第 1 章 mysql 的安装与配置

1. 社区版是自由下载而且完全免费，但不提供任何技术支持，适用于普通用户。企业版则是收费，不能在线下载，相应地，它提供了更多的功能和更完备的技术支持。
2. mysql 每个版本分别有 3 种类型：

- standard：推荐大多数用户下载。
- max：附加一些新特性，但这些新特性还没有通过正式的测试发布。
- debug：包括一些调试信息，会影响系统性能，所以不推荐使用。

3. mysql 的 rpm 包包括很多套件，一般只安装 server 和 client 就可以了。其中 server 包是 mysql 服务端套件，为用户提供核心的 mysql 服务；client 包是连接 mysql 服务的客户端工具，方便管理员和开发人员在服务器上进行各种管理工具。（server、client）
4. linux 下安装 mysql：

```
[root@localhost zzx]# rpm -ivh mysql-server-community-5.0.45-0.rhel3.i386.rpm
#server 版
[root@localhost zzx]# rpm -ivh mysql-client-community-5.0.45-0.rhel3.i386.rpm
#client 版
[root@localhost zzx]# mysql -uroot                #运行
```

5. linux 下也可以在多个位置部署配置文件，大多数情况下都放在/etc 下，文件名为 my.cnf。对于初学者来说，建议用 mysql 自带的多个样例参数文件来代替实际的参数文件。在 linux 下，如果安装方式是 rpm 包，则自带的参数文件会放到/usr/share/mysql。（/etc/my.cnf）

```
[root@localhost mysql]# pwd
/usr/share/mysql
[root@localhost mysql]# ls *.cnf
my-huge.cnf my-innodb-heavy-4g.cnf my-large.cnf my-medium.cnf my-small.cnf
```

用户可以根据实际需求选择不同的配置文件 cp 到/etc 下，改名为 my.cnf，并根据实际需要做一些配置的改动。

6. 在 linux 平台下，可以采用如下命令查看 mysql 服务的状态：

```
[root@localhost bin]# netstat -lntup
```

7. 在 linux 平台上启动和关闭 mysql 有两种方法，一种是通过命令行方式启动和关闭，另外一种是通过服务的方式启动和关闭（适用于 rpm 包安装方式）。（启动、关闭、命令行、服务）

- 在命令行方式下，启动 mysql 服务命令如下：

```
[root@localhost bin]# cd /usr/bin
[root@localhost bin]# ./mysqld_safe &
[1] 23013
```

```
[root@localhost bin]# starting mysqld daemon with databases from /var/lib/mysql
```

关闭 mysql 服务命令如下：

```
[root@localhost bin]# mysqladmin -uroot shutdown
```

- 如果 mysql 是用 rpm 包安装的，则启动 mysql 服务过程如下：

```
[root@localhost zzx]# service mysql start
starting mysql[ ok ]
```

如果在启动状态，需要重启服务，可以用以下命令直接重启，而不需要先关闭再启动：

```
[root@localhost mysql]# service mysql restart
shutting down mysql..[ ok ]
starting mysql[ o k ]
```

关闭 mysql 服务过程如下：

```
[root@localhost bin]# service mysql stop
```

第 2 章 sql 基础

1. sql 语句以分号结束，不区分大小写。
2. sql 语句主要可以划分为以下 3 个类别。（d 定 m 操 c 控）

- ddl 语句：数据定义语言，这些语句定义了不同的数据段、数据库、表、列、索引等数据库对象的定义。常用的语句关键字主要包括 create、drop、alter 等。
- dml 语句：数据操纵语句，用于添加、删除、更新和查询数据库记录，并检查数据完整性，常用的语句关键字主要包括 insert、delete、udpate 和 select 等。
- dcl 语句：数据控制语句，用于控制不同数据段直接的许可和访问级别的语句。这些语句定义了数据库、表、字段、用户的访问权限和安全级别。主要的语句关键字包括 grant、revoke 等。

3. ddl 是数据定义语言的缩写，简单来说，就是对数据库内部的对象进行创建、删除、修改的操作语言。它和 dml 语言的最大区别是 dml 只是对表内部数据的操作，而不涉及到表的定义、结构的修改，更不会涉及到其他对象。
4. create 命令用于创建数据库以及表：（create、创建数据库和表）

- 创建数据库：

```
create database dbname
```

- 创建表：

```
create table tablename (column_name_1 column_type_1 constraints,
column_name_2 column_type_2 constraints ， ……column_name_n column_type_n
constraints)
```

5. use 命令用于选择相应的数据库：（use、选择数据库）

- 选择数据库：

```
use dbname
```

6. show 命令用于显示数据库内容或者表内容。（show、显示数据库和表）

- 显示数据库内容：

```
mysql> show databases;
+--------------------------+
| database                    |
+--------------------------+
| information_schema |
| cluster                          |
| mysql                           |
| test                               |
| test1                             |
+--------------------------+
5 rows in set (0.00 sec)
```

- 显示表内容：

```
mysql> show tables;
empty set (0.00 sec)
```

7. 删除数据库：（drop database、删除数据库）

```
drop database dbname;
```

例如，要删除 test1 数据库可以使用以下语句：

```
mysql> drop database test1;
```

8. 在数据库中创建一张表的基本语法如下：

```
create table tablename (column_name_1 column_type_1 constraints,
column_name_2 column_type_2 constraints ,  ……column_name_n column_type_n
constraints)
```

column_name 是列的名字，column_type 是列的数据类型，contraints 是这个列的约

束条件。　　（列名字后接列数据类型）

例如，创建一个名称为 emp 的表。表中包括 3 个字段，ename（姓名），hiredate（雇用日期）、sal（薪水），字段类型分别为 varchar（10）、date、int（2）：

```
mysql> create table emp(ename varchar(10),hiredate date,sal
decimal(10,2),deptno int(2));
query ok, 0 rows affected (0.02 sec)
```

表创建完毕后，如果需要查看一下表的定义，可以使用如下命令：（desc、查看表定

义）

```
desc tablename
```

这个命令不能显示表的内容，但可以显示表的列名及相关信息。

例如，查看 emp 表，将输出以下信息：

```
mysql> desc emp;
+----------+-----------------+--------+---------+---------+-------+
| field    | type            | null   | key     | default | extra |
+----------+-----------------+--------+---------+---------+-------+
| ename    | varchar(10)     | yes    |         |         |       |
| hiredate | date            | yes    |         |         |       |
| sal      | decimal(10,2)   | yes    |         |         |       |
| deptno   | int(2)          | yes    |         |         |       |
+----------+-----------------+--------+---------+---------+-------+
4 rows in set (0.00 sec)
```

注意：desc 只能用于查看表定义，不能用于查看数据库定义。

9. 虽然 desc 命令可以查看表定义，但是其输出的信息还是不够全面，为了查看更全面的表定义信息，有时就需要通过查看创建表的 sql 语句来得到（查看 sql 语句），可以使用如下命令实现：（更全面的定义、去括加 show）

```
mysql> show create table emp \g;
*************************** 1. row ***************************
table: emp
create table: create table `emp` (
`ename` varchar(20) default null,
`hiredate` date default null,
`sal` decimal(10,2) default null,
`deptno` int(2) default null,
key `idx_emp_ename` (`ename`)
) engine=innodb default charset=gbk
1 row in set (0.02 sec)
error:
no query specified
mysql>
```

除了可以看到表定义以外，还可以看到表的 engine（存储引擎）和 charset（字符集）等信息。“\g”选项的含义是使得记录能够按照字段竖着排列，对于内容比较长的记录更易于显示。（g、竖着排列）

10. 删除表（drop table、删除表）

```
drop table tablename
```

例如，要删除数据库 emp 可以使用以下命令：

```
mysql> drop table emp;
query ok, 0 rows affected (0.00 sec)
```

11. 修改表 （alter table、修改表）

- 修改表类型，语法如下： （modify、add column、drop column、change 字段、rename 表）

```
alter table tablename modify [column] column_definition [first | after col_name]
```

例如，修改表 emp 的 ename 字段定义，将 varchar(10)改为 varchar(20)：

```
mysql> alter table emp modify ename varchar(20);
```

- 增加表字段，语法如下：

```
alter table tablename add [column] column_definition [first | after col_name]
```

例如，表 emp 上新增加字段 age，类型为 int(3)：

```
mysql> alter table emp add column age int(3);
```

- 删除表字段，语法如下：

```
alter table tablename drop [column] col_name
```

例如，将字段 age 删除掉：

```
mysql> alter table emp drop column age;
```

- 字段改名，语法如下：

```
alter table tablename change [column] old_col_name column_definition [first|after col_name]
```

例如，将 age 改名为 age1，同时修改字段类型为 int(4)：

```
mysql> alter table emp change age age1 int(4) ;
```

- 修改字段排列顺序。前面介绍的的字段增加和修改语法（add/cnahge/modify）中，都有一个可选项 first|after column_name，这个选项可以用来修改字段在表中的位置，默认 add 增加的新字段是加在表的最后位置，而 change/modify 默认都不会改变字段的位置。（first、after）

  例如，将新增的字段 birth date 加在 ename 之后：

```
mysql> alter table emp add birth date after ename;
```

修改字段 age，将它放在最前面：

```
mysql> alter table emp modify age int(3) first;
```

- 表改名，语法如下：　　（rename、改名）

```
alter table tablename rename [to] new_tablename
```

例如，将表 emp 改名为 emp1，命令如下：

```
mysql> alter table emp rename emp1;
```

12. dml 操作是指对数据库中表记录的操作，主要包括表记录的插入（insert）、更新（update）、删除（delete）和查询（select），是开发人员日常使用最频繁的操作。（操、插、更、删、查）

13. insert 插入记录：　　（insert into…values…）

```
insert into tablename (field1,field2,……fieldn) values(value1,value2,……
valuesn);
```

例如，向表 emp 中插入以下记录：ename 为 zzx1，hiredate 为 2000-01-01，sal 为 2000，deptno 为 1，命令执行如下：

```
mysql> insert into emp (ename,hiredate,sal,deptno) values('zzx1','2000-01-
01','2000',1);
query ok, 1 row affected (0.00 sec)
```

也可以不用指定字段名称，但是 values 后面的顺序应该和字段的排列顺序一致：

```
mysql> insert into emp values('lisa','2003-02-01','3000',2);
query ok, 1 row affected (0.00 sec)
```

- 在 mysql 中，insert 语句还有一个很好的特性，可以一次性插入多条记录，语法如下：　　（多条逗）

```
insert into tablename (field1, field2,……fieldn)
values
(record1_value1, record1_value2,……record1_valuesn),
(record2_value1, record2_value2,……record2_valuesn),
……
(recordn_value1, recordn_value2,……recordn_valuesn)
;
```

下面的例子中，对表 dept 一次插入两条记录：

```
mysql> insert into dept values(5,'dept5'),(6,'dept6');
```

14. update 更新表的记录值：　　（update…set…where）

```
update tablename set field1=value1, field2.=value2, ……fieldn=valuen [where
condition]
```

例如，将表 emp 中 ename 为“lisa”的薪水（sal）从 3000 更改为 4000：

```
mysql> update emp set sal=4000 where ename='lisa';
```

- 在 mysql 中，update 命令可以同时更新多个表中数据，语法如下：（多个、逗号）

```
update t1,t2…tn set t1.field1=expr1,tn.fieldn=exprn [where condition]
```

在下例中，同时更新表 emp 中的字段 sal 和表 dept 中的字段 deptname：（别名）

```
mysql> update emp a,dept b set a.sal=a.sal*b.deptno,b.deptname=a.ename where a.deptno=b.deptno;
```

注意： emp a 表示的应该是定义 emp 的别名 a。

15. delete 删除记录： （记录 delete from…where…）

```
delete from tablename [where condition]
```

例如，在 emp 中将 ename 为‘dony’的记录全部删除，命令如下：

```
mysql> delete from emp where ename='dony';
query ok, 1 row affected (0.00 sec)
```

- 在 mysql 中可以一次删除多个表的数据，语法如下：

```
delete t1,t2…tn from t1,t2…tn [where condition]
```

如果 from 后面的表名用别名，则 delete 后面的也要用相应的别名，否则会提示语法错误。在下例中，将表 emp 和 dept 中 deptno 为 3 的记录同时都删除：

```
mysql> delete a,b from emp a,dept b where a.deptno=b.deptno and a.deptno=3;
query ok, 2 rows affected (0.04 sec)
```

16. select 查询记录：

```
select * from tablename [where condition]
```

- 查询不重复的记录。有时需要将表中的记录去掉重复后显示出来，可以用 distinct 关键字来实现： （distinct 不重复）

```
mysql> select * from emp;
+--------+--------------+----------+--------+
| ename |   hiredate      |   sal       | deptno  |
+--------+--------------+----------+--------+
| zzx       | 2000-01-01 | 2000.00 | 1         |
| lisa       | 2003-02-01 | 4000.00 | 2         |
| bjguan | 2004-04-02 | 5000.00 | 1         |
+--------+--------------+---------+---------+
3 rows in set (0.00 sec)

mysql> select * from emp where deptno=1;
+--------+--------------+----------+--------+
| ename | hiredate        | sal         | deptno |
+--------+--------------+----------+--------+
| zzx       | 2000-01-01 | 2000.00 |   1        |
| bjguan | 2004-04-02 | 5000.00 |   1        |
+--------+--------------+----------+---------+
2 rows in set (0.00 sec)
```

当 where 后面的条件是字符型时，需要加上单引号：（where、字符、单引）

```
mysql> select host,password,user from user where user='root';
```

- 条件查询。

```
mysql> select * from emp where deptno=1;
```

结果集中将符合条件的记录列出来。上面的例子中，where 后面的条件是一个字段的‘=’比较，除了‘=’外，还可以使用>、<、>=、<=、!=等比较运算符；多个条件之间还可以使

用 or、and 等逻辑运算符进行多条件联合查询。

以下是一个使用多字段条件查询的例子：

```
mysql> select * from emp where deptno=1 and sal<3000;
```

- order by 排序和限制。 （order by、排序）

```
select * from tablename [where condition] [order by field1 [desc|asc] , field2 [desc|asc], ……fieldn [desc|asc]]
```

