 限制结果

A．SELECT prod\_name FROM products LIMIT x,y; x和y为数字，第一个为开始位置，第二个数字为要检索的行数。

B．检索出来的第一行为行0而不是行1。

C．在行数不够时，MySQL将只返回它能返回的那么多行。

D．MySQL 5的LIMIT语法

4.7 使用完全限定的表名

A．使用完全限定的名字来引用列（同时使用表名和列字）

B．SELECT products.prod\_name FROM products;

C．SELECT products.prod\_name FROM crashcource.products;

1. 排序检索数据

5.1 排序数据

A．关系数据库设计理论认为：如果不明确规定排序顺序，则不应该假定检索出的数据的顺序有意义。

B．子句（clause）SQL语句由子句构成，一个子句通常由一个关键字和所提供的数据组成。

C．SELECT prod\_name FROM products ORDER BY prod\_name;

D．通常，ORDER BY子句使用的列将是为显示所选择的列。但是，用非检索的列排序数据是完全合法的。

5.2 按多个列排序

A．SELECT prod\_id, prod\_price, prod\_name

FROM products

ORDER BY prod\_price, prod\_name;

5.3 指定排序方向

A．默认是升序排序，为了进行降序排序，必须指定DESC关键字。DESC关键字只应用到直接位于其后面的列名。如果想在多个列上进行降序排序，必须对每个列指定DESC。

B．SELECT prod\_id, prod\_price, prod\_name FROM products ORDER BY prod\_price DESC;

C．区分大小写和排序顺序 在字典排序顺序中，A被视为与a相同，这是MySQL的默认行为。不过有需要可以改变数据库的设置。

D．使用ORDER BY和LIMIT的组合，能够找出一个列中最高或最低的值。SELECT prod\_price FROM products ORDER BY prod\_price DESC LIMIT 1;

1. 过滤数据

6.1 使用WHERE子句

A．只检索所需数据需要指定搜索条件（search criteria）或过滤条件（filter condition）

B．SELECT prod\_name, prod\_price FROM products WHERE prod\_price = 2.5;

C．SQL过滤与应用过滤 数据也可以在应用层过滤。但是，这种实现并不令人满意。

D．WHERE子句排在ORDER BY之前。

6.2 WHERE子句操作符

A．SELECT prod\_name, prod\_price FROM products WHERE prod\_name = ‘fuses’;MySQL在执行匹配时默认不区分大小写。

B．何时使用引号：单引号用来限定字符串的。如果将值与串类型的列进行比较，则需要限定引号，用来跟数值列进行比较的值不用引号。

C．范围值检查：为了检查某个范围的值，可使用BETWEEN操作符。SELECT prod\_name, prod\_price FROM products WHERE prod\_price BETWEEN 5 AND 10;

D．NULL 无值（no value）它与字段包含0，空字符串或仅仅包含空格不同。SELECT prod\_name FROM products WHERE prod\_price IS NULL;

E．NULL与不匹配，在过滤数据时，一定要验证返回数据中确实给出了被过滤列具有NULL的行。

1. 数据过滤

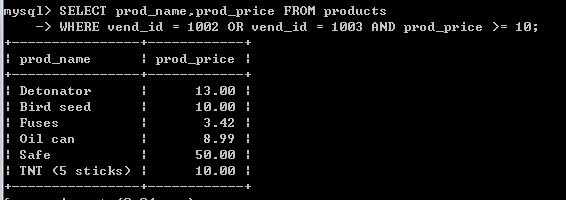
7.1 组合WHERE子句

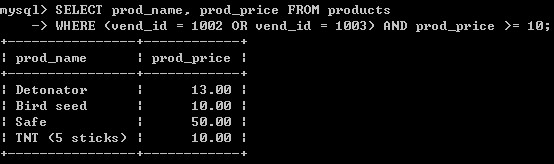
A．为了进行更强的过滤控制，MySQL允许给出多个WHERE子句，以AND子句的方式或OR子句的方式使用。

B．操作符（operator）用来联结或改变WHERE子句中的子句的关键字，也称为逻辑操作符（logical operator）

C．AND 用在WHERE子句中的关键字，用来指示检索满足所有给定条件的行。OR 它指示MySQL检索匹配任一条件的行。

D．在WHERE子句中使用圆括号 任何时候使用具有AND和OR操作符的WHERE子句，都应该使用圆括号明确地分组操作符。





7.2 IN操作符

A．IN操作符用来指定条件范围，范围中的每个条件都可以进行匹配，IN取合法值的由逗号分隔清单，全都括在圆括号中。

B．IN的最大优点是可以包含其他SELECT语句，使得能够更动态地建立WHERE子句。

SELECT prod\_name,prod\_price

FROM products

WHERE vend\_id IN (1002,1003)

ORDER BY prod\_name;

7.3 NOT

A．WHERE子句中用来否定后跟条件的关键字。（NOT IN）

B．MySQL中的NOT MySQL支持使用NOT对IN，BETWEEN和EXISTS子句取反，这与多数其他DBMS允许使用NOT对各种条件取反有很大的差别。SELECT prod\_name, prod\_price FROM products WHERE vend\_id NOT IN (1002, 1003) ORDER BY prod\_name;

1. 用通配符进行过滤

8.1 LIKE操作符

A．利用通配符可创建比较特定数据的搜索模式。

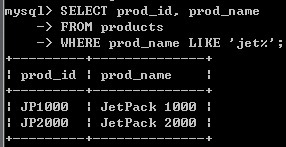
　　B．通配符（wildcard）用来匹配值的一部分的特殊字符。

C．搜索模式（search pattern）由字面值，通配符或两者组合构成的搜索条件。

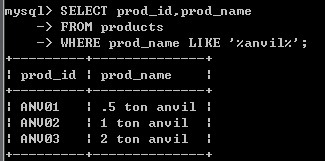
D．LIKE指示MySQL，后跟的搜索模式利用通配符匹配而不是直接相等匹配进行比较。

E．谓词：操作符在它作为谓词（predicate）时，不是操作符，从技术上说，LIKE是谓词而不是操作符。

F．百分号（%）通配符：在搜索串中，%表示任何字符出现任意次数。以下例子使用了搜索模式’jet%’:



G．区分大小写：根据MySQL的配置方式，搜索可以是区分大小写的。



H．注意尾空格：尾空格可能会干扰通配符匹配。注意NULL：虽然似乎％通配符可以匹配任何东西，但是有一个例外，即NULL。

I．下划线（\_）通配符：下划线只匹配单个字符而不是多个字符，不能多也不能少。

8.2 使用通配符的技巧

A．使用通配符搜索的处理一般要比前面讨论的其他搜索所花时间更长。

B、不要多度使用通配符。在确实需要使用通配符时，除非绝对有必要，否则不要把他们用在搜索模式的开始处。仔细注意通配符的位置。

1. 用正则表达式进行搜索

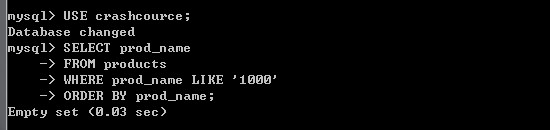
9.1 正则表达式介绍

A．正则表达式是用来匹配文本的特殊的串（字符集合）。所有种类的程序设计语言，文本编辑器，操作系统等都支持正则表达式。

9.2 使用MySQL正则表达式

A．MySQL仅仅支持多数正则表达式实现的一个很小的子集。

B．LIKE匹配整个列值。如果被匹配的文本仅在列值中出现，LIKE将不会找到它，相应的行也不被返回（除非使用通配符）。而REGEXP在列值内进行匹配，如果被匹配的文本在列值中出现，REGEXP将会找到它，相应的行将被返回。看下面的例子：





C．匹配不区分大小写MySQL中的正则表达式匹配不区分大小写，为区分大小写可使用BINARY关键字，如WHERE prod\_name REGEXP BINARY ‘JetPack .000’。

D．进行OR匹配：为搜索两个串之一，使用|，如下所示：



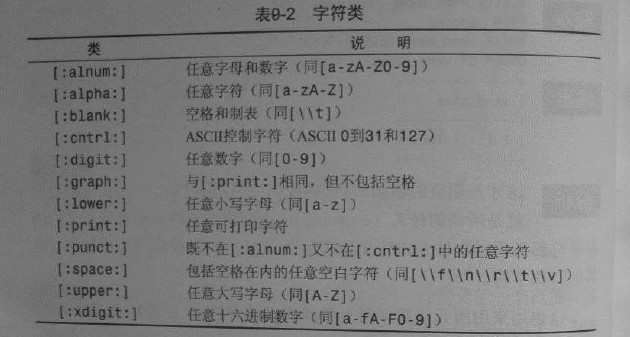
E．匹配几个字符之一：匹配任何单一字符。如下所示：



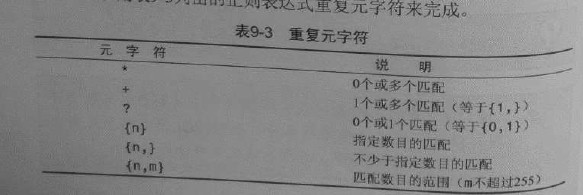
F．可使用-来定义一个范围。如[1-9],[a-b]。.在正则表达式中匹配任意一个字符。

G．为了匹配特殊字符，必须用\\为前导，例如\\-,\\.等。这种处理即转义（escaping）。

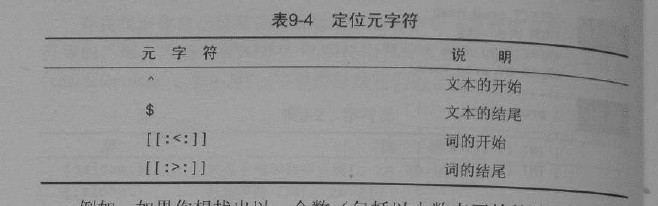
1. 为了更方便工作，可以使用预定义的字符集，称为字符类（character class）。



1. 正则表达式重复元字符：

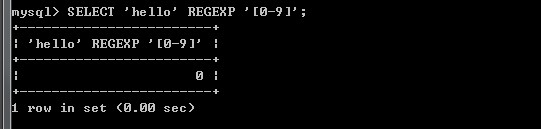


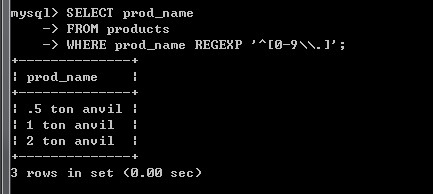
J．定位符：



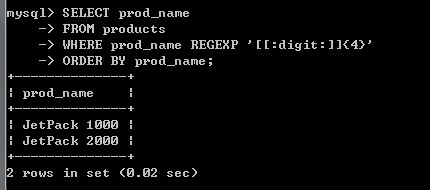
K．使REGEXP起类似LIKE的作用，它们的区别在于LIKE匹配整个串而REGEXP匹配子串。利用定位符，通过用^开始每个表达式，用$结束每个表达式。

L．简单的正则表达式测试：可以在不使用数据库表的情况下用SELECT来测试正则表达式。REGEXP检查总是返回0或1（匹配）。









1. 创建计算字段

10.1计算字段

A．存储在数据库表中的数据一般不是应用程序所需要的格式。我们需要直接从数据库中检索出转换，计算或格式化过的数据；而不是检索出数据，然后再在客户机应用程序或报告程序中重新格式化。

B．字段（field）：基本上和列（COLUMN）的意思相同，经常互换使用，不过数据库列一般成为列，而术语字段通常用在计算字段的连接上。

C．客户机与服务器的格式

10.2 拼接字段

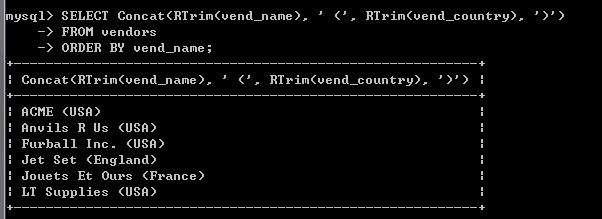
A．拼接（concatenate）将值联结到一起构成单个值。（把两个列拼接起来）

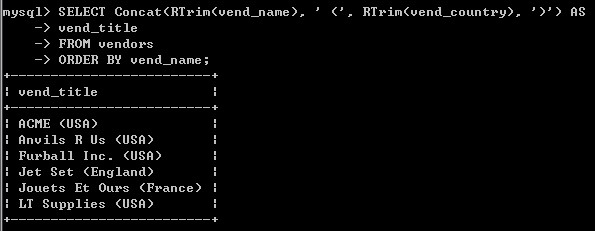
B．MySQL的不同之处：多数DBMS使用+或||来实现拼接，而MySQL使用Concat()函数来实现。

C．Trim函数：MySQL除了支持RTrim()（去掉串右边的空格），还支持LTrim()及Trim()。

D．使用别名（alias），别名是一个字段或值的替换名。别名用AS关键字赋予。任何客户机应用都可以按名引用这个列。别名也可称导出列（derived COLUMN）。