其中，desc 和 asc 是排序顺序关键字，desc 表示按照字段进行降序排列，asc 则表示升序排列，如果不写此关键字默认是升序排列。 （asc 升 desc 降、order by sal desc）

例如，把 emp 表中的记录按照工资高低进行显示：

```
mysql> select * from emp order by sal;
```

对于 deptno 相同的前两条记录，如果要按照工资由高到低排序，可以使用以下命令：

```
mysql> select * from emp order by deptno,sal desc;
```

  - 对于排序后的记录，如果希望只显示一部分，而不是全部，这时，就可以使用 limit 关键字来实现，limit 的语法如下： （limit、一部分）

```
select ……[limit offset_start,row_count]
```

其中 offset_start 表示记录的起始偏移量，row_count 表示显示的行数。在默认情况下，起始偏移量为 0，只需要写记录行数就可以，这时候，显示的实际就是前 n 条记录，例如，显示 emp 表中按照 sal 排序后的前 3 条记录： （前 3 条、limit 3、order by sal limit 3）

```
mysql> select * from emp order by sal limit 3;
```

- 聚合：很多情况下，我们需要进行一些汇总操作，比如统计整个公司的人数或者统计每个部门的人数，这个时就要用到 sql 的聚合操作。聚合操作的语法如下：

```
select [field1,field2,……fieldn] fun_name
from tablename
[where where_contition]
[group by field1,field2,……fieldn
[with rollup]]
[having where_contition]
```

对其参数进行以下说明。 （select、fun_name、聚合）

- fun_name 表示要做的聚合操作，也就是聚合函数，常用的有 sum（求和）、count(*)（记录数）、max（最大值）、min（最小值）。　　（聚、和、数、大、小）
- group by 关键字表示要进行分类聚合的字段，比如要按照部门分类统计员工数量，部门就应该写在 group by 后面。　　（group by 分组）
- with rollup 是可选语法，表明是否对分类聚合后的结果进行再汇总。
- having 关键字表示对分类后的结果再进行条件的过滤。

➢ 注意：having 和 where 的区别在于 having 是对聚合后的结果进行条件的过滤，而 where 是在聚合前就对记录进行过滤，如果逻辑允许，我们尽可能用 where 先过滤记录，这样因为结果集减小，将对聚合的效率大大提高，最后再根据逻辑看是否用 having 进行再过滤。　　（过滤 where 前 having 后）　　（w 前 h 后）

例如，要 emp 表中统计公司的总人数：

```
mysql> select count(1) from emp;
```

在此基础上，要统计各个部门的人数：

```
mysql> select deptno,count(1) from emp group by deptno;
```

更细一些，既要统计各部门人数，又要统计总人数：

```
mysql> select deptno,count(1) from emp group by deptno with rollup;
```

统计人数大于 1 人的部门：

```
mysql> select deptno,count(1) from emp group by deptno having count(1)>1;
```

- 子查询：某些情况下，当我们查询的时候，需要的条件是另外一个 select 语句的结果，这个时候，就要用到子查询。用于子查询的关键字主要包括 in、not in、=、!=、exists、not exists 等。例如，从 emp 表中查询出所有部门在 dept 表中的所有记录：

```
mysql> select * from emp where deptno in(select deptno from dept);
```

如果子查询记录数唯一，还可以用=代替 in　　（select ...from...where ... in)

- 记录联合：我们经常会碰到这样的应用，将两个表的数据按照一定的查询条件查询出来后，将结果合并到一起显示出来，这个时候，就需要用 union 和 union all 关键字来实现这样的功能，具体语法如下：

```
select * from t1
union|union all
select * from t2
……
union|union all
select * from tn;
```

union 和 union all 的主要区别是 union all 是把结果集直接合并在一起，而 union 是将 union all 后的结果进行一次 distinct，去除重复记录后的结果。来看下面例子，将 emp 和 dept 表中的部门编号的集合显示出来：　　（union、all 直接、非去除重复）

```
mysql> select deptno from emp
-> union all
-> select deptno from dept;
```

17. 连接可以分为以下几种：

(1) inner join（内连接）：获取两个表中字段匹配关系的记录。（内连接、两个表、匹配的）

```
mysql> select a.runoob_id, a.runoob_author, b.runoob_count from runoob_tbl a
inner join tcount_tbl b on a.runoob_author = b.runoob_author;
+-------------+-----------------+----------------+
| a.runoob_id | a.runoob_author | b.runoob_count |
+-------------+-----------------+----------------+
| 1           | 菜鸟教程        | 10             |
| 2           | 菜鸟教程        | 10             |
| 3           | runoob.com      | 20             |
| 4           | runoob.com      | 20             |
+-------------+-----------------+----------------+
4 rows in set (0.00 sec)
```

inner join 可以省略 inner。（内连接 、select...from...inner join...on...）

(2) left join（左连接）：获取左表所有记录，即使右表没有对应匹配的记录。（左连接、左边所有）

```
mysql> select ename,deptname from emp left join dept on emp.deptno=dept.deptno;
```

emp 和 dept 是要联结的表。（select...from...left join...on）

(3) right join（右连接）： 获取右表所有记录，即使左表没有对应匹配的记录。（右连接、右边所有）

18. dcl 语句主要是 dba 用来管理系统中的对象权限时所使用。

19. where 子句中可以使用等号 = 来设定获取数据的条件，也可以使用 like 子句：（where、like 子句）

```
mysql> select * from runoob_tbl  where runoob_author like '%com';
```

20. 为了处理查询值为 Null 的情况，MySQL 提供了三大运算符:（处理 NULL、IS NULL、IS NOT NULL、<=>）

- IS NULL: 当列的值是 NULL,此运算符返回 true。
- IS NOT NULL: 当列的值不为 NULL, 运算符返回 true。
- <=>: 比较操作符（不同于=运算符），当比较的的两个值为 NULL 时返回 true。

```
mysql> select * from runoob_test_tbl where runoob_count != NULL;
```

21. 临时表只在当前连接可见，当关闭连接时，Mysql 会自动删除表并释放所有空间。使用 create temporary table 来创建临时表：（临时表、create temporary table）

```
mysql> create temporary table salessummary (
    -> product_name varchar(50) not null
    -> , total_sales decimal(12,2) not null default 0.00
    -> , avg_unit_price decimal(7,2) not null default 0.00
    -> , total_units_sold int unsigned not null default 0
);
query ok, 0 rows affected (0.00 sec)
```

使用 DROP TABLE 删除临时表：

```
drop table salessummary
```

22. 子句顺序：（where、group by、having、order by）（w、g、h、o）

表10-2 SELECT子句及其顺序

子 句	说 明	是否必须使用
SELECT	要返回的列或表达式	是
FROM	从中检索数据的表	仅在从表选择数据时使用
WHERE	行级过滤	否
GROUP BY	分组说明	仅在按组计算聚集时使用
HAVING	组级过滤	否
ORDER BY	输出排序顺序	否

23. 如果不知道 mysql 安装后自带的帮助文档能够提供些什么，可以用“? contents”命令来显示所有可供查询的的分类：（文档、?contents）

```
? contents
```

对于列出的分类，可以使用“? 类别名称”的方式针对用户感兴趣的内容做进一步的查看。例如，想看看 mysql 中都支持哪些数据类型，可以执行“? data types”命令：

```
mysql> ? data types
```

24. 在实际应用当中，如果需要快速查阅某项语法时，可以使用关键字进行快速查询。例如，想知道 show 命令都能看些什么东西，可以用如下命令：

```
mysql> ? show
```

第 3 章 mysql 支持的数据类型

1. mysql 中的数值类型

整数类型	字节	最小值	最大值
TINYINT	1	有符号-128 无符号 0	有符号 127 无符号 255
SMALLINT	2	有符号-32768 无符号 0	有符号 32767 无符号 65535
MEDIUMINT	3	有符号-8388608 无符号 0	有符号 8388607 无符号 1677215
INT、INTEGER	4	有符号-2147483648 无符号 0	有符号 2147483647 无符号 4294967295
BIGINT	8	有符号-9223372036854775808 无符号 0	有符号 9223372036854775807 无符号 18446744073709551615
浮点数类型	**字节**	**最小值**	**最大值**
FLOAT	4	±1.175494351E-38	±3.402823466E+38
DOUBLE	8	±2.2250738585072014E-308	±1.7976931348623157E+308
定点数类型	**字节**	**描述**	
DEC(M,D), DECIMAL(M,D)	M+2	最大取值范围与 DOUBLE 相同，给定 DECIMAL 的有效取值范围由 M 和 D 决定	
位类型	**字节**	**最小值**	**最大值**
BIT(M)	1～8	BIT(1)	BIT(64)

在整数类型中，按照取值范围和存储方式不同，分为 tinyint、smallint、mediumint、int、bigint 这 5 个类型。对于整型数据，mysql 还支持在类型名称后面的小括号内指定显示宽度，例如 int(5)表示当数值宽度小于 5 位的时候在数字前面填满宽度，如果不显示指定宽度则默认为 int(11)。一般配合 zerofill 使用，顾名思义，zerofill 就是用“0”填充的意思，也就是在数字位数不够的空间用字符“0”填满。（括填宽）

例：创建表 t1，有 id1 和 id2 两个字段，指定其数值宽度分别为 int 和 int(5)。

```
mysql> create table t1 (id1 int,id2 int(5));
```

2. 设置了宽度限制后，如果插入大于宽度限制的值，则不会对插入的数据有任何影响，还是按照类型的实际精度进行保存。　　（大于、实际精度）

3. 对于小数的表示，mysql 分为两种方式：浮点数和定点数。浮点数包括 float（单精度）和 double（双精度），而定点数则只有 decimal 一种表示。定点数在 mysql 内部以字符串形式存放，比浮点数更精确，适合用来表示货币等精度高的数据。浮点数和定点数都可以用类型名称后加“(m,d)”的方式来进行表示，“(m,d)”表示该值一共显示 m 位数字（整数位+小数位），其中 d 位位于小数点后面，m 和 d 又称为精度和标度。创建测试表，分别将 id1、id2、id3 字段设置为 float(5,2)、double(5,2)、decimal(5,2)。　　（浮点数、定点数 decimal、定点货币）

```
create table `t1` (
`id1` float(5,2) default null,
`id2` double(5,2) default null,　　（整数位、小数位、default）
`id3` decimal(5,2) default null
)
```

4. 浮点数如果不写精度和标度，则会按照实际精度值显示，如果有精度和标度，则会自动将四舍五入后的结果插入，系统不会报错；定点数如果不写精度和标度，则按照默认值 decimal(10,0)来进行操作，并且如果数据超越了精度和标度值，系统则会报错。（不写精度，浮点插入，定点报错）

5. 数据插入 bit 类型字段时，首先转换为二进制，如果位数允许，将成功插入；如果位数小于实际定义的位数，则插入失败，下面的例子中，在 t2 表插入数字 2，因为它的二进制码是“10”，而 id 的定义是 bit(1)，将无法进行插入：（数据插入、位数小于实际、失败）

```
mysql> insert into t2 values(2);
query ok, 1 row affected, 1 warning (0.00 sec)

mysql> show warnings;
+-------------+---------
+-----------------------------------------------------------+
| level  --      | code    | message
|
+-------------+--------
+-------------------------------------------------------------+
| warning      | 1264    | out of range value adjusted for column 'id' at row 1 |
+-------------+--------
+-------------------------------------------------------------+
1 row in set (0.01 sec)
```

6. show warnings; 显示最后一个执行的语句所产生的错误、警告和通知 （show warnings、show errors）

   show errors; 只显示最后一个执行语句所产生的错误

7. mysql 中有多种数据类型可以用于日期和时间的表示，不同的版本可能有所差异，表 3-2 中列出了 mysql 5.0 中所支持的日期和时间类型。
（date、time、year、datetime、timestamp）

表 3-2　　MySQL 中的日期和时间类型

日期和时间类型	字节	最小值	最大值
DATE	4	1000-01-01	9999-12-31
DATETIME	8	1000-01-01 00:00:00	9999-12-31 23:59:59
TIMESTAMP	4	19700101080001	2038 年的某个时刻
TIME	3	-838:59:59	838:59:59
YEAR	1	1901	2155

创建表 t，字段分别为 date、time、datetime 三种日期类型：

```
mysql> create table t (d date,t time,dt datetime);
query ok, 0 rows affected (0.01 sec)

mysql> desc t;
+--------+----------+--------+-------+-----------+--------+
| field    | type      | null  | key  | default    | extra  |
+--------+----------+------  +-------+------------+--------+
| d         | date          | yes    |         | null |            |
| t          |  time         | yes    |         | null |            |
| dt        | datetime | yes    |         | null |            |
+-------+-----------+---------+-----+---------+-------+
3 rows in set (0.01 sec)
```

用 now()函数插入当前日期：

```
mysql> insert into t values(now(),now(),now());
```

8. 时间：

- utc 时间也就是本初子午线时间，是时间协调时间，可以认为 utc 时间与等同于 gmt 时间，utc + 时区差 = 本地时间。　　（本初子午线、协调时间）
- 本地时间：本地时间所在的时间。
- unix 时间戳：在计算机中看到的 utc 时间都是从（1970 年 01 月 01 日 0:00:00)开始计算秒数的。所看到的 utc 时间那就是从 1970 年这个时间点起到具体时间共有多少秒。

9. 创建测试表 t，字段 id1 为 timestamp 类型：

```
mysql> create table t (id1 timestamp);
query ok, 0 rows affected (0.03 sec)

mysql> desc t;
+---------+-----------+-------+------+-------------------------------
+-------+
|  field    | type         |  null   |  key | default
```

```
| extra |
+--------+-------------+-------+------+---------------------------------
+-------+
|  id2      | timestamp |  yes |         | current_timestamp  |           |
+--------+-------------+-------+------+---------------------------------
+-------+
1 row in set (0.00 sec)
```