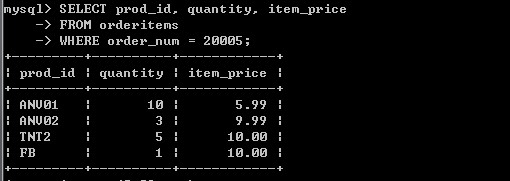


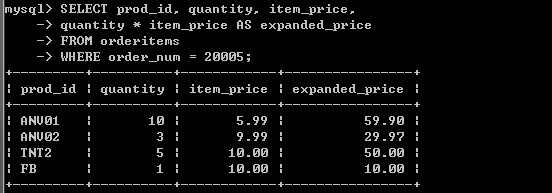




10.3 执行算术计算

A．如何测试计算：SELECT提供了测试和试验函数与计算的一个很好的办法。



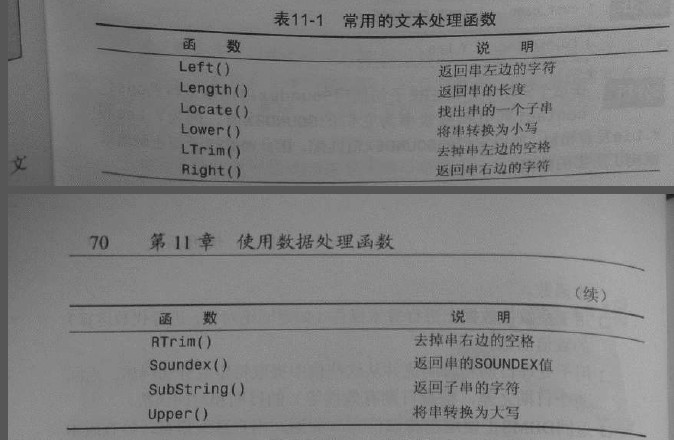


1. 使用数据处理函数

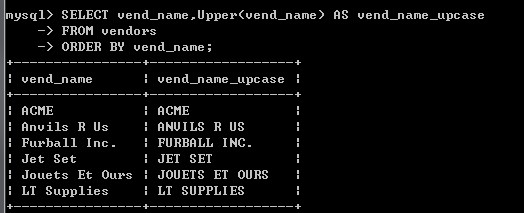
11.1 函数

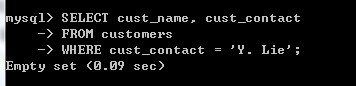
A．函数没有SQL的可移植性强：能运行在多个系统上的代码称为可移植的（portable）。如果你使用函数，应该保证做好代码注释。

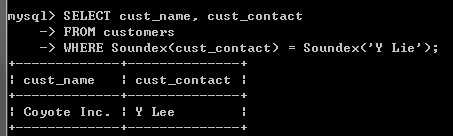
B．用于处理文本串的文本函数。用于在数值数据上进行算术操作的数值函数。用于处理日期和时间值并从这些值中提取特定成分的日期和时间函数。返回DBMS正使用的特殊信息的系统函数。

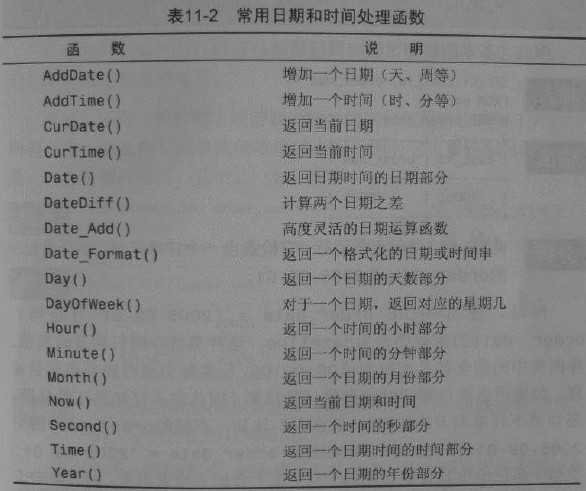


1. SOUNDEX是一个将任何文本串转换为描述其语音表示的字母数字模式的算法。

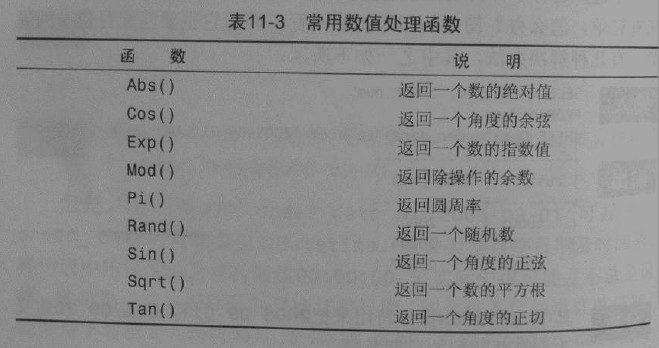








1. MySQL使用的日期格式：yyyy-mm-dd，应该总是使用4位数字的年份。

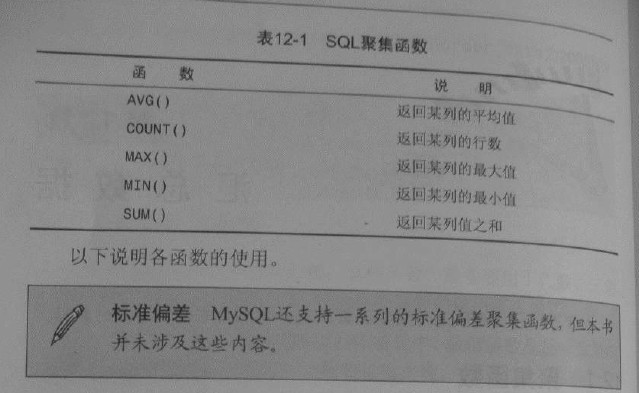


1. 汇总数据

12.1 聚集函数

A．确定表中行数，获得表中行组的和，找出表列的最大值，最小值和平均值。要的是汇总信息，而不是表中数据。

1. 聚集函数（aggregate function）：运行在行组上，计算和返回单个值的函数。

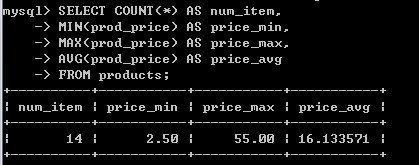


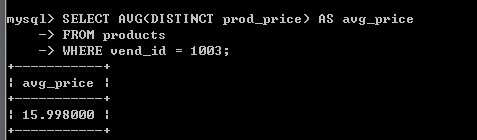
C．只用于单个列，AVG()只能用来确定特定数值列的平均值，而且列名必须作为函数参数给出。AVG()函数忽略列值为NULL的行。

12.2 聚集不同值

A．对所有的行执行计算，指定ALL参数或不给参数（因为ALL是默认行为）。只包含不同的值，指定DISTINCT参数。DISTINCT必须使用列名，不能用于计算或表达式。

12.3 组合聚集函数

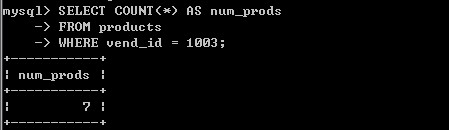




1. 分组数据

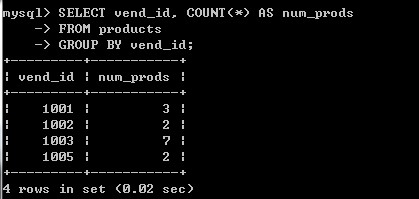
如何分组数据，以便能汇总表内容的子集。

13.1 数据分组

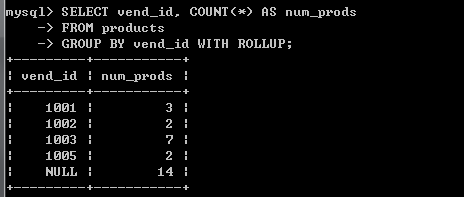


目前为止的所有计算都是在表的所有数据或匹配特定的WHERE子句的数据上进行的。

13.2 创建分组

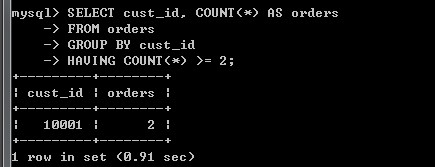


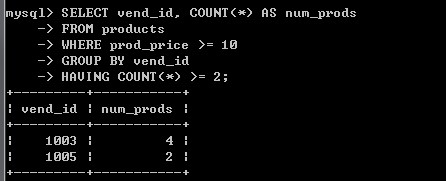
1. GROUP BY子句指示MySQL分组数据，然后对每个组而不是整个结果集进行聚集。
2. GROUP BY子句可以包含任意数目的列。除了聚集计算语句外，SELECT语句中的每个列都必须在GROUP BY子句中给出。
3. 如果在GROUP BY子句中嵌套了分组，数据将在最后规定的分组上进行汇总。在子句中列出的每个列都必须是检索列或有效的表达式。
4. 如果分组中具有NULL值，则NULL将作为一个分组返回。
5. GROUP BY子句必须出现在WHERE子句之后，ORDER BY子句之前。
6. 使用WITH ROLLUP关键字，可以得到每个分组以及每个分组汇总级别（针对每个分组）的值。

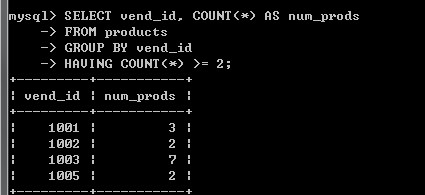


13.3 过滤分组

A．WHERE过滤指定的是行而不是分组。WHERE没有分组的概念。我们可以使用HAVING子句过滤分组。HAVING支持所有WHERE操作符。WHERE在数据分组前进行过滤，HAVING在数据分组后进行分组。

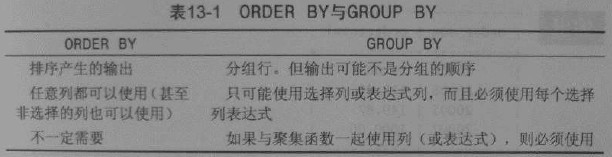




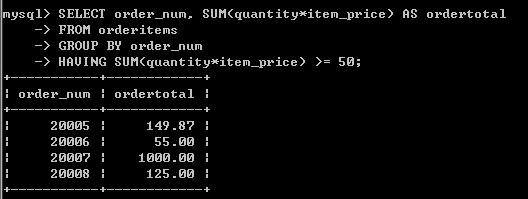


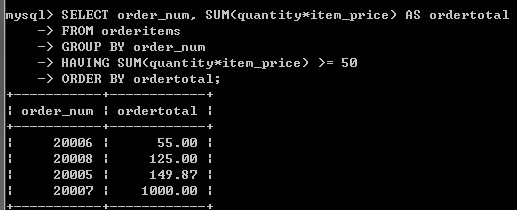
13.4 分组和排序

1. ORDER BY与GROUP BY的区别：

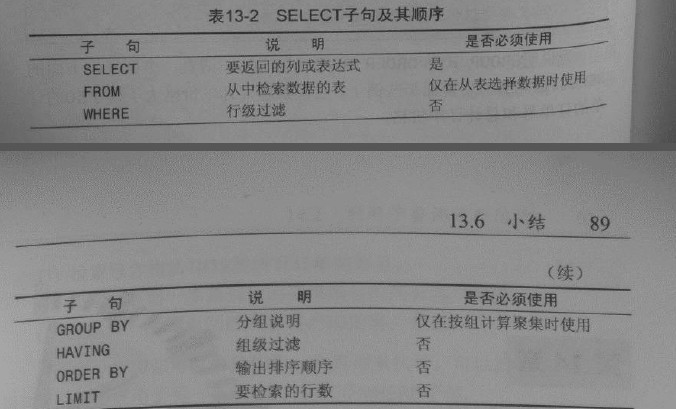


C．不要忘记ORDER BY ：一般在使用GROUP BY子句时，应该也给出ORDER BY子句。这是保证数据正确排序的唯一方法。





13.5 SELECT子句顺序



1. 使用子查询

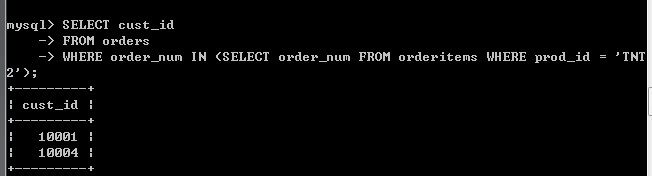
14.1 子查询

A．MySQL引入了对子查询的支持。SELECT语句是SQL的查询。迄今为之我们所看到的所有SELECT语句都是简单查询，即从单个数据库表中检索数据的单条语句。

B．子查询（subquery），即嵌套在其他查询中的查询。查询（query）：任何SQL语句都是查询，但此术语一般指SELECT语句。

14.2 利用子查询进行过滤

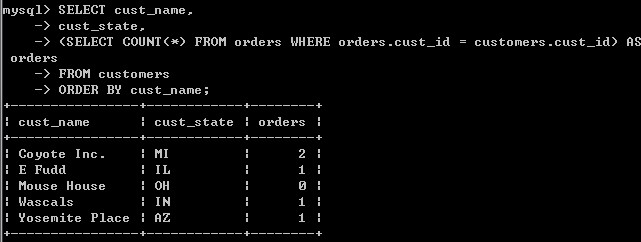
A．格式化SQL：把子查询分解为多行并且适当地进行缩进，能极大地简化子查询的使用。