10. 通过一个例子，说明如何采用不同的格式将日期“2007-9-3 12:10:10”插入到 datetime 列中。（insert into t6 values）

```
mysql> create table t6(dt datetime);
query ok, 0 rows affected (0.03 sec)

mysql> insert into t6 values('2007-9-3 12:10:10');
query ok, 1 row affected (0.00 sec)

mysql> insert into t6 values('2007/9/3 12+10+10');
query ok, 1 row affected (0.00 sec)

mysql> insert into t6 values('20070903121010');
query ok, 1 row affected (0.01 sec)

mysql> insert into t6 values(20070903121010);
query ok, 1 row affected (0.00 sec)

mysql> select * from t6;
+-------------------------+
| dt                                |
+-------------------------+
| 2007-09-03 12:10:10 |
| 2007-09-03 12:10:10 |
| 2007-09-03 12:10:10 |
| 2007-09-03 12:10:10 |
+-------------------------+
4 rows in set (0.00 sec)
```

11. mysql 中提供了多种对字符数据的存储类型，不同的版本可能有所差异。以 5.0 版本为例，mysql 包括
char、varchar、binary、varbinary、blob、text、enum 和 set 等多种字符串类型。（char、varchar、binary、varbinary、blob（二进制）、text）

表 3-4 MySQL 中的字符类型

字符串类型	字节	描述及存储需求
CHAR（M）	M	M 为 0～255 之间的整数
VARCHAR（M）		M 为 0～65535 之间的整数，值的长度+1 个字节
TINYBLOB		允许长度 0～255 字节，值的长度+1 个字节
BLOB		允许长度 0～65535 字节，值的长度+2 个字节
MEDIUMBLOB		允许长度 0～167772150 字节，值的长度+3 个字节
LONGBLOB		允许长度 0～4294967295 字节，值的长度+4 个字节
TINYTEXT		允许长度 0～255 字节，值的长度+2 个字节
TEXT		允许长度 0～65535 字节，值的长度+2 个字节
MEDIUMTEXT		允许长度 0～167772150 字节，值的长度+3 个字节
LONGTEXT		允许长度 0～4294967295 字节，值的长度+4 个字节
VARBINARY（M）		允许长度 0～M 个字节的变长字节字符串，值的长度+1 个字节
BINARY（M）	M	允许长度 0～M 个字节的定长字节字符串

12. char 和 varchar 很类似，都用来保存 mysql 中较短的字符串。二者的主要区别在于存储方式的不同：char 列的长度固定为创建表时声明的长度，长度可以为从 0～255 的任何值；而 varchar 列中的值为可变长字符串，长度可以指定为 0～255（5.0.3 以前）或者 65535（5.0.3 以后）之间的值。在检索的时候，char 列删除了尾部的空格，而 varchar 则保留这些空格。（varchar 可变、保留空格）

```
mysql> create table vc (v varchar(4), c char(4));
mysql> insert into vc values ('ab  ', 'ab  ');
```

13. enum 中文名称叫枚举类型，它的值范围需要在创建表时通过枚举方式显式指定，对 1～255 个成员的枚举需要 1 个字节存储；对于 255～65535 个成员，需要 2 个字节存储。最多允许有 65535 个成员。（enum、枚举类型、gender enum('m','f')）

```
mysql> create table t (gender enum('m','f'));
mysql> insert into t values('m'),('1'),('f'),(null);

mysql> select * from t;
+--------+
| gender |
+--------+
| m      |
```

```
|  m      |
|  f      |
| null |
+--------+
4 rows in set (0.01 sec)
```

对于插入不在 enum 指定范围内的值时，并没有返回警告，而是插入了 enum('m','f')的第一值'm'。（不在指定范围、第一值）

14. set 和 enum 类型非常类似，也是一个字符串对象，里面可以包含 0～64 个成员。根据成员的不同，存储上也有所不同。

- 1～8 成员的集合，占 1 个字节。
- 9～16 成员的集合，占 2 个字节。
- 17～24 成员的集合，占 3 个字节。
- 25～32 成员的集合，占 4 个字节。
- 33～64 成员的集合，占 8 个字节。

set 和 enum 除了存储之外，最主要的区别在于 set 类型一次可以选取多个成员，而 enum 则只能选一个。（set 多个、enum 一个）

```
create table t (col set  ('a','b','c','d') ;
insert into t values('a,b'),('a,d,a'),('a,b'),('a,c'),('a');
```

第 4 章 mysql 中的运算符

1. mysql 支持的算术运算符包括加、减、乘、除和模运算。它们是最常使用、最简单的一类运算符。下例中简单地描述了这几种运算符的使用方法：（加、减、乘、除、模）

```
mysql> select 0.1+ 0.3333 ,0.1-0.3333, 0.1*0.3333, 1/2,1%2;
+--------------+-------------+--------------+----------+------+
| 0.1+ 0.3333  |  0.1-0.3333 |  0.1*0.3333 |  1/2       | 1%2 |
+--------------+-------------+--------------+----------+------+
| 0.4333           | -0.2333      |  0.03333       | 0.5000  | 1      |
+----------------+-------------+--------------+----------+-----+
1 row in set (0.00 sec)
```

2. 比较运算符：当使用 select 语句进行查询时，mysql 允许用户对表达式的左边操作数和右边操作数进行比较，比较结果为真，则返回 1，为假则返回 0，比较结果不确定则返回 null。

表 4-2　　MySQL 支持的比较运算符

运算符	作用
=	等于
<>或!=	不等于
<=>	NULL 安全的等于(NULL-safe)
<	小于
<=	小于等于
>	大于
>=	大于等于
BETWEEN	存在与指定范围
IN	存在于指定集合
IS NULL	为 NULL
IS NOT NULL	不为 NULL
LIKE	通配符匹配
REGEXP 或 RLIKE	正则表达式匹配

```
mysql> select 'a'>='b','abc'>='a' ,1>=0 ,1>=1;
mysql> select 10 between 10 and 20, 9 between 10 and 20;
mysql> select 1 in (1,2,3) , 't' in ('t','a','b','1','e'),0 in (1,2);
mysql> select 0 is null, null is null;
mysql> select 123456 like '123%',123456 like '%123%',123456 like '%321%';
```

3. 逻辑运算符又称为布尔运算符，用来确认表达式的真和假。

表 4-3　　MySQL 中的逻辑运算符

运算符	作用
NOT 或!	逻辑非
AND 或&&	逻辑与
OR 或 ||	逻辑或

```
mysql> select not 0, not 1, not null ;
mysql> select (1 and 1),(0 and 1) ,(3 and 1 ) ,(1 and null);
mysql> select (1 or 0) ,(0 or 0),(1 or null) ,(1 or 1),(null or null);
```

```
mysql> select 1 xor 1 ,0 xor 0,1 xor 0,0 xor 1,null xor 1;
```

4. 位运算是将给定的操作数转化为二进制后，对各个操作数每一位都进行指定的逻辑运算，得到的二进制结果转换为十进制数后就是位运算的结果。（运算后要转换为十进制）

表 4-4　　MySQL 支持的位运算符

运算符	作用
&	位与（位 AND）
|	位或 （位 OR ）
^	位异或（位 XOR）
~	位取反
>>	位右移
<<	位左移

```
mysql> select 2&3;
mysql> select 2&3&4;
mysql> select ~1 ,~ 18446744073709551614;
```

第 5 章 常用函数

1. 所有函数都配合 select 语句使用：（函数、select xxx()）

```
select xxx();
```

2. 字符串函数

函数	功能
CANCAT(S1,S2,...Sn)	连接 S1,S2,...Sn 为一个字符串
INSERT(str,x,y,instr)	将字符串 str 从第 x 位置开始，y 个字符长的子串替换为字符串 instr
LOWER(str)	将字符串 str 中所有字符变为小写
UPPER(str)	将字符串 str 中所有字符变为大写
LEFT(str ,x)	返回字符串 str 最左边的 x 个字符
RIGHT(str,x)	返回字符串 str 最右边的 x 个字符
LPAD(str,n ,pad)	用字符串 pad 对 str 最左边进行填充，直到长度为 n 个字符长度
RPAD(str,n,pad)	用字符串 pad 对 str 最右边进行填充，直到长度为 n 个字符长度
LTRIM(str)	去掉字符串 str 左侧的空格
RTRIM(str)	去掉字符串 str 行尾的空格
REPEAT(str,x)	返回 str 重复 x 次的结果
REPLACE(str,a,b)	用字符串 b 替换字符串 str 中所有出现的字符串 a
STRCMP(s1,s2)	比较字符串 s1 和 s2
TRIM(str)	去掉字符串行尾和行头的空格
SUBSTRING(str,x,y)	返回从字符串 str x 位置起 y 个字符长度的字串

用法以 cancat(s1,s2,…sn)函数为例：

```
mysql> select concat('aaa','bbb','ccc') ,concat('aaa',null);
+--------------------------+--------------------+
| concat('aaa','bbb','ccc') | concat('aaa',null) |
+--------------------------+--------------------+
| aaabbbccc                 | null               |
+--------------------------+--------------------+
1 row in set (0.05 sec)
```

3. 数值函数

函数	功能
ABS(x)	返回 x 的绝对值
CEIL(x)	返回大于 x 的最大整数值
FLOOR(x)	返回小于 x 的最大整数值
MOD(x，y)	返回 x/y 的模
RAND()	返回 0 到 1 内的随机值
ROUND(x,y)	返回参数 x 的四舍五入的有 y 位小数的值
TRUNCATE(x,y)	返回数字 x 截断为 y 位小数的结果

4. 日期和时间函数

函数	功能
CURDATE()	返回当前日期
CURTIME()	返回当前时间
NOW()	返回当前的日期和时间
UNIX_TIMESTAMP(date)	返回日期 date 的 UNIX 时间戳
FROM_UNIXTIME	返回 UNIX 时间戳的日期值
WEEK(date)	返回日期 date 为一年中的第几周
YEAR(date)	返回日期 date 的年份
HOUR(time)	返回 time 的小时值
MINUTE(time)	返回 time 的分钟值
MONTHNAME(date)	返回 date 的月份名
DATE_FORMAT(date,fmt)	返回按字符串 fmt 格式化日期 date 值
DATE_ADD(date,INTERVAL expr type)	返回一个日期或时间值加上一个时间间隔的时间值
DATEDIFF(expr,expr2)	返回起始时间 expr 和结束时间 expr2 之间的天数

5. 流程函数（if、ifnull、case when...then...else...end）

函数	功能
IF(value,t f)	如果 value 是真，返回 t；否则返回 f
IFNULL(value1,value2)	如果 value1 不为空返回 value1，否则返回 value2
CASE WHEN [value1] THEN[result1]...ELSE[default]END	如果 value1 是真，返回 result1，否则返回 default
CASE [expr] WHEN [value1] THEN[result1]...ELSE[default]END	如果 expr 等于 value1，返回 result1，否则返回 default

下面的例子中模拟了对职员薪水进行分类，这里首先创建并初始化一个职员薪水表：

```
mysql> create table salary (userid int,salary decimal(9,2));
mysql> insert into salary values(1,1000),(2,2000), (3,3000),(4,4000),(5,5000),
(1,null);
mysql> select if(salary>2000,'high','low') from salary;
```

6. 其他常用函数

函数	功能
DATABASE()	返回当前数据库名
VERSION()	返回当前数据库版本
USER()	返回当前登录用户名
INET_ATON(IP)	返回 IP 地址的数字表示
INET_NTOA(num)	返回数字代表的 IP 地址
PASSWORD(str)	返回字符串 str 的加密版本
MD5()	返回字符串 str 的 MD5 值

第 6 章 图形化工具的使用

1. mysql administrator 是 mysql 为 4.0 以上版本数据库提供的可视化界面的 mysql 数据库管理控制台，可以方便地管理和操作 mysql 数据库。提供的功能包括启动关闭数据库、用户管理、参数配置、数据库对象管理、备份恢复管理等。
2. 管理控制台上的部分功能只能用于管理本地数据库，对于远程数据库这些选项将不可用。例如，服务控制、参数配置和日志查询等。
3. 连接管理可以用来查看当前活跃的数据库连接，这与 show processlist 命令的执行结果相同。如果当前连接的用户有 process 权限，那么可以查看全部的线程；如果当前连接的用户有 super 权限，那么该用户还可以通过单击窗口下面的“kill thread”按钮杀掉指定的线程，或者通过单击“kill user”按钮杀掉指定用户的全部线程。
4. phpmyadmin（简称 pma），是一个用 php 编写的、可以通过 web 控制和操作 mysql 数据库的工具。其最突出的特点是可以直接从 web 上去管理 mysql，不需要直接在 mysql 数据库服务器上进行维护。

第 7 章 表类型（存储引擎）的选择

1. mysql 中有一个存储引擎的概念，针对不同的存储需求可以选择最优的存储引擎。（不同的存储需求）
2. mysql 5.0 支持的存储引擎包括
   myisam、innodb、bdb、memory、merge、example、ndb
   cluster、archive、csv、blackhole、federated 等，其中 innodb 和 bdb 提供事务安全表，其他存储引擎都是非事务安全表。

3. 默认情况下，创建新表不指定表的存储引擎，则新表是默认存储引擎的，如果需要修改默认的存储引擎，则可以在参数文件中设置 default-table-type。查看当前的默认存储引擎，可以使用以下命令：　　（默认存储引擎、default-table-type）

```
mysql> show variables like 'table_type';
+-----------------+------------+
| variable_name |  value       |
+-----------------+------------+
| table_type       |  myisam  |
+-----------------+------------+
1 row in set (0.00 sec)
```

（table_type）

4. 可以通过下面两种方法查询当前数据库支持的存储引擎：

- 第一种方法：　　（查询存储引擎、show engines）

```
mysql> show engines \g
```

- 第二种方法：

```
mysql> show variables like 'have%';
```

5. 在创建新表的时候，可以通过增加 engine 关键字设置新建表的存储引擎：（存储引擎、engine=、default charset=）

```
create table ai (
i bigint(20) not null auto_increment,
primary key (i)
) engine=myisam default charset=gbk;
```

auto_increment：自动增长

也可以使用 alter table 语句，将一个已经存在的表修改成其他的存储引擎。下面的例子介绍了如何将表 ai 从 myisam 存储引擎修改成 innodb 存储引擎：（alter table…engine=…）

```
mysql> alter table ai engine = innodb;
```