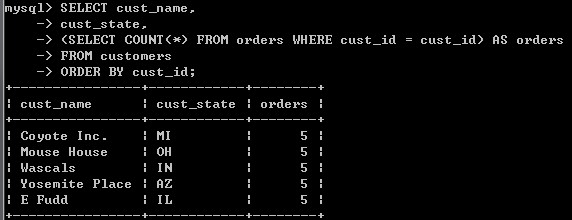
B．子查询和性能

14.3 作为计算字段使用子查询

A．相关子查询（correlated subquery）涉及外部查询的子查询。



如果以上查询不使用完全限定的列名：



C，这里给出的样例代码运行良好，但它并不是解决这种数据检索的最有效的方法。

D，逐渐增加子查询来建立查询：用子查询建立（和测试）查询的最可靠的方法是逐渐进行。

1. 联结表

15.1 联结（join）

A．在数据检索查询的执行中联结（join）表。

B．关系表的设计就是要保证把信息分解成多个表，一类数据一个表。各表通过某些常用的值（即关系设计中的关系（relational））互相关联。

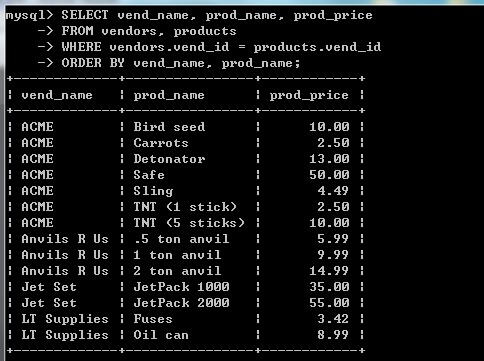
C．外键（foreign key）：外键为某个表中的一列，它包含另外一个表的主键值，定义了两个表之间的关系。

D．可伸缩性（scale）：能够适应不断增加的工作量而不失败。

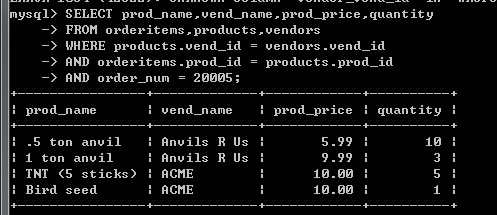
E．为什么要使用联结：如果数据存储在多个表中怎样用单条SELECT语句检索出数据。

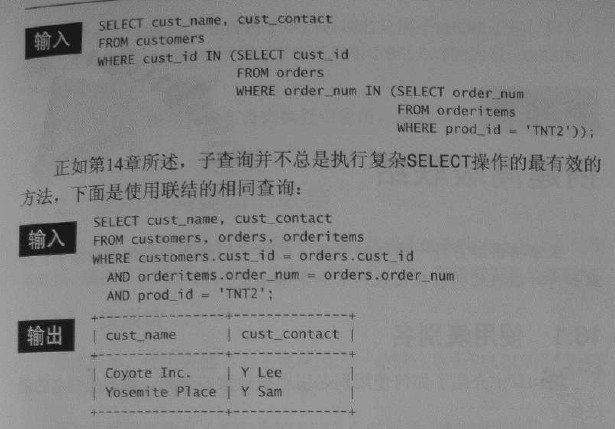
F．维护引用完整性：通过在表的定义中指定主键和外键来实现的。

15.2 创建联结



1. 笛卡尔积（cartesian product）：由没有联结条件的表关系返回的结果为笛卡尔积。
2. 不要忘了WHERE子句：应该保证所有联结都有WHERE子句，否则MySQL将返回比想要的数据多得多的数据。
3. 叉联结：叉联结（cross join）的笛卡尔积的联结类型。
4. 目前为之所用的联结称为内部联结或等值联结（equijoin），它基于两个表之间的相等测试。SELECT vend\_name, prod\_name, prod\_price FROM vendors INNER JOIN products ON vendors.vend\_id = products.vend\_id;(ANSI SQL规范首选INNER JOIN语法)
5. 联结多个表：性能考虑——MySQL在运行时关联指定的每个表以处理联结，这种处理可能是非常耗费资源的。
6. 多做实验：为执行任一给定的SQL操作，一般存在不止一种方法。很少有绝对正确或绝对错误的方法。性能可能会受操作类型，表中数据量，是否存在索引或键以及其他一些条件的影响。



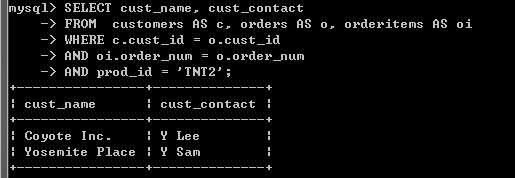


1. 创建高级联结

16.1 使用表别名

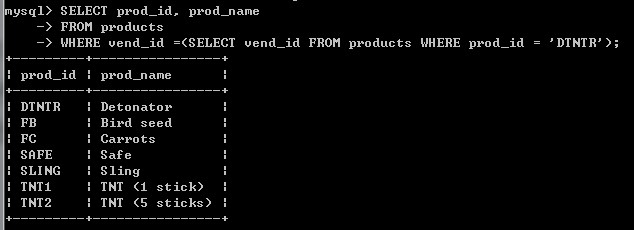
A．别名除了用于列名和计算字段外，SQL还允许给表名起别名，这样做主要是为了：缩短SQL语句；允许在单条SELECT语句中多次使用相同的表。

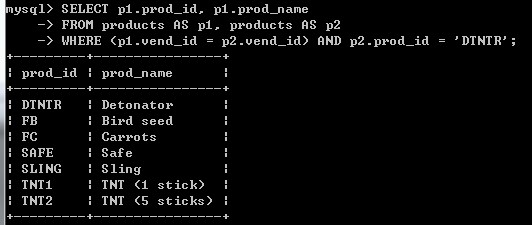
B．表别名只在查询执行中使用，与列别名不一样，表别名不返回到客户机。



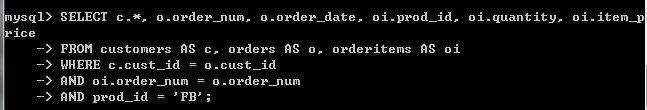
16.2 使用不同类型的联结

A．用自联结而不用子查询 自联结通常作为外部语句用来替代从相同表中检索数据时使用的子查询语句。

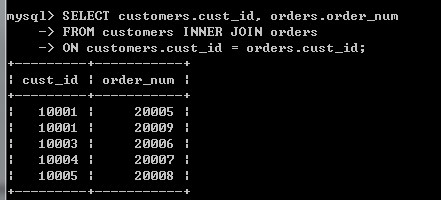


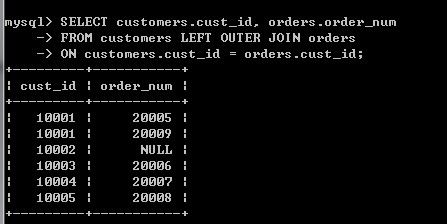


1. 自然联结排除多次出现，使每个列只返回一次。它是这样一种联结，其中你只能选择哪些唯一的列。这一般是通过对表使用通配符(SELECT \*)，对所有其他表的列使用明确的子集来完成的。

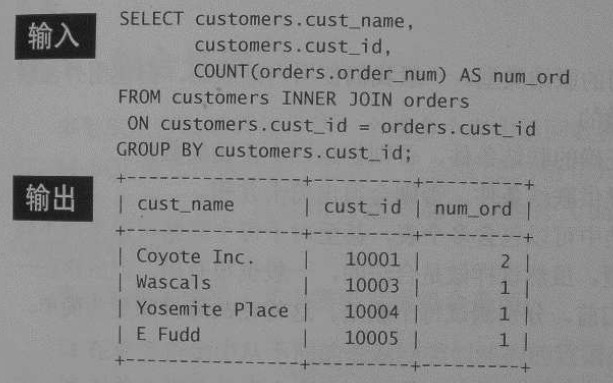


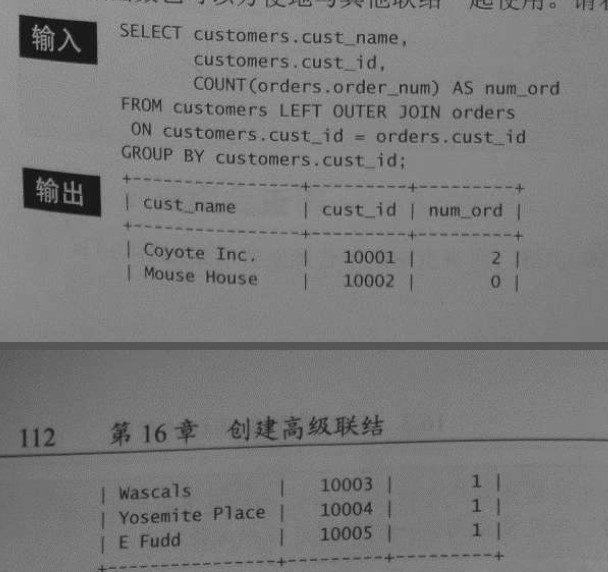
1. 外部联结：联结包含了哪些在相关表中没有关联行的行。例如，列出所有产品以及订购数量，包括没有人订购的产品。
2. LEFT OUTER JOIN从FROM子句的左边表中选择所有行。RIGHT OUTER JOIN则反之
3. 没有\*=操作符 MySQL不支持简化字符\*= 和=\*的使用，这两种操作符在其他DBMS中是很流行的。外部联结的类型：左外部联结和右外部联结。





16.3 使用带聚集函数的联结





16.2 使用联结和联结条件

A．注意所使用的联结类型。一般我们使用内部联结，但使用外部联结也是有效的。

B．保证使用正确的联结条件，否则将返回不正确的数据。

C．应该总是提供联结条件，否则会得出笛卡尔积。

D．在一个联结中可以包含多个表，设置对于每个联结可以采用不同的联结类型。虽然这样做是合法的，一般也很有用，但应该在一起测试它们前，分别测试每个联结。

1. 组合查询

17.1 组合查询

A．MySQL允许执行多个查询（多条SELECT语句），并将结果作为单个查询结果集返回。这些组合查询通常称为并（union）或复合查询（compound query）。

B．需要使用组合查询的两种基本情况：在单个查询中从不同的表返回类似结构的数据。对单个表执行多个查询，按单个查询返回数据。

17.2 创建组合查询

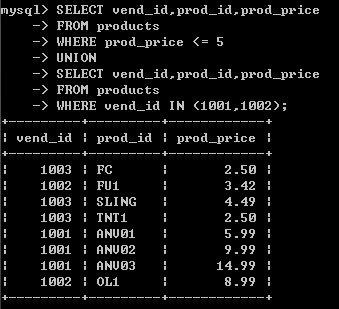
A．UNION操作符：利用UNION，可给出多条SELECT语句，将它们的结果组合成单个结果集。

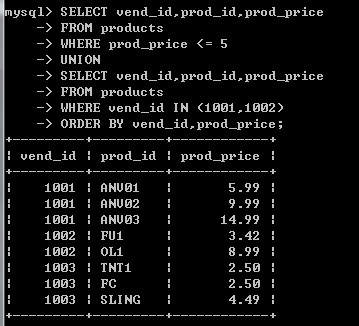
B．UNION规则：UNION必须由两条或两条以上的SELECT语句组成，语句之间用关键字UNION分隔。UNION中的每个查询必须包含相同的列，表达式或聚集函数。列数据类型必须兼容。