6. myisam 是 mysql 的默认存储引擎。myisam 不支持事务、也不支持外键，其优势是访问的速度快，对事务完整性没有要求或者以 select、insert 为主的应用基本上都可以使用这个引擎来创建表。　　（myisam、不支持事务、速度快）
7. innodb 存储引擎提供了具有提交、回滚和崩溃恢复能力的事务安全。但是对比 myisam 的存储引擎，innodb 写的处理效率差一些并且会占用更多的磁盘空间以保留数据和索引。　　（innodb 事务安全、更多的磁盘空间）
8. 主键是能确定一条记录的唯一标识，比如，身份证号码就是主键。外键用于与另一张表的关联，是能确定另一张表记录的字段，用于保持数据的一致性。比如，a 表中的

一个字段，是 b 表的主键，就可以是 a 表的外键。　　（主键唯一，外键关联）

9. memory 存储引擎使用存在内存中的内容来创建表。memory 类型的表访问非常得快，因为它的数据是放在内存中的，并且默认使用 hash 索引，但是一旦服务关闭，表中的数据就会丢失掉。　　（memory、默认 hash 索引）

```
mysql> create table tab_memory engine=memory
-> select city_id,city,country_id
-> from city group by city_id;
```

(1)　　给 memory 表创建索引的时候，可以指定使用 hash 索引还是 btree 索引：（指定、using hash）

```
mysql> create index mem_hash using hash on tab_memory (city_id) ;
```

10.　　create index 用于创建索引。语法：

```
create [unique|fulltext|spatial] index index_name
      [index_type]
      on tbl_name (index_col_name,...)
      [index_type]

index_col_name:
col_name [(length)] [asc | desc]

index_type:
using {btree | hash | rtree}
```

给 memory 表创建索引的时候，可以指定使用 hash 索引还是 btree 索引：

```
mysql> create index mem_hash using hash on tab_memory (city_id) ;
```

（create index…using…on…）

11.　　merge 存储引擎是一组 myisam 表的组合，这些 myisam 表必须结构完全相同，merge 表本身并没有数据，对 merge 类型的表可以进行查询、更新、删除的操作，这些操作实际上是对内部的实际的 myisam 表进行的。　（merge 组合、engine=merge union=）

创建 3 个测试表 payment_2006、payment_2007 和 payment_all，其中 payment_all 是前两个表的 merge 表：

```
mysql> create table payment_2006(
-> country_id smallint,
-> payment_date datetime,
-> amount decimal(15,2),
-> key idx_fk_country_id (country_id)
-> )engine=myisam;

mysql> create table payment_2007(
-> country_id smallint,
-> payment_date datetime,
```

```
-> amount decimal(15,2),
-> key idx_fk_country_id (country_id)
-> )engine=myisam;

mysql> create table payment_all(
-> country_id smallint,
-> payment_date datetime,
-> amount decimal(15,2),
-> index(country_id)
-> )engine=merge union=(payment_2006,payment_2007) insert_method=last;
```

payment_all 表中的数据是 payment_2006 和 payment_2007 表的记录合并后的结果集。

向 merge 表插入一条记录，由于 merge 表的定义是 insert_method=last，就会向最后一个表中插入记录，也就是 payment_2007。

12. 下面是常用存储引擎的适用环境：

- myisam：默认的 mysql 插件式存储引擎。如果应用是以读操作和插入操作为主，只有很少的更新和删除操作，并且对事务的完整性、并发性要求不是很高，那么选择这个存储引擎是非常适合的。myisam 是在 web、数据仓储和其他应用环境下最常使用的存储引擎之一。（读写、插入为主、myisam）
- innodb：用于事务处理应用程序，支持外键。如果应用对事务的完整性有比较高的要求，在并发条件下要求数据的一致性，数据操作除了插入和查询以外，还包括很多的更新、删除操作，那么 innodb 存储引擎应该是比较合适的选择。（大量的更新、删除、innodb）
- memory：将所有数据保存在 ram 中，在需要快速定位记录和其他类似数据的环境下，可提供极快的访问。memory 的缺陷是对表的大小有限制，太大的表无法 cache 在内存中，其次是要确保表的数据可以恢复，数据库异常终止后表中的数据是可以恢复的。memory 表通常用于更新不太频繁的小表，用以快速得到访问结果。
- merge：merge 表的优点在于可以突破对单个 myisam 表大小的限制，并且通过将不同的表分布在多个磁盘上，可以有效地改善 merge 表的访问效率。这对于诸如数据仓储等 vldb 环境十分适合。

第 8 章 选择合适的数据类型

1. char 属于固定长度的字符类型，而 varchar 属于可变长度的字符类型。
2. varchar：如果 mysql 运行在非“严格模式”，则超过列长度的值依然会按实际值保存；如果 mysql 运行在严格模式，超过列长度的值将不会保存，并且会出现错误提示。（varchar、非严格实际、严格错误）
3. 一般在保存少量字符串的时候，我们会选择 char 或者 varchar；而在保存较大文本时，通常会选择使用 text 或者 blob，二者之间的主要差别是 blob 能用来保存二进制数据，比如照片；而 text 只能保存字符数据，比如一篇文章或者日记。（blob 二进制、text 字符数据）

4. text 和 blob 中有分别包括 text、mediumtext、longtext 和 blob、mediumblob、longblob3 种不同的类型，它们之间的主要区别是存储文本长度不同和存储字节不同。（标准、中、大三种）
5. 删除 text 或者 blob 的数据后会留下很大的“空洞”，以后填入这些“空洞”的记录在插入的性能上会有影响。为了提高性能，建议定期使用 optimize table 功能对这类表进行碎片整理，避免因为“空洞”导致性能问题。　　（optimize、整理空洞）

```
optimize table t;                    # t 是数据库表
```

6. 在不必要的时候避免检索大型的 blob 或 text 值。把 blob 或 text 列分离到单独的表中。

第 9 章 字符集

1. 查看所有可用的字符集的命令是 show character set：（show character set、字符集）

```
mysql> show character set;
```

2. mysql 的字符集包括字符集（character）和校对规则（collation）两个概念。字符集是用来定义 mysql 存储字符串的方式，校对规则则是定义了比较字符串的方式。字符集和校对规则是一对多的关系，mysql 支持 30 多种字符集的 70 多种校对规则。　　（字符集、校对规则）
3. 每个字符集至少对应一个校对规则。可以用“show collation like '***';”命令或者查看 information_schema.collations。查看相关字符集的校对规则。（至少对应一个校对规则）

```
mysql> SHOW COLLATION LIKE 'gbk%';
+---------------+---------+----+---------+----------+---------+
| Collation     | Charset | Id | Default | Compiled | Sortlen |
+---------------+---------+----+---------+----------+---------+
| gbk_chinese_ci | gbk     | 28 | Yes     | Yes      | 1       |
| gbk_bin       | gbk     | 87 |         | Yes      | 1       |
+---------------+---------+----+---------+----------+---------+
2 rows in set (0.00 sec)
```

校对规则命名约定：它们以其相关的字符集名开始，通常包括一个语言名，并且以_ci（大小写不敏感）、_cs（大小写敏感）或_bin（二元，即比较是基于字符编码的值而与 language 无关）结束。　　（ci 不、cs 敏）　（i 不 s 敏）

4. 在 my.cnf 中设置以下语句：

```
[mysql]
default-character-set=gbk
```

这样服务器启动后，所有连接默认就是使用 gbk 字符集进行连接的。

第 10 章 索引的设计和使用

1. mysql 索引的建立对于 mysql 的高效运行是很重要的，索引可以大大提高 mysql 的检索速度。 （索引、检索速度）
2. 索引分单列索引和组合索引。单列索引，即一个索引只包含单个列，一个表可以有多个单列索引，但这不是组合索引。组合索引，即一个索包含多个列。（单列索引、组合索引）
3. myisam 和 innodb 存储引擎的表默认创建的都是 btree 索引。mysql 目前还不支持函数索引，但是支持前缀索引，即对索引字段的前 n 个字符创建索引。（默认、btree）
4. mysql 中还支持全文本（fulltext）索引，该索引可以用于全文搜索。（全文本索引）
5. 默认情况下，memory 存储引擎使用 hash 索引，但也支持 btree 索引。（hash、btree）
6. 索引在创建表的时候可以同时创建，也可以随时增加新的索引。创建新索引的语法为： （create index…on … ）

```
create [unique|fulltext|spatial] index index_name
    [using index_type]
    on tbl_name (index_col_name,...)

index_col_name:
col_name [(length)] [asc | desc]
```

例如，要为 city 表创建了 10 个字节的前缀索引，语法是：

```
mysql> create index cityname on city (city(10));
query ok, 600 rows affected (0.26 sec)
records: 600 duplicates: 0 warnings: 0
```

如果以 city 为条件进行查询，可以发现索引 cityname 被使用。

7. 创建表的时候直接指定索引：

```
create table mytable(
id int not null,
username varchar(16) not null,
index [indexname] (username(length))
);
```

8. 索引的删除语法为： （drop index…on…）

```
drop index index_name on tbl_name
```

例如，想要删除 city 表上的索引 cityname，可以操作如下：

```
mysql> drop index cityname on city;
query ok, 600 rows affected (0.23 sec)
records: 600 duplicates: 0 warnings: 0
```

9. 索引的设计可以遵循一些已有的原则，创建索引的时候请尽量考虑符合这些原则，便于提升索引的使用效率，更高效地使用索引。

- 搜索的索引列， 不一定是所要选择的列
- 使用惟一索引。考虑某列中值的分布。索引的列的基数越大，索引的效果越好。
- 使用短索引。
- 利用最左前缀。 在创建一个 n 列的索引时， 实际是创建了 mysql 可利用的 n 个索引。多列索引可起几个索引的作用，因为可利用索引中最左边的列集来匹配行。这样的列集称为最左前缀。
- 不要过度索引。

10. memory 存储引擎的表可以选择使用 btree 索引或者 hash 索引， 两种不同类型的索引各有其不同的适用范围。

第 11 章 视图

1. 视图（view）是一种虚拟存在的表，对于使用视图的用户来说基本上是透明的。视图并不存在数据库中实际存在，行和列数据来自定义视图的查询中使用的表，并且是在使用视图时动态生成的。 （视图不存在）
2. 视图的操作包括创建或者修改视图、删除视图，以及查看视图定义。
3. 创建视图需要有 create view 的权限，并且对于查询涉及的列有 select 权限。如果使用 create or replace 或者 alter 修改视图，那么还需要该视图的 drop 权限。

- 创建视图的语法为： （create view …as…）

```
create [or replace] [algorithm = {undefined | merge | temptable}]
view view_name [(column_list)]
as select_statement
[with [cascaded | local] check option]
```

例如,要创建了视图 staff_list_view,可以使用以下命令：

```
mysql> create or replace view staff_list_view as
-> select s.staff_id,s.first_name,s.last_name,a.address
-> from staff as s,address as a
-> where s.address_id = a.address_id ;
query ok, 0 rows affected (0.00 sec)
```

- 修改视图的语法为： （alter view …as…）

```
alter [algorithm = {undefined | merge | temptable}]
view view_name [(column_list)]
as select_statement
[with [cascaded | local] check option]
```

4. mysql 视图的定义有一些限制，例如，在 from 关键字后面不能包含子查询，这和其他数据库是不同的，如果视图是从其他数据库迁移过来的，那么可能需要因此做一些改动，可以将子查询的内容先定义成一个视图，然后对该视图再创建视图就可以实现类似的功能了。 （限制、不能包含子查询）
5. 用户可以一次删除一个或者多个视图，前提是必须有该视图的 drop 权限。

```
drop view [if exists] view_name [, view_name] ...[restrict | cascade]
```

例如，删除 staff_list 视图：

```
mysql> drop view staff_list;
query ok, 0 rows affected (0.00 sec)
```

6. 在使用 show table status 命令的时候，不但可以显示表的信息，同时也可以显示视图的信息。所以，可以通过下面的命令显示视图的信息：（show table status 、显示视图信息）

```
show table status [from db_name] [like 'pattern']
```

下面演示的是查看 staff_list 视图信息的操作： （show table status like）

```
mysql> show table status like 'staff_list' \g
```

7. 如果需要查询某个视图的定义，可以使用 show create view 命令进行查看：

```
mysql> show create view staff_list \g
```

第 12 章 存储过程和函数

1. 存储过程和函数是事先经过编译并存储在数据库中的一段 sql 语句的集合。存储过程类似函数，和函数的区别在于函数必须有返回值，而存储过程可以没有，存储过程的参数可以使用 in、out、inout 类型（输入、输出、既可输入又可输出），而函数的参数只能是 in 类型的。（类似函数、语句集合、无返回值）

- in： 输入参数，表示该参数的值必须在调用存储过程时指定，在存储过程中修改该参数的值不能被返回，为默认值。也就是说，存储过程内部修改值后，并不影响外部该变量的值（in、输入、不能返回、并不影响）
- out： 输出参数，该值可在存储过程内部被改变，并可返回。也就是说，存储过程内部修改值后，会影响外部该变量的值（out、输出、能返回、影响外部）
- inout： 输入输出参数，调用时指定，并且可被改变和返回。（inout、输入输出、可改变与返回）

2. 创建、修改存储过程或者函数的示例：

```
mysql> delimiter //            （delimiter、修改结束符）
mysql> create procedure proc1(out s int)
    -> begin
    -> select count(*) into s from user;
    -> end
    -> //
mysql> delimiter ;
```

过程体的开始与结束使用 begin 与 end 进行标识。（过程体、begin 与 end）

3. mysql 的存储过程和函数中允许包含 ddl 语句，也允许在存储过程中执行提交（commit，即确认之前的修改）或者回滚（rollback，即放弃之前的修改），但是存储过程和函数中不允许执行 load data infile 语句。此外，存储过程和函数中可以调用其他的过程或者函数。下面创建了一个新的过程 film_in_stock：（不允许、load data infile 语句、create procedure ）

```
    mysql> delimiter $$                // 修改结束符
    mysql>
    mysql> create procedure film_in_stock(in p_film_id int, in p_store_id int,
out p_film_count int)
         -> reads sql data
         -> begin
         -> select inventory_id
         -> from inventory
         -> where film_id = p_film_id
         -> and store_id = p_store_id
         -> and inventory_in_stock(inventory_id);
         ->
         -> select found_rows() into p_film_count;
         -> end
         -> $$                                    // 结束符
query ok, 0 rows affected (0.00 sec)

    mysql>
    mysql> delimiter ;              // 还原结束符
```