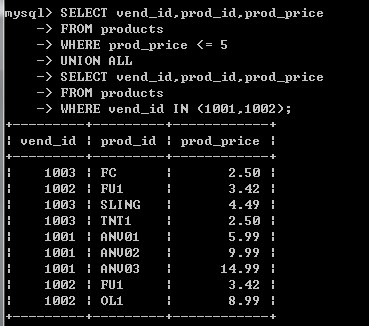
C．包含或取消重复的行：使用UNION ALL，MySQL不取消重复的行。

D．对组合查询结果排序：在用UNION组合查询时，只能使用一条ORDER BY。

E．组合不同的表：使用UNION的组合查询可以应用不同的表。







1. 全文本搜索

18.1 理解全文本搜索

A．并非所有引擎都支持全文本搜索：MySQL支持几种基本的数据库引擎。MyISAM支持全文本搜索，而InnoDB不支持全文本搜索。

18.2 使用全文本搜索

A．为了进行全文本搜索，必须索引被搜索的列，而且要随着数据的改变不断地重新索引。

B．不要在导入数据时使用FULLTEXT：如果正在导入数据到一个新表，此时不应该启用FULLTEXT索引。

C．使用完整的Match()说明：传递给Match()的值必须与FULLTEXT()定义中的相同。如果指定多个列，则必须列出它们（而且次序正确）

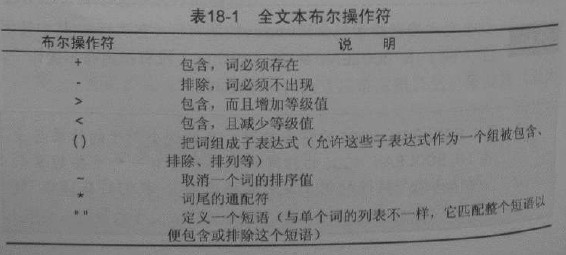
D．搜索不区分大小写：除非使用BINARY方式，否则全文本搜索不区分大小写。

E．使用查询扩展：查询扩展用来设法放宽所返回的全文本搜索结果的范围。利用查询扩展，能找出可能相关的结果，即使它们并不精确包含所查找的词。

F．行越多越好：表中的行越多，使用查询扩展返回的结果越好。

G．布尔文本搜索（boolean mode）：即使没有FULLTEXT索引也可以使用。但这是一种非常缓慢的操作。在布尔方式中，不按等级值降序排序返回的行。

H．以布尔方式，可以提供关于如下内容的细节：要匹配的词，要排斥的词。排列提示，表达式分组，另外一些内容。



1. 插入数据

19.1 数据插入

A．插入可以用几种方式使用：插入完整的行；插入行的一部分；插入多行；插入某些查询的结果。

B．插入及系统安全：可针对每个表或每个客户，利用MySQL的安全机制禁止使用INSERT语句。

19.2 插入完整的行

A．INSERT INTO Customers VALUES(NULL, ‘Pep E. LaPew’, ‘100 Main Street’, ‘Los Angeles’, ‘CA’, ‘90046’, ‘USA’, NULL, NULL); 没有输出：INSERT语句一般不会产生输出。

B．总是使用列的列表：一般不要使用没有明确给出列的列表的INSERT语句。

C．仔细地给出值：不管哪种INSERT语法，都必须给出VALUES的正确数目。

D．省略列：如果表的定义允许，则可以在INSERT操作中省略某些列。省略的列必须满足以下某个条件。该列定义为允许NULL值（无值或空值）。在表定义给出默认值。

E．如果数据检索是最重要的，则你可以通过在INSERT和INTO之间添加关键字LOW\_PRIORITY，指示MySQL降低INSERT语句的优先级。这也适用与UPDATA和DELETE语句。

F．编写INSERT语句的更安全的方法：INSERT INTO Customers(cust\_name, cust\_address, cust\_city, cust\_state, cust\_zip, cust\_country, cust\_contact, cust\_email) VALUES( ‘Pep E. LaPew’, ‘100 Main Street’, ‘Los Angeles’, ‘CA’, ‘90046’, ‘USA’, NULL, NULL);它在表名后的括号里明确地给出了列名。在插入行时，MySQL将用VALUES列表中的相应值填入列表中的对应项。

19.3 插入多个行

A．可以使用多条INSERT语句，设置一次提交它们，每条语句用一个分号结束。或者只要每条INSERT语句中的列名（和次序）相同：INSERT INTO Customers(cust\_name, cust\_address, cust\_city, cust\_state, cust\_zip, cust\_country) VALUES( ‘Pep E. LaPew’, ‘100 Main Street’, ‘Los Angeles’, ‘CA’, ‘90046’, ‘USA’), ( ‘M. Martain’, ‘42 Galaxy Way’, ‘New York’, ‘NY’, ‘11213’, ‘USA’);其中单条INSERT语句有多组值，每组值用一对圆括号括起来，用逗号分隔。后者的性能高。

19.4 插入检索出的数据

A．INSERT SELECT：它由一条INSERT语句和一条SELECT语句组成的。

B．INSERT SELECT中的列名：不一定要求列名匹配。MySQL只用的是列的位置，因此SELECT中第一列将用来填充表列中指定的第一列，以此类推。

1. 更新和删除数据

20.1 更新（修改）数据

A．UPDATE：更新表中特定行；更新表中所有行。

B．UPDATE语句由3部分组成：要更新的表；列名和它们的新值（SET）；确定要更新行的过滤条件。

C．在UPDATE语句中使用子查询。IGNORE关键字：如果用UPDATE语句更新多行，并且在更新这些行中的一行或多行时出现一个错误，则整个UPDATE操作被取消。忽略错误，继续进行更新。UPDATE IGNORE customers…

D．为了删除某个列的值，可设置它为NULL

20.2 删除数据

A．DELETE语句：从表中删除特定的行；从表中删除所有行。

B．删除表的内容而不是表；更快的删除：如果想从表中删除所有行，不要使用DELETE，可使用TRUNCATE TABLE语句（实际是删除原来的表并重新创建一个表）。

20.3 更新和删除的知道原则

A．小心使用：MySQL没有撤销（undo）按钮。

B．除非确实打算更新和删除每一行，否则绝对不要使用不带WHERE子句的UPDATE或DELETE语句。

C．保证每个表都有主键。

D．先使用SELECT测试WHERE子句，保证它过滤的是正确的记录。

E．使用强制实施引用完整性的数据，这样MySQL将不允许删除具有与其他表相关联的数据的行。

1. 创建和操纵表

21.1 创建表

A．一般有两种创建表的方法：使用具有交互式创建和管理表的工具（实际上也是使用了MySQL语句）；表也可以直接用MySQL语句操纵。

B．处理现有的表：在创建新表时，指定的表名必须不存在，否则将出错。不能简单地用创建表语句覆盖已有的表。

C．如果你仅想在一个表不存在时创建它，应该在表名后给出IF NOT EXITS。

D．如果不指定NOT NULL，则认为指定的是NULL。NULL值是没有值，它不是空串。空串是一个有效的值，它不是无值。

E．使用AUTO\_INCREMENT：AUTO\_INCREMENT告诉MySQL，本列每当增加一行时自动增量。每个表只允许一个这样的列，而且它必须被索引。

F．指定默认值（DEFAULT）：如果插入行时没有给出值，MySQL允许指定此时使用的默认值。使用默认值而不是NULL值，特别是对用于计算或数据分组的列更是如此。

G．引擎类型：ENGINE=InnoDB语句,与其他DBMS一样，MySQL有一个具体管理和处理数据的内部引擎。但MySQL与其他DBMS不一样，它具有多种引擎。它打包多个引擎，这些引擎都隐藏在MySQL服务器内，全都能执行Create TABLE和SELECT等命令。

H．InnoDB是一个可靠的事务处理引擎，它不支持全文本搜索。MEMORY在功能上等同于MyISAM，但由于数据存储在内存中，速度很快，适合于临时表。MyISAM是一个性能极高的引擎，它支持全文本搜索。但不支持事务处理。

I．外键不能跨引擎：即使用一个引擎的表不能引用具有使用不同引擎的外键。

21.2 更新表

A．ALTER TABLE之后给出要更改的表名，所做更改的列表。例如，ALTER TABLE vendors ADD vend\_phone CHAR(20); ALTER TABLE vendors DROP COLUMN vend\_phone;

B．ALTER TABLE的一种常见用途是定义外键。

21.3 删除表

A．DROP TABLE语句

21.4 重命名表

A．RENAME TABLE customers TO customers;

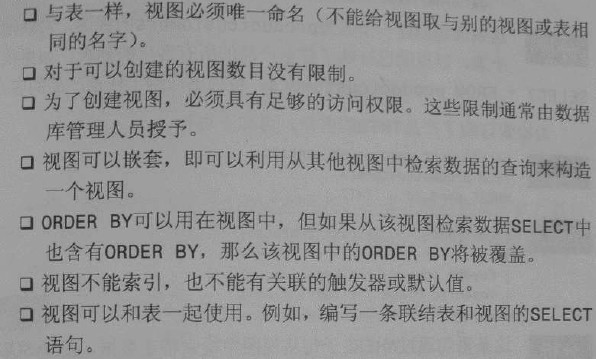
1. 使用视图

22.1 视图

A．需要MySQL 5 ，它添加了对视图的支持。视图是虚拟的表，它只包含使用时动态检索数据的查询。

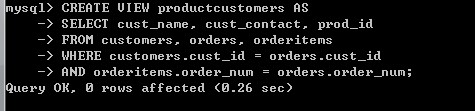
B．为什么使用视图：重用SQL语句。简化复杂的SQL操作。使用表的组成部分而不是整个表。保护数据，给用户授予表的特定部分的访问权限而不是整个表的访问权限。更改数据格式和表示。使用视图要考虑性能问题。

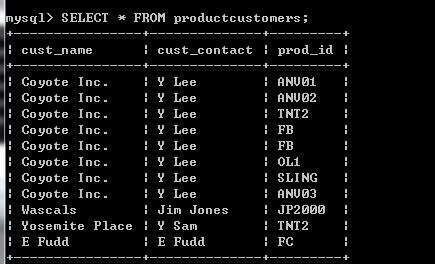
C．视图的规则和限制

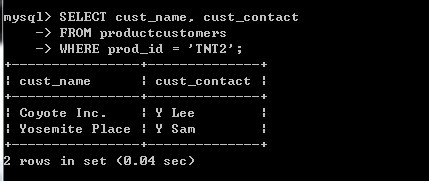


22.2 使用视图

A．CREATE VIEW ，SHOW CREATE VIEW viewname，DROP VIEW viewname，更新视图时，可以先用DROP再用CREATE，也可以直接用CREATE OR REPLACE VIEW。







1. 创建可重用的视图：创建不受特定数据限制的视图是一种好方法。例如上面创建的视图返回购买所有产品的客户而不仅仅是生产TNT2的客户。
2. 用视图重新格式化检索出的数据：
3. 用视图过滤不想要的数据：
4. WHERE子句与WHERE子句：如果从视图检索数据时使用了一条WHERE子句，则两组子句（一组在视图中，另一组是传递给视图的）将自动组合。
5. 使用视图与计算字段
6. 更新视图：通常，视图是可更新的。基本上可以说，如果MySQL不能正确地确定被更新的基数据，则不允许更新。一般，应该将视图用于检索（SELECT语句）而不用于更新（INSERT，UPDATE和DELETE）

1. 使用存储过程

23.1 存储过程（需要MySQL 5）

A．存储过程简单来说，就是为以后的使用而保存的一条或多条MySQL语句的集合。可将其视为批文件，虽然它们的作用不仅限于批处理。

23.2 为什么要使用存储过程

A．通过把处理封装在容易使用的单元中，简化复杂的操作。

B．由于不要求反复建立一系列处理步骤，这保证数据的完整性。（防止错误）

C．简化对变动的管理。（安全性，通过存储过程限制对基础数据的访问减少了数据讹误的机会）

D．提高性能。

E．你可能有没创建存储过程的安全访问权限。编写存储过程需要更高的技能，更加丰富的经验（这是缺陷）MySQL将编写存储过程的安全和访问与执行存储过程的安全和访问区分开来。

23.3 使用存储过程（如何执行（运行）它们）

A．执行存储过程（调用：CALL语句）：CALL productpricing(@pricelow, @pricehigh, @priceaverage);

B．创建存储过程，例如：CREATE PROCEDURE productpricing() BEGIN SELECT Avg(prod\_price) AS priceaverage FROM products; END;

C．临时改变命令行实用程序的语句分隔符：DELIMITER // 这表示告诉命令行实用程序使用//作为新的语句结束分隔符。

D．删除存储过程：DROP PROCEDURE (IF EXISTS) productpricing;

E．使用参数：一般存储过程并不显示结果，而是把结果返回给指定的变量（variable：内存中的一个特定的位置，用来临时存储数据）

F．MySQL支持IN（传递给存储过程），OUT（从存储过程传出）和INOUT（对存储过程传入和传出）类型的参数，所有MySQL变量都必须以@开始。

G．建立智能存储过程（包含业务规则和智能处理）

H．检查存储过程SHOW CREATE PROCEDURE语句，SHOW PROCEDURE STATUS;

1. 使用游标

24.1 游标（需要MySQL 5）

A．有时，需要在检索出来的行中前进或后退一行或多行。

B．游标（cursor）是一个存储在MySQL服务器上的数据库查询，它不是一条SELECT语句，而是被该语句检索出来的结果集。

C．主要是用于交互式应用。只能用于存储过程：不像多数DBMS，MySQL游标只能用于存储过程（和函数）。

24.2 使用游标

A．创建游标：用DECLARE语句创建 CREATE PEOCEDURE processorders() BEGIN DECLARE ordernumbers CURSOR FOR SELECT order\_num FROM orders; END;

B．打开和关闭游标：OPEN CURSOR语句和CLOSE CURSOR语句。

C．在一个游标打开后，可以使用FETCH语句分别访问它的每一行。

D．DECLARE语句定义的局部变量必须在定义任何游标或句柄之前定义。

1. 使用触发器

25.1 触发器

1. 管理事务处理
2. 全球化和本地化
3. 安全管理
4. 数据库维护
5. 改善性能