“delimiter $$”命令将语句的结束符从“;”修改成“$$”，然后在存储过程的最后写上结束符。在存储过程或者函数创建完毕，通过“delimiter ;”命令再将结束符还原成“;”。（delimiter、修改结束符）　　（delimiter：读 dɪ'lɪmɪtə、修改结束符）

4. 和视图的创建语法稍有不同，存储过程和函数的 create 语法不支持使用 create or replace 对存储过程和函数进行修改，如果需要对已有的存储过程或者函数进行修改，需要执行 alter 语法。　　（存储、不支持 create or replace ）

5. 查看存储过程或者函数的状态（show {procedure | function} status）

```
show procedure status [like 'pattern']
```

或者：

```
show function status [like 'pattern']
```

下面演示的是查看过程 film_in_stock 的信息：

```
mysql> show procedure status like 'film_in_stock'\g
```

6. 查看存储过程或者函数的定义

```
show create procedure 数据库.存储过程名;
```

下面演示的是查看过程 film_in_stock 的定义：

```
mysql> show create procedure film_in_stock \g
```

7. 存储过程和函数中可以使用变量，而且在 mysql 5.1 版本中，变量是不区分大小写的。
8. 通过 declare 可以定义一个局部变量，该变量的作用范围只能在 begin…end 块中，可以用在嵌套的块中。变量的定义必须写在复合语句的开头，并且在任何其他语句的前面。可以一次声明多个相同类型的变量。如果需要，可以使用 default 赋默认值（declare、局部变量）

   定义一个变量的语法如下：

```
declare var_name[,...] type [default value]
```

例如，定义一个 date 类型的变量，名称是 last_month_start：

```
declare last_month_start date;
```

9. 变量可以直接赋值，或者通过查询赋值。直接赋值使用 set，可以赋常量或者赋表达式，具体语法如下：　　（set 赋值）

```
set 变量名 = 表达式值 [,variable_name = expression ...]
```

给刚才定义的变量 last_month_start 赋值，具体语法如下：

```
set last_month_start = date_sub(current_date(), interval 1 month);
```

也可以通过 select 查询将结果赋给变量，这要求查询返回的结果必须只有一行，具体语法如下：　　（select、只有一行）

```
select col_name[,...] into var_name[,...] table_expr
```

10. mysql 变量术语：

- 用户变量：以"@"开始，形式为"@变量名"，用户变量跟 mysql 客户端是绑定的，设置的变量，只对当前用户使用的客户端生效。　　（@用户变量）
- 全局变量：定义时，以如下两种形式出现，set global 变量名　或者　set @@global.变量名，对所有客户端生效。只有具有 super 权限才可以设置全局变量。（双@全局变量）
- 局部变量：作用范围在 begin 到 end 语句块之间。在该语句块里设置的变量，declare 语句专门用于定义局部变量。set 语句是设置不同类型的变量，包括会话变量和全局变量。

11. 条件的定义和处理可以用来定义在处理过程中遇到问题时相应的处理步骤。

- 条件的定义　　（declare…condition for…）

```
declare condition_name condition for condition_value

condition_value:
    sqlstate [value] sqlstate_value
    | mysql_error_code
```

这个定义是为mysql中的condition_value相应的错误定义一个别名，以供declare…handler for…使用。condition_value可以是一个mysql error code或一个sqlstate(由5个字符组成)。以下示例为1051声明一个别名并供declare…handler for…使用：　　（错误、别名）

```
declare no_such_table condition for 1051;
declare continue handler for no_such_table
  begin
    -- body of handler
  end;
```

- 条件的处理　　（declare…handler for…）

```
declare handler_type handler for condition_value[,...] sp_statement

handler_type:      // 处理类型
    continue       // 继续执行
    | exit             // 退出
    | undo

condition_value:
    sqlstate [value] sqlstate_value
    | condition_name
    | sqlwarning
    | not found
    | sqlexception
    | mysql_error_code
```

条件的处理用于处理在遇到某种问题时是继续执行是还是退出。例如下面的例子没有定义条件的处理，则遇到主键重复的错误时退出，语句‘set @x = 3;’无法执行。
（用于继续还是退出）

```
delimiter $$

create procedure actor_insert ()
       begin
       declare continue handler for sqlstate '23000' set @x2 = 1;
       set @x = 1;
       insert into actor(actor_id,first_name,last_name) values
(201,'test','201');
       set @x = 2;
       insert into actor(actor_id,first_name,last_name) values (1,'test','1');
       set @x = 3;
```

```
        end;
        $$

delimiter ;
```

使用条件处理的方法如下：

```
delimiter $$

create procedure actor_insert ()
        begin
        declare continue handler for sqlstate '23000' set @x2 = 1;
        set @x = 1;
        insert into actor(actor_id,first_name,last_name) values
(201,'test','201');
        set @x = 2;
        insert into actor(actor_id,first_name,last_name) values (1,'test','1');
        set @x = 3;
        end;
        $$

delimiter ;
```

12. 游标是用来接结果集的一个类型的变量。在存储过程和函数中可以使用游标对结果集进行循环的处理。游标的使用包括游标的声明、open、fetch 和 close，其语法分别如下。（游标、结果集变量、declare…cursor for…）

- 声明游标：（定义一个游标并直接赋值）

```
declare cursor_name cursor for select_statement
```

- open 游标：（open、打开结果集）

```
open cursor_name
```

- fetch 游标：（fetch...into...、把每条记录放入变量 var_name 中）

```
fetch cursor_name into var_name [, var_name] ...
```

- close 游标：

```
close cursor_name
```

游标实例：

```
        --在 windows 系统中写存储过程时，如果需要使用 declare 声明变量，需要添加这个关键字，否则会报错。
delimiter //
drop procedure if exists statisticstore;
create procedure statisticstore()
begin
    --创建接收游标数据的变量
    declare c int;
```

```
    declare n varchar(20);
    --创建总数变量
    declare total int default 0;
    --创建结束标志变量
    declare done int default false;
    --创建游标
    declare cur cursor for select name,count from store where name = 'iphone';
    --指定游标循环结束时的返回值
    declare continue handler for not found set done = true;
    --设置初始值
    set total = 0;
    --打开游标
    open cur;
    --开始循环游标里的数据
    read_loop:loop
    --根据游标当前指向的一条数据
    fetch cur into n,c;
    --判断游标的循环是否结束
    if done then
        leave read_loop;    --跳出游标循环
    end if;
    --获取一条数据时，将 count 值进行累加操作，这里可以做任意你想做的操作，
    set total = total + c;
    --结束游标循环
    end loop;
    --关闭游标
    close cur;

    --输出结果
    select total;
end;
--调用存储过程
call statisticstore();
```

13. 可以使用 if、case、loop、leave、iterate、repeat 及 while 语句进行流程的控制，下面将逐一进行说明。

(1) if 语句 (if…then…end if)

```
mysql > delimiter //
mysql > create procedure proc2(in parameter int)
    -> begin
    -> declare var int;
    -> set var=parameter+1;
    -> if var=0 then
    -> insert into t values(17);
    -> end if;
```

```
    -> if parameter=0 then
    -> update t set s1=s1+1;
    -> else
    -> update t set s1=s1+2;
    -> end if;
    -> end;
    -> //
mysql > delimiter ;
```

(2) case 语句 （case…when…then…end case）

```
mysql > delimiter //
mysql > create procedure proc3 (in parameter int)
    -> begin
    -> declare var int;
    -> set var=parameter+1;
    -> case var
    -> when 0 then
    -> insert into t values(17);
    -> when 1 then
    -> insert into t values(18);
    -> else
    -> insert into t values(19);
    -> end case;
    -> end;
    -> //
mysql > delimiter ;
```

(3) loop 实现简单的循环：（loop…end loop）

```
mysql > delimiter //
mysql > create procedure proc6 ()
    -> begin
    -> declare v int;
    -> set v=0;
    -> loop_lable:loop
    -> insert into t values(v);
    -> set v=v+1;
    -> if v >=5 then
    -> leave loop_lable;
    -> end if;
    -> end loop;
    -> end;
    -> //
mysql > delimiter ;
```

(4) leave 语句：用来从标注的流程构造中退出，通常和 begin ... end 或者循环

一起使用。　　（leave 退出）

(5)　　iterate 语句必须用在循环中，作用是跳过当前循环的剩下的语句，直接进入下一轮循环。　　（iterate 跳过、类似 continue）

```
mysql > delimiter //
mysql > create procedure proc10 ()
    -> begin
    -> declare v int;
    -> set v=0;
    -> loop_lable:loop
    -> if v=3 then
    -> set v=v+1;
    -> iterate loop_lable;
    -> end if;
    -> insert into t values(v);
    -> set v=v+1;
    -> if v>=5 then
    -> leave loop_lable;
    -> end if;
    -> end loop;
    -> end;
    -> //
mysql > delimiter
```

(6)　　repeat…end repeat 语句：有条件的循环控制语句，当满足条件的时候退出循环，它在执行操作后检查结果，而 while 则是执行前进行检查。具体语法如下：（repeat…end repeat 循环、操作后检查、类似 until）

```
mysql > delimiter //
mysql > create procedure proc5 ()
    -> begin
    -> declare v int;
    -> set v=0;
    -> repeat
    -> insert into t values(v);
    -> set v=v+1;
    -> until v>=5
    -> end repeat;
    -> end;
    -> //
mysql > delimiter ;
```

(7)　　while 语句实现的也是有条件的循环控制语句，即当满足条件时执行循环的内容，具体语法如下：　　（while…do…end while）

```
mysql > delimiter //
mysql > create procedure proc4()
```

```
    -> begin
    -> declare var int;
    -> set var=0;
    -> while var<6 do
    -> insert into t values(var);
    -> set var=var+1;
    -> end while;
    -> end;
    -> //
mysql > delimiter ;
```

while 循环和 repeat 循环的区别在于：while 是满足条件才执行循环，repeat 是满足条件退出循环；while 在首次循环执行之前就判断条件，所以循环最少执行 0 次，而 repeat 是在首次执行循环之后才判断条件，所以循环最少执行 1 次。（repeat、满足后退出、）

第 14 章 事务控制和锁定语句

1. 数据库系统引入事务的主要目的：事务会把数据库从一种一致状态转换成另外一种状态。在数据库提交工作时，可以确保其要么所有修改都已经保存了，要么所有修改都不保存。 （事务、要么保存、要么不保存）
2. lock tables 可以锁定用于当前线程的表。如果表被其他线程锁定，则当前线程会等待，直到可以获取所有锁定为止。unlock tables 可以释放当前线程获得的任何锁定。当前线程执行另一个 lock tables 时，或当与服务器的连接被关闭时，所有由当前线程锁定的表被隐含地解锁，具体语法如下： （lock、锁定、unlock、释放）

```
lock tables
    tbl_name [as alias] {read [local] | [low_priority] write}
    [, tbl_name [as alias] {read [local] | [low_priority] write}] ...
unlock tables
```

示例：

```
获得表 film_text 的 read 锁定
mysql> lock table film_text read;
query ok, 0 rows affected (0.00 sec)
```

3. 事物控制语句：

- begin 或 start transaction；显式地开启一个事务；（begin、start transaction、开始事务）
- commit；提交事务，并使已对数据库进行的所有修改称为永久性的；（commit、提交事务）
- rollback；有可以使用 rollback work，不过二者是等价的。回滚会结束用户的事务，并撤销正在进行的所有未提交的修改；（rollback、回滚）

- savepoint identifier；savepoint 允许在事务中创建一个保存点，一个事务中可以有多个 savepoint；（savepoint identifier、创建保存点）
- release savepoint identifier；删除一个事务的保存点，当没有指定的保存点时，执行该语句会抛出一个异常；（release savepoint identifier、 删除保存点）
- rollback to identifier；把事务回滚到标记点；（rollback to identifier、回滚到标记点）
- set transaction；用来设置事务的隔离级别。InnoDB 存储引擎提供事务的隔离级别有 read uncommitted、read committed、repeatable read 和 serializable。（set transaction、设置隔离级别）

4. 在同一个事务中，最好不使用不同存储引擎的表，否则 rollback 时需要对非事务类型的表进行特别的处理，因为 commit、rollback 只能对事务类型的表进行提交和回滚。 （同一个事务、不使用不同存储引擎）
5. 和 oracle 的事务管理相同，所有的 ddl 语句是不能回滚的，并且部分的 ddl 语句会造成隐式的提交。 （ddl 不能回滚）
6. 在事务中可以通过定义 savepoint，指定回滚事务的一个部分，但是不能指定提交事务的一个部分。对于复杂的应用，可以定义多个不同的 savepoint，满足不同的条件时，回滚不同的 savepoint。需要注意的是，如果定义了相同名字的 savepoint，则后面定义的 savepoint 会覆盖之前的定义。（savepoint 指定回滚）

```
mysql> savepoint test;
```

定义 savepoint，名称为 test。

```
mysql> rollback to savepoint test;
```

回滚到刚才定义的 savepoint （rollback 回滚）

```
mysql> commit;
```

用 commit 命令提交。 （commit 提交）

7. 启动一个事务命令：

```
mysql> start transaction;                    // 也可以使用 begin
```

8. 关闭自动提交：

```
set   autocommit =0;
```

开启自动提交：

```
set   autocommit =1;
```

个人总结之 grant

1. 使用 root 创建一个 mysql 数据库的用户时需要指定该数据库的访问权限，如当应用以 lan 用户登陆数据库时给予可读权限，以 wlan 用户登陆时给予可写权限。grant 命令就是对某个用户名赋予数据库赋予访问权限的。 （grant、权限）
2. mysql 赋予用户权限命令的简单格式可概括为：

```
grant 权限 on 数据库对象 to 用户@地址
```

例如：

```
grant select on *.* to dba@'localhost'；        #dba可以查询 mysql 中所有数据库中的表。
grant all    on *.* to dba@'localhost'；          #dba 可以管理 mysql 中的所有数据库
grant select, insert, update, delete on testdb.orders to dba@'localhost'；
#用户dba可以查询、
                                                                    插入、更新、删除数据库testdb.orders表
```

其中，dba是用户，localhost是地址，非本地时可以填写ip。

又如：

```
 grant all on wordpress.* to wordpress@'localhost' identified by '847339';
```

创建一个专用的wordpress数据库管理用户wordpress，密码为847339。格式为：

```
grant 权限 on 数据库对象 to 用户@‘地址’ identified by ‘密码’
```

3. mysql各种权限（共27个）：

(1) usage

连接（登陆）权限，建立一个用户，就会自动授予其usage权限（默认授予）。该权限只能用于数据库登陆，不能执行任何操作；且usage权限不能被回收，也即revoke用户并不能删除用户。

```
mysql> grant usage on *.* to ‘p1′ @’localhost’ identified by ‘123′;
```

(2) select：必须有select的权限，才可以使用select table。

(3) create：必须有create的权限，才可以使用create table。

(4) create routine：必须具有create routine的权限，才可以使用。

(5) create temporary tables(注意这里是tables，不是table，指创建临时表的权限)：必须有create temporary tables的权限，才可以使用create temporary tables。

(6) create view

(7) create user

(8) insert

(9) alter

(10) alter routine

(11) update

(12) delete

(13) drop

(14) show database

(15) show view

(16) index

(17) excute

(18) lock tables

(19) references：有了 references 权限，用户就可以将其它表的一个字段作为某一个表的外键约束。

(20) reload：必须拥有 reload 权限，才可以执行 flush [tables | logs | privileges]。

(21) replication client：拥有此权限可以查询 master server、slave server 状态。

(22) replication slave：拥有此权限可以查看从服务器，从主服务器读取二进制日志。

(23) shutdown

(24) grant option：拥有 grant option，就可以将自己拥有的权限授予其他用户（仅限于自己已经拥有的权限）。

(25) file：拥有 file 权限才可以执行 select ..into outfile 和 load data infile…操作，但是不要把 file, process, super 权限授予管理员以外的账号，这样存在严重的安全隐患。

(26) super：这个权限允许用户终止任何查询；修改全局变量的 set 语句。

(27) process：通过这个权限，用户可以执行 show processlist 和 kill 命令。默认情况下，每个用户都可以执行 show processlist 命令，但是只能查询本用户的进程。

第 15 章 SQL 中的安全问题

1. SQL 注入就是利用某些数据库的外部接口将用户数据插入到实际的数据库操作语言（SQL）当中，从而达到入侵数据库乃至操作系统的目的。它的产生主要是由于程序对用户输入的数据没有进行严格的过滤，导致非法数据库查询语句的执行。（SQL 注入、外部接口插入数据库操作语言中）

第 16 章 SQL Mode 及相关问题

1. 与其他数据库不同，MySQL 可以运行不同的 SQL Mode（SQL 模式）下。SQL Mode 定义了 MySQL 应支持的 SQL 语法、数据校验等，这样可以更容易地在不同的环境中使用 MySQL。（可以运行不同的 SQL Mode）
2. 查看默认 SQL Mode 的命令如下：

```
mysql> select @@sql_mode;
```

3. 常用的SQL Mode 值及其说明：

sql_mode 值	描述
ANSI	等同于 REAL_AS_FLOAT、PIPES_AS_CONCAT、ANSI_QUOTES、IGNORE_SPACE 和 ANSI 组合模式，这种模式使语法和行为更符合标准的SQL
STRICT_TRANS_TABLES	严格模式，不允许非法日期，也不允许超过字段长度的值插入字段中，对于插入不正确的值给出错误
TRADITIONAL	严格模式，对于插入不正确的值是给出错误而不是警告。可以应用在事务表和非事务表，用在事务表时，只要出现错误就会立即回滚

第17章 常用SQL 技巧和常见问题

1. regexp关键字用于匹配正则表达式：　　（regexp、匹配）

```
mysql> select name from person_tbl where name regexp '^st';
```

查找name字段中以'st'为开头的所有数据。

```
mysql> select name from person_tbl where name regexp 'ok$';
```

查找name字段中包含'mar'字符串的所有数据。

2. 默认情况下，表别名在UNIX 中对大小写敏感，但在Windows 或Mac OS X 中对大小写不敏感。（默认情况下、表别名、大小写敏感）

3. 定义主键的一种方法是创建它：

```
CREATE TABLE Vendors
(
    vend_id        CHAR(10)    NOT NULL PRIMARY KEY,
    vend_name      CHAR(50)    NOT NULL,
    vend_address   CHAR(50)    NULL,
    vend_city      CHAR(50)    NULL,
    vend_state     CHAR(5)     NULL,
    vend_zip       CHAR(10)    NULL
    vend_country   CHAR(50)    NULL
);
```

使用primary key 使之成为主键。（primary key 、主键）

第 18 章 SQL 优化

1. MySQL 客户端连接成功后，通过 show [session|global]status 命令可以查看服务器状态信息，show [session|global] status 可以根据需要加上参数“session”或者“global”来显示 session 级（当前连接）的统计结果和 global 级（自数据库上次启动至今）的统计结果。如果不写，默认使用参数是“session”。下面的命令显示了当前 session 中所有统计参数的值：（查看状态 show status like、session、global）

```
mysql> show status like 'Com_%';
+---------------------------+-------+
| Variable_name             | Value |
+---------------------------+-------+
| Com_admin_commands | 0         |
| Com_alter_db               | 0        |
| Com_alter_event            | 0       |
| Com_alter_table            | 0       |
| Com_analyze                | 0       |
| Com_backup_table         | 0        |
……
```

Com_xxx 表示每个 xxx 语句执行的次数，我们通常比较关心的是以下几个统计参数。（Com_xxx、语句执行次数）

- Com_select：执行 select 操作的次数，一次查询只累加 1。
- Com_insert：执行 INSERT 操作的次数，对于批量插入的 INSERT 操作，只累加一次。
- Com_update：执行 UPDATE 操作的次数。
- Com_delete：执行 DELETE 操作的次数

2. MySQL 的慢查询日志是 MySQL 提供的一种日志记录，它用来记录在 MySQL 中响应时间超过阀值的语句，具体指运行时间超过 long_query_time 值的 SQL，则会被记录到慢查询日志中。long_query_time 的默认值为 10，意思是运行 10S 以上的语句。默认情况下，Mysql 数据库并不启动慢查询日志，需要我们手动来设置这个参数，当然，如果不是调优需要的话，一般不建议启动该参数，因为开启慢查询日志会或多或少带来一定的性能影响。慢查询日志支持将日志记录写入文件，也支持将日志记录写入数据库表。（慢查询、超过的语句、long_query_time）

3. 以下几个参数便于用户了解数据库的基本情况。

- Connections：试图连接 MySQL 服务器的次数。（connections、连接次数）
- Uptime：服务器工作时间。（uptime、工作时间）
- Slow_queries：慢查询的次数。（slow_queries、慢查询次数）

4. 可以通过以下两种方式定位执行效率较低的 SQL 语句。（慢查询、show processlist 、查看当前的线程）

- 通过慢查询日志定位那些执行效率较低的 SQL 语句
- 慢查询日志在查询结束以后才纪录，所以在应用反映执行效率出现问题的时候查询慢查询日志并不能定位问题，可以使用 show processlist 命令查看当前 MySQL 在进行

的线程，包括线程的状态、是否锁表等，可以实时地查看 SQL 的执行情况，同时对一些锁表操作进行优化。

5. 通过以上步骤查询到效率低的 SQL 语句后，可以通过 explain 或者 desc 命令获取 mysql 如何执行 select 语句的信息，包括在 select 语句执行过程中表如何连接和连接的顺序。　　（explain 或者 desc、如何执行 select 的信息）

```
mysql> explain select sum(moneys) from sales a,company b where a.company_id =
b.id and a.year
= 2006\G;
*************************** 1. row ***************************
id: 1
select_type: SIMPLE
table: a
type: ALL
possible_keys: NULL
key: NULL
key_len: NULL
ref: NULL
rows: 1000                                                    #这里扫描行数
过多
Extra: Using where
*************************** 2. row ***************************
id: 1
select_type: SIMPLE
table: b
type: ref
possible_keys: ind_company_id
key: ind_company_id
key_len: 5
ref: sakila.a.company_id
rows: 1
Extra: Using where; Using index
2 rows in set (0.00 sec)
```

优化：a 表的 year 字段创建索引：

```
mysql> create index ind_sales2_year on sales2(year);
```

6. MySQL 中索引的存储类型目前只有两种（BTREE 和 HASH），具体和表的存储引擎相关：MyISAM 和 InnoDB 存储引擎都只支持 BTREE 索引；MEMORY/HEAP 存储引擎可以支持 HASH 和 BTREE 索引。下面是创建前缀索引的一个例子：（索引、BTREE 、HASH）

```
mysql> create index ind_company2_name on company2(name(4));
Query OK, 1000 rows affected (0.03 sec)
Records: 1000 Duplicates: 0 Warnings: 0
```

7. 如果索引正在工作，Handler_read_key 的值将很高，这个值代表了一个行被索引值读的次数，很低的值表明增加索引得到的性能改善不高，因为索引并不经常使用。Handler_read_rnd_next 的值高则意味着查询运行低效，并且应该建立索引补救。这个值的含义是在数据文件中读下一行的请求数。如果正进行大量的表扫描，Handler_read_rnd_next 的值较高，则通常说明表索引不正确或写入的查询没有利用索引，具体如下。 （索引、Handler_read_key 低、表示性能改善不高、没有利用）

```
mysql> show status like 'Handler_read%';
+----------------------------+-------+
| Variable_name              | Value |
+----------------------------+-------+
| Handler_read_first         | 0     |
| Handler_read_key           | 5     |
| Handler_read_next          | 0     |
| Handler_read_prev          | 0     |
| Handler_read_rnd           | 0     |
| Handler_read_rnd_next      | 2055  |
+----------------------------+-------+
6 rows in set (0.00 sec)
```

8. 分析表的语法如下：

```
analyze [local | no_write_to_binlog] table tbl_name [, tbl_name] ...
```

本语句用于分析和存储表的关键字分布，分析的结果将可以使得系统得到准确的统计信息，使得 SQL 能够生成正确的执行计划。下例中对表 sales 做了表分析：（分析表、关键字分布、analyze table）

```
mysql> analyze table sales;
+--------------+---------+-------------+-----------+
| Table        | Op      | Msg_type    | Msg_text  |
+--------------+---------+-------------+-----------+
| sakila.sales | analyze | status      | OK        |
+--------------+---------+-------------+-----------+
1 row in set (0.00 sec)
```

9. 检查表的语法如下：（检查表、是否有错误、check table）

```
check table tbl_name [, tbl_name] ... [option] ... option = {quick | fast | medium | extended | changed}
```

检查表的作用是检查一个或多个表是否有错误。CHECK TABLE 对 MyISAM 和 InnoDB 表有作用。对于 MyISAM 表，关键字统计数据被更新，例如：

```
mysql> check table sales;
```

10. 定期优化表。优化表的语法如下：（优化表、 optimize table）

```
optimize [local | no_write_to_binlog] table tbl_name [, tbl_name] ...
```

如果已经删除了表的一大部分，或者如果已经对含有可变长度行的表（含有 VARCHAR、BLOB 或 TEXT 列的表）进行了很多更改，则应使用 OPTIMIZE TABLE 命令来进行表优化。这个命令可以将表中的空间碎片进行合并，并且可以消除由于删除或者更新造成的空间浪费，但 OPTIMIZE TABLE 命令只对 MyISAM、BDB 和 InnoDB 表起作用。

```
mysql> optimize table sales;
```

11. load data infile 语句从一个文本文件中以很高的速度读入一个表中。（load data infile…into table…、高速读入另一个表）

```
mysql> load data infile "./data.txt" into table db2.my_table;
```

12. 当进行数据 insert 的时候，可以考虑采用以下几种优化方式。（插入优化）

- 如果同时从同一客户插入很多行，尽量使用多个值表的 insert 语句，这种方式将大大缩减客户端与数据库之间的连接、关闭等消耗，使得效率比分开执行的单个 insert 语句快(在一些情况中几倍)。下面是一次插入多值的一个例子：（使用多个值的 insert 语句）

```
insert into test values(1,2),(1,3),(1,4)…
```

- 如果从不同客户插入很多行，能通过使用 insert delayed 语句得到更高的速度。delayed 的含义存储在内存里面等待排队，当 mysql 有空余时，再插入。（insert delayed、等待排队、有空再插入）
- 将索引文件和数据文件分在不同的磁盘上存放（利用建表中的选项）；（索引、数据、不同磁盘）
- 当从一个文本文件装载一个表时，使用 load data infile。这通常比使用很多 insert 语句快 20 倍。

13. 默认情况下，mysql 对所有 group by col1,col2....的字段进行排序。这与在查询中指定 order by col1,col2...类似。因此，如果显式包括一个包含相同的列的 order by 子句，则对 mysql 的实际执行性能没有什么影响。如果查询包括 group by 但用户想要避免排序结果的消耗，则可以指定 order by null 禁止排序。
（order by null 避免消耗）

14. SQL 提示是优化数据库的一个重要手段，简单来说就是在 SQL 语句中加入一些人为的提示来达到优化操作的目的。

- 在查询语句中表名的后面，添加 use index 来提供希望 MySQL 去参考的索引列表，就可以让 MySQL 不再考虑其他可用的索引。（use index、参考的索引列表）

```
mysql> explain select * from sales2 use index (ind_sales2_id) where id = 3\G;
```

- 如果用户只是单纯地想让 MySQL 忽略一个或者多个索引，则可以使用 ignore index 作为提示。（ignore index、忽略索引列表）

```
mysql> explain select * from sales2 ignore index (ind_sales2_id) where id =
```

```
3\G;
```

第 19 章 优化数据库对象

1. 在 MySQL 中，可以使用函数 procedure analyse()对当前应用的表进行分析，该函数可以对数据表中列的数据类型提出优化建议，用户可以根据应用的实际情况酌情考虑是否实施优化。以下是函数 procedure analyse()的使用方法：（procedure analyse 提出优化建议）

```
select * from tbl_name procedure analyse();
select * from tbl_name procedure analyse(16,256);
```

输出的每一列信息都会对数据表中的列的数据类型提出优化建议。以上第二个语句告诉 procedure analyse()不要为那些包含的值多于 16 个或者 256 字节的 enum 类型提出建议。如果没有这样的限制，输出信息可能很长；enum 定义通常很难阅读。（多于 16 或 256 字节的 enum、不提建议）

2. 通过拆分提高表的访问效率（拆分、提高访问效率）
   - 第一种方法是垂直拆分，即把主码和一些列放到一个表，然后把主码和另外的列放到另一个表中。如果一个表中某些列常用，而另外一些列不常用，则可以采用垂直拆分，另外垂直拆分可以使得数据行变小，一个数据页就能存放更多的数据，在查询时就会减少 I/O 次数。（垂直拆分、拆分常用列）
   - 第二种方法是水平拆分，即根据一列或多列数据的值把数据行放到两个独立的表中，注意，并不区分是否常用。水平拆分通常在以下几种情况下使用：（水平拆分、拆分成独立的表、不区分常用）
     - 表很大，分割后可以降低在查询时需要读的数据和索引的页数，同时也降低了索引的层数，提高查询速度。（表很大）
     - 表中的数据本来就有独立性，例如，表中分别记录各个地区的数据或不同时期的数据，特别是有些数据常用，而另外一些数据不常用。（独立性）
     - 需要把数据存放到多个介质上。
3. 对于数据量较大的表，在其上进行统计查询通常会效率很低，并且还要考虑统计查询是否会对在线的应用产生负面影响。通常在这种情况下，使用中间表可以提高统计查询的效率。（数据量大的表、使用中间表提高效率）

第 20 章 锁问题

1. MySQL 的锁机制比较简单，其最显著的特点是不同的存储引擎支持不同的锁机制。比如，MyISAM 和 MEMORY 存储引擎采用的是表级锁（table-level locking）；BDB 存储引擎采用的是页面锁（page-level locking），但也支持表级锁；InnoDB 存储引擎既支持行级锁（row-level locking），也支持表级锁，但默认情况下是采用行级锁。MySQL 这 3 种锁的特性可大致归纳如下：
   - 表级锁：开销小，加锁快；不会出现死锁；锁定粒度大，发生锁冲突的概率最高,并

发度最低。 （表级锁、开销小、加锁快）

- 行级锁：开销大，加锁慢；会出现死锁；锁定粒度最小，发生锁冲突的概率最低,并发度也最高。 （行级表、开销大、加锁慢）
- 页面锁：开销和加锁时间界于表锁和行锁之间；会出现死锁；锁定粒度界于表锁和行锁之间，并发度一般。（处理两者之间）

表级锁更适合于以查询为主，只有少量按索引条件更新数据的应用，如 Web 应用；而行级锁则更适合于有大量按索引条件并发更新少量不同数据，同时又有并发查询的应用，如一些在线事务处理（OLTP）系统。（表级锁、适合少量索引的查询、行级锁、适合大量索引的查询）

2. 可以通过检查 table_locks_waited 和 table_locks_immediate 状态变量来分析系统上的表锁定争夺：（show status like 'table%'）

```
mysql> show status like 'table%';
+----------------------------+-------+
| Variable_name              | Value |
+----------------------------+-------+
| Table_locks_immediate | 2979 |
| Table_locks_waited       | 0       |
+----------------------------+-------+
2 rows in set (0.00 sec))
```

如果 Table_locks_waited 的值比较高，则说明存在着较严重的表级锁争用情况。（Table_locks_waited 高、争用严重）

3. MySQL 的表级锁有两种模式：表共享读锁（Table Read Lock）和表独占写锁（Table Write Lock）。下例获得表 film_text 的 WRITE 锁定：（表共享读锁、表独占写锁）

```
mysql> lock table film_text write;
```

释放锁：（ lock table...write...）

```
mysql> unlock tables;
```

4. MyISAM 在执行查询语句（SELECT）前，会自动给涉及的所有表加读锁，在执行更新操作（UPDATE、DELETE、INSERT 等）前，会自动给涉及的表加写锁，这个过程并不需要用户干预，因此，用户一般不需要直接用 LOCK TABLE 命令给 MyISAM 表显式加锁。（会自动加锁、不需要显式加锁）
5. 并发事务：多个事务操作同一段数据。 （并发事务、多个事务、同一数据）

第 21 章 优化 MySQL Server

1. MySQL 服务启动后，我们可以用 show variables 和 show status 命令查看 MySQL 的服务器静态参数值和动态运行状态信息。其中前者是在数据库启动后不会动态更改的值，比如缓冲区大小、字符集、数据文件名称等；后者是数据库运行期间的动态变化的信息，比如锁等待、当前连接数等。（show variables、静态参数值、不会更改、show status、运行状态、会更改）

2. 也可以在操作系统下直接查看数据库参数或数据库状态信息，具体命令如下：（查看参数和状态信息、mysqladmin -uroot variables）

```
[mysql@db3 bin]$ mysqladmin -uroot variables
```

3. MySQL 服务器的参数很多，如果需要了解某个参数的详细定义，可以使用以下命令：

```
[mysql@bj72 mysql]$ mysqld --verbose --help|more
```

查看 mysqld 命令的参数定义。（mysqld 参数定义、mysqld --verbose --help|more）

第 22 章 磁盘 I/O 问题

1. MySQL 的数据库名和表名是与文件系统的目录名和文件名对应的，默认情况下，创建的数据库和表都存放在参数 datadir 定义的目录下。这样如果不使用 RAID 或逻辑卷，所有的表都存放在一个磁盘设备上，无法发挥多磁盘并行读写的优势！在这种情况下，我们就可以利用操作系统的符号连接（Symbolic Links）将不同的数据库或表、索引指向不同的物理磁盘，从而达到分布磁盘 I/O 的目的：（符号连接、实现多磁盘并行读写）
2. atime 是 Linux/UNIX 系统下的一个文件属性，每当读取文件时，操作系统都会将读操作发生的时间回写到磁盘上。对于读写频繁的数据库文件来说，记录文件的访问时间一般没有任何用处，却会增加磁盘系统的负担，影响 I/O 的性能！因此，可以通过设置文件系统的 mount 属性，阻止操作系统写 atime 信息，以减轻磁盘 I/O 的负担。在 Linux 下的具体做法是修改文件系统配置文件/etc/fstab，指定 noatime 选项：（atime、记录发生时间、可以阻止记录 atime）

```
LABEL=/home /home ext3 noatime 1 2
```

然后重新 mount 文件系统：

```
#mount -oremount /home
```

完成上述操作，以后读/home 下文件就不会再写磁盘了。

第 25 章 MySQL 中的常用工具

1. mysql（客户端连接工具）语法如下：　（mysql、客户端连接工具）

```
mysql [OPTIONS] [database]
```

示例：

```
Mysql -u root password 847339
```

2. myisampack 是一个表压缩工具，可以使用很高的压缩率来对 MyISAM 存储引擎的表进行压缩，使得压缩后的表占用比压缩前小得多的磁盘空间。但是压缩后的表也将成为一个只读表，不能进行 DML 操作。此工具的用法如下：（myisampack、压缩 MyISAM 表）

```
shell> myisampack [options] filename
```

3. mysqladmin 是一个执行管理操作的客户端程序。可以用它来检查服务器的配置和当前的状态，创建并删除数据库等。它的功能和 mysql 客户端非常类似，主要区别在于它更侧重于一些管理方面的功能，比如关闭数据库。mysqladmin 的用法如下：（mysqladmin、管理操作）

```
shell> mysqladmin [options] command [command-options]
[command [command-options]] ...
```

使用方法和常用的选项和 mysql 非常类似，这里就不再赘述。这里将可以执行的命令行简单列举如下：

```
create databasename                    Create a new database
debug                                        Instruct server to write debug
information to log
drop databasename                      Delete a database and all its tables
extended-status                          Gives an extended status message
from the server
flush-hosts                                 Flush all cached hosts
flush-logs                                  Flush all logs
flush-status                               Clear status variables
flush-tables                               Flush all tables
flush-threads                             Flush the thread cache
flush-privileges                         Reload grant tables (same as reload)
kill id,id,...                                 Kill mysql threads
password new-password           Change old password to new-password, MySQL 4.1
hashing.
old-password new-password     Change old password to new-password in old
format.
ping                                          Check if mysqld is alive
processlist                                Show list of active threads in
server
reload                                      Reload grant tables
refresh                                     Flush all tables and close and
open logfiles
shutdown                                 Take server down
status                                       Gives a short status message from
the server
start-slave                                Start slave
stop-slave                                Stop slave
variables                                  Prints variables available
version                                    Get version info from server
```

这里简单举一个关闭数据库的例子：

```
[root@localhost test]# mysqladmin -uroot -p shutdown
```

```
Enter password:
```

4. Mysqlbinlog （mysqlbinlog、日志管理工具）

```
shell> mysqlbinlog [options] log-files1 log-files2...
```

option 有很多选项，常用的如下：

- -d, --database=name 指定数据库名称，只列出指定的数据库相关操作。
- -o, --offset=# 忽略掉日志中的前 n 行命令
- -r, --result-file=name 将输出的文本格式日志输出到指定文件
- -s, --short-form 显示简单格式，省略掉一些信息
- --set-charset=char-name：在输出为文本格式时，在文件第一行加上 set names char-name，这个选项在某些情况下装载数据时，非常有用。
- --start-datetime=name - stop-datetime=name：指定日期间隔内的所有日志
- --start-position=# --stop-position=#：指定位置间隔内的所有日志

5. mysqlcheck 客户端工具可以检查和修复 MyISAM 表，还可以优化和分析表。有 3 种方式可以来调用 mysqlcheck：（mysqlcheck、修复、优化、分析表）

```
shell> mysqlcheck[options] db_name [tables]
shell> mysqlcheck[options] ---database DB1 [DB2 DB3...]
shell> mysqlcheck[options] --all—database
```

option 中有以下常用选项：

- -c, --check 检查表
- -r, --repair 修复表
- -a, --analyze 分析表
- -o, --optimize 优化表

其中，默认选项是-c（检查表）。

```
[root@localhost mysql]# mysqlcheck -uroot -c test
```

6. mysqldump 客户端工具用来备份数据库或在不同数据库之间进行数据迁移。备份内容包含创建表或装载表的 SQL 语句。mysqldump 目前是 MySQL 中最常用的备份工具。有 3 种方式来调用 mysqldump：（mysqldump、备份数据库）

```
shell> mysqldump [options] db_name [tables] #备份单个数据库或者库中部分数据表
shell> mysqldump [options] ---database DB1 [DB2 DB3...] #备份指定的一个或者多个数据库
shell> mysqldump [options] --all—database #备份所有数据库
```

- 连接选项：

```
-u , --user=name 指定用户名
-p , --password[=name] 指定密码
```

```
-h , --host=name 指定服务器IP 或者域名
-P , --port=# 指定连接端口
```

这4 个选项经常一起配合使用，如果客户端位于服务器上，通常不需要指定host。如果不指定端口，默认连接到3306 端口，以下是一个远程客户端连接到服务器的例子：

- 输出内容选项

```
--add-drop-database 每个数据库创建语句前加上DROP DATABASE 语句
--add-drop-table 在每个表创建语句前加上DROP TABLE 语句
```

这两个选项可以在导入数据库的时候不用先手工删除旧的数据库，而是会自动删除，提高导入效率，但是导入前一定要做好备份并且确认旧数据库的确已经可以删除，否则误操作将会造成数据的损失。在默认情况下，这两个参数都自动加上。

- 输出格式选项：
  - --compact 选项使得输出结果简洁，不包括默认选项中的各种注释。
  - -c --complete-insert 选项使得输出文件中的insert 语句包括字段名称，默认是不包括字段名称的。
  - -T 选项将指定数据表中的数据备份为单纯的数据文本和建表SQL 两个文件。
- --default-character-set=name 选项可以设置导出的客户端字符集。

7. mysqlhotcopy 备份数据库或单个表的最快途径，其缺点是mysqlhotcopy 只用于备份MyISAM，而且它需要运行在Linux/UNIX 环境中。需要注意的是，mysqlhotcopy 是Perl 脚本，因此需要安装Perl 的MySQL 数据库接口包。（mysqlhotcopy、备份数据库、最快途径）
8. mysqlimport 是客户端数据导入工具，用来导入mysqldump 加-T 选项后导出的文本文件。mysqlimport 的基本用法如下：（mysqlimport、客户端数据导入工具）

```
shell> mysqlimport [options] db_name textfile1 [textfile2 ...]
```

9. mysqlshow 客户端对象查找工具，用来很快地查找存在哪些数据库、数据库中的表、表中的列或索引。和mysql 客户端工具很类似，不过有些特性是mysql 客户端工具所不具备的。mysqlshow 的使用方法如下：（mysqlshow、查找存在哪些数据库和表）

```
shell> mysqlshow[option] [db_name [tbl_name [col_name]]]
```

如果不加任何选项，默认情况下，会显示所有数据库。下例中显示了当前MySQL 中的所有数据库：

```
[zzx@localhost ~]$ mysqlshow -uroot
```

下面是mysqlshow 的一些常用选项。

- --count（显示数据库和表的统计信息）。如果不指定数据库，则显示每个数据库的名称、表数量、记录数量；如果指定数据库，则显示指定数据库的每个表名、字段数量，记录数量；如果指定具体数据库中的具体表，则显示表的字段信息。
- -k - keys（显示指定表中的所有索引）。此选项显示了两部分内容，一部分是指定表的表结构，另外一部分是指定表的当前索引信息。下例中显示了test 库中表emp 的表结构和当前索引信息：

```
[zzx@localhost mysql]$ mysqlshow -uroot test emp -k
```

- -i -status（显示表的一些状态信息）。

10. perror（错误代码查看工具）用法很简单，如下所示：（perror、错误代码查看工具）

```
perror [OPTIONS] [ERRORCODE [ERRORCODE...]]
```

在下面例子中，可以看一下错误号30 和60 分别是指什么错误：

```
[zzx@localhost mysql]$ perror 30 60
OS error code 30: Read-only file system
OS error code 60: Device not a stream
```

11. replace 是MySQL 自带的一个对文件中的字符串进行替换的工具，类似于Linux 下的sed，不过它的使用更加简单灵活。具体使用方法如下：（replace、字符串替换工具）

```
shell> replace from to [from to] ... -- file [file] ...
shell> replace from to [from to] ... < file
```

其中--表示字符串结束，文件的开始，可跟多个源文件，替换完毕后会覆盖原文件。<表示后面的文件作为输入，替换后的文本显示在标准输出上，不会覆盖原文件。示例将文件a 中的a1 和b1 分别替换为aa1 和bb1：

```
[zzx@localhost ~]$ replace a1 aa1 b1 bb1 -- a
```

第26章 MySQL 日志

1. 在MySQL 中，有4 种不同的日志，分别是错误日志、二进制日志（BINLOG 日志）、查询日志和慢查询日志，这些日志记录着数据库在不同方面的踪迹。（错误日志、二进制日志、查询日志、慢日志）
2. 错误日志是MySQL 中最重要的日志之一，它记录了当mysqld 启动和停止时，以及服务器在运行过程中发生任何严重错误时的相关信息。当数据库出现任何故障导致无法正常使用时，可以首先查看此日志。可以用--log-error[=file_name]选项来指定mysqld（MySQL 服务器）保存错误日志文件的位置。（错误日志、--log-error）
3. 二进制日志（BINLOG）记录了所有的DDL（数据定义语言）语句和DML（数据操纵语言）语句，但是不包括数据查询语句。语句以“事件”的形式保存，它描述了数据的更改过程。此日志对于灾难时的数据恢复起着极其重要的作用。当用--log-bin[=file_name]选项启动时，mysqld 将包含所有更新数据的SQL 命令写入日志文件。如果没有给出file_name 值，默认名为主机名后面跟“-bin”。如果给出了文件名，但没有包含路径，则文件默认被写入参数DATADIR（数据目录）指定的目录。（二进制日志、DDL 和DML、--log-bin）
4. 由于日志以二进制方式存储，不能直接读取，需要用mysqlbinlog 工具来查看，语

法如下：（mysqlbinlog 查看二进制日志）

```
shell> mysqlbinlog log-file;
```

5. system 命令可以在数据库中执行 shell 命令：（system、执行 shell 命令）

```
mysql> system pwd
```

6. 查询日志记录了客户端的所有语句，而二进制日志不包含只查询数据的语句。日志将写入参数 DATADIR（数据目录）指定的路径下，默认文件名是 host_name.log。（datadir 目录下、默认文件名 host_name.log）（查询日志、所有语句、host_name.log）
7. 慢查询日志记录了包含所有执行时间超过参数 long_query_time（单位：秒）所设置值的 SQL 语句的日志。获得表锁定的时间不算作执行时间。（--log-slow-queries、慢查询）
8. 慢查询日志的默认文件名是 host_name-slow.log。（默认文件名、host_name-slow.log）
9. 和错误日志、查询日志一样，慢查询日志记录的格式也是纯文本，可以被直接读取。（慢查询、可直接读取）

第 27 章 备份与恢复

1. 对于一个 DBA 来说，定制合理的备份策略无疑是很重要的，以下是我们在进行备份或恢复操作时需要考虑的一些因素。

- 确定要备份的表的存储引擎是事务型还是非事务性，两种不同的存储引擎备份方式在处理数据一致性方面是不太一样的。（确定是事务型的还是非事务型的）
- 确定使用全备份还是增量备份。全备份的优点是备份保持最新备份，恢复的时候可以花费更少的时间；缺点是如果数据量大，将会花费很多的时间，并对系统造成较长时间的压力。增量备份则恰恰相反，只需要备份每天的增量日志，备份时间少，对负载压力也小；缺点就是恢复的时候需要全备份加上次备份到故障前的所有日志，恢复时间会长些。（确定是增量备份还是全备份）
- 可以考虑采取复制的方法来做异地备份，但是记住，复制不能代替备份，它对数据库的误操作也无能为力。（可以考虑异地备份）
- 要定期做备份，备份的周期要充分考虑系统可以承受的恢复时间。备份要在系统负载较小的时候进行。（要定期备份）
- 确保 MySQL 打开 log-bin 选项，有了 BINLOG，MySQL 才可以在必要的时候做完整恢复，或基于时间点的恢复，或基于位置的恢复。
- 要经常做备份恢复测试，确保备份是有效的，并且是可以恢复的。

2. MySQL 中的逻辑备份是将数据库中的数据备份为一个文本文件，备份的文件可以被查看和编辑。在 MySQL 中，使用 mysqldump 工具来完成逻辑备份。有以下 3 种方法来

调用 mysqldump：

- 备份指定的数据库，或者此数据库中某些表。（备份指定的数据库、 mysqldump db_name）

```
shell> mysqldump [options] db_name [tables]
```

- 备份指定的一个或多个数据库。（备份一个或多个、 mysqldump ---database）

```
shell> mysqldump [options] ---database DB1 [DB2 DB3...]
```

- 备份所有数据库。（备份所有、mysqldump --all--database）

```
shell> mysqldump [options] --all--database
```

如果没有指定数据库中的任何表，默认导出所有数据库中所有表。

以下给出一些使用 mysqldump 工具进行备份的例子。

备份所有数据库：

```
[zzx@localhost ~]$ mysqldump -uroot -p --all-database > all.sql
Enter password:
```

备份数据库 test：

```
[zzx@localhost ~]$ mysqldump -uroot -p test > test.sql
Enter password:
```

备份数据库 test 下的表 emp：

```
[zzx@localhost ~]$ mysqldump -uroot -p test emp > emp.sql
Enter password:
```

备份数据库 test 下的表 emp 和 dept：

```
[zzx@localhost ~]$ mysqldump -uroot -p test emp dept > emp_dept.sql
Enter password:
```

3. mysqldump 的恢复也很简单，将备份作为输入执行即可，具体语法如下：（恢复数据、mysql -uroot -p dbname < bakfile）

```
mysql -uroot -p dbname < bakfile
```

注意，将备份恢复后数据并不完整，还需要将备份后执行的日志进行重做，语法如下：

```
mysqlbinlog binlog-file | mysql -u root -p***
```

以下是 mysqldump 备份与恢复的例子。

```
[root@localhost mysql]# mysqldump -uroot -p -l -F test >test.dmp
Enter password:
```

其中-l 参数表示给所有表加读锁，-F 表示生成一个新的日志文件。

恢复备份：

```
[root@localhost mysql]# mysql -uroot -p test < test.dmp
Enter password:
```

使用 mysqlbinlog 恢复自 mysqldump 备份以来的 BINLOG：

```
[root@localhost mysql]# mysqlbinlog localhost-bin.000015 | mysql -u root  -p
test
Enter password:
```

4. 由于误操作，比如误删除了一张表，这时使用完全恢复是没有用的，因为日志里面还存在误操作的语句，我们需要的是恢复到误操作之前的状态，然后跳过误操作语句，再恢复后面执行的语句，完成我们的恢复。这种恢复叫不完全恢复，在 MySQL 中，不完全恢复分为基于时间点的恢复和基于位置的恢复。（误删、完全恢复没用、语句中存在误删语句）
5. 物理备份又分为冷备份和热备份两种，和逻辑备份相比，它的最大优点是备份和恢复的速度更快，因为物理备份的原理都是基于文件的 cp。
6. 冷备份其实就是停掉数据库服务，cp 数据文件的方法。这种方法对 MyISAM 和 InnoDB 存储引擎都适合，但是一般很少使用，因为很多应用是不允许长时间停机的。（冷备份、停掉数据库服务）

进行备份的操作如下：停掉 MySQL 服务，在操作系统级别备份 MySQL 的数据文件和日志文件到备份目录。

进行恢复的操作如下：首先停掉 MySQL 服务，在操作系统级别恢复 MySQL 的数据文件；然后重启 MySQL 服务，使用 mysqlbinlog 工具恢复自备份以来的所有 BINLOG。

7. MySQL 中，对于不同的存储引擎热备份方法也有所不同。

第 28 章 MySQL 权限与安全

1. 在权限存取的两个过程中，系统会用到“mysql”数据库（安装 MySQL 时被创建，数据库名称叫“mysql”）中 user、host 和 db 这 3 个最重要的权限表。
2. 账号创建好后，可以通过如下命令进行查看权限：（show grants for user@host、查看权限）

```
show grants for user@host;
```

如下例所示：

```
mysql> show grants for z1@localhost;
```

host 可以不写，默认是“%”，如下所示：

```
mysql> show grants for z1;
```

3. 可以进行权限的的新增和回收。和账号创建一样，权限变更也有两种方法：使用 grant（新增）和 revoke（回收）语句，或者更改权限表。
4. revoke 语句可以回收已经赋予的权限，例如，收回 z2@localhost 上的 insert 和 select 权限：（revoke...on...from user@地址、回收权限）

```
mysql> revoke select,insert on *.* from z2@localhost;
```

usage 权限不能被回收，也就是说，revoke 用户并不能删除用户。

5. 修改账号密码：（修改密码、mysqladmin、set password）

- 方法 1：可以用 mysqladmin 命令在命令行指定密码。

```
shell> mysqladmin -u user_name -h host_name password "newpwd"
```

- 方法 2：执行 set password 语句。下例中将账号'jeffrey'@'%'的密码改为'biscuit'。

```
mysql> set password for 'jeffrey'@'%' = password('biscuit');
```

如果是更改自己的密码，可以省略 for 语句：

```
mysql> set password = password('biscuit');
```

- 方法 3：还可以在全局级别使用 grant usage 语句(在*.*)来指定某个账户的密码而不影响账户当前的权限。

```
mysql> grant usage on *.* to 'jeffrey'@'%' identified by 'biscuit';
```

- 方法 4：直接更改数据库的 user 表。

6. 要彻底删除账号，同样也有两种方法：drop user 命令和修改权限表。（删除账号、drop user、修改表权限）

- drop user 语法非常简单，具体如下：

```
drop user user [, user] ...
```

举一个简单的例子，将 z2@localhost 用户删除：

```
mysql> show grants for z2@localhost ;
```

- 修改权限表方法只要把 user 用户中的用户记录删除即可

7. 尽量避免以 root 权限运行 MySQL。（避免 root 权限登陆）

8. 在某些版本中，安装完毕 MySQL 后，会自动安装一个空账号，此账号具有对 test 数据库的全部权限，建议删除此空账号，或者对此账号加密码：（默认会有全权限的空账号、删除空账号）

```
mysql> drop user ''@'localhost';
```

9. MySQL 安装完毕后，root 默认口令为空，需要马上修改 root 口令：（需要马上改口令、set password=password('newpassword');）

```
[root@localhost zzx]# mysql -uroot
mysql> set password=password('newpassword');
```

10. process 权限能被用来执行“show processlist”命令，查看当前所有用户执

行的查询的明文文本，包括设定或改变密码的查询。在默认情况下，每个用户都可以执行“show processlist”命令，但是只能查询本用户的进程。因此，对 process 权限管理不当，有可能会使得普通用户能够看到管理员执行的命令。（process 权限、用来执行 show processlist）

```
mysql> grant process on *.* to 'z1'@'localhost';
mysql> show processlist;
```

11. super 权限能执行 kill 命令，终止其他用户进程。（super 权限能执行 kill 命令）

```
mysql> grant super on *.* to z1@localhost;
mysql> kill 51;
```

终止 id 为 51 的进程。

12. drop 表的时候，其他用户对此表的权限并没有被收回，这样导致重新创建同名的表时，以前其他用户对此表的权限会自动赋予，进而产生权限外流。因此，在删除表时，要同时取消其他用户在此表上的相应权限。（删除表时、需要取消用户在此表的权限）

第 29 章 MySQL 复制

1. 复制是指将主数据库的 DDL 和 DML 操作通过二进制日志传到复制服务器（也叫从服务器）上，然后在从服务器上对这些日志重新执行（也叫重做），从而使得从服务器和主服务器的数据保持同步。（复制、二进制文件传到从服务器、更新从服务器数据）

第 31 章 MySQL 常见问题和应用技巧

1. 忘记 MySQL 的 root 密码：（忘记 root 密码）

(1) 登录到数据库所在服务器，手工 kill 掉 MySQL 进程：

```
kill `cat /mysql-data-directory/hostname.pid`
```

其中，/mysql-data-directory/hostname.pid 指的是 MySQL 数据目录下的.pid 文件，它记录了 MySQL 服务的进程号。

(2) 使用--skip-grant-tables 选项重启 MySQL 服务：

```
[root@localhost mysql]# ./bin/mysqld_safe --skip-grant-tables --user=zzx &
[1] 20881
[root@localhost mysql]# Starting mysqld daemon with databases from
/home/zzx/mysql/data
```

(3) 用空密码的 root 用户连接到 MySQL，并且更改 root 口令：

```
[root@localhost mysql]# mysql -uroot
mysql> set password = password('123');
mysql> use mysql;
```

```
mysql> update user set password=password('123') where user='root' and
host='localhost';
```

此时，由于使用了--skip-grant-tables 选项启动，使用“set password”命令更改密码失败，直接更新user 表的password 字段后更改密码成功。

(4) 刷新权限表，使得权限认证重新生效：

```
mysql> flush privileges;
Query OK, 0 rows affected (0.00 sec)
```

(5) 重新用root 登录时，必须输入新口令。

□ □

1. □ □ A □ □ B □

(1) □ □ □ □ □ A □ □ □ □ □ □ □ □ □ B □ □ □ □ □ □ □ □ □ □ □ □ □ □

(2) □ □ □ □ □ A □ □ □ □ □ □ □ □ □ B □ □ □ □ □ □ □ □ □ □ □ □

(3) □ □ □ □ □ A □ □ □ □ □ □ □ □ □ B □ □ □ □ □ □ □ □ □ □ □ □

2.