

本节所讲内容：

- mysqlhotcopy
- mysqldump
- 修饰符
- 索引

mysqlhotcopy工具-----> 裸文件备份

原理：

如果备份时候不能停止mysql服务器，可以采用mysqlhotcopy工具。此工具比mysqldump命令快。

mysql工具是一个perl脚本，主要在linux下使用

mysqlhotcopy使用lock tables、flush tables和cp或scp来快速备份数据库.它是备份数据库或单个表最快的途径,完全属于物理备份,但只能用于备份MyISAM存储引擎和运行在数据库目录所在的机器上.

与mysqldump备份不同,mysqldump属于逻辑备份,备份时是执行的sql语句.使用

404

mysqlhotcopy命令前需要安装相应的软件依赖包.

Mysqlhotcopy 本质是使用锁表语句后再使用 cp 或 scp 拷贝数据库

```
[root@xuegod63 ~]# mysqladmin -u root password '123456'    #创建 root密码
```

1. 安装mysqlhotcopy所依赖的软件包(perl-DBD)

安装方法一：

```
[root@tong2 ~]# yum install perl-DBD* -y
```

安装方法二：

```
[root@xuegod63 ~]# ls /mnt/Packages/*DBD*
```

```
/mnt/Packages/perl-DBD-MySQL-4.013-3.el6.x86_64.rpm
```

```
/mnt/Packages/perl-DBD-Pg-2.15.1-3.el6.x86_64.rpm
```

```
/mnt/Packages/perl-DBD-SQLite-1.27-3.el6.x86_64.rpm
```

2.添加关于 mysqlhotcopy相关配置

```
[root@tong2 ~]# vim /etc/my.cnf    --在配置文件中添加如下参数
```

```
[mysqlhotcopy]
```

```
interactive-timeout
```

```
host=localhost
```

```
user=root
```

```
password=123456
```

```
port=3306
```

```
[root@tong2 ~]# /etc/init.d/mysqld restart    --重启服务
```

```
Shutting down MySQL.. SUCCESS!
```

```
Starting MySQL. SUCCESS!
```

2.查看mysqlhotcopy的帮助信息

```
[root@tong2 ~]# mysqlhotcopy --help
```

```
--allowold          don't abort if target dir already exists (rename it _old)  --
```

不覆盖以前备份的文件

`--addtodest` don't rename target dir if it exists, just add files to it

--属于增量备份

`--noindices` don't include full index files in copy --不备份索引

文件

`--debug` enable debug

--启用调试输出

`--regex=#` copy all databases with names matching regexp --使用正

规表达式

`--checkpoint=#` insert checkpoint entry into specified db.table --插入

检查点条目

`--flushlog` flush logs once all tables are locked --所

有表锁定后刷新日志

`--resetmaster` reset the binlog once all tables are locked --一旦锁表

重置binlog文件

`--resetslave` reset the master.info once all tables are locked --一旦锁表重

置master.info文件

3.备份一个数据库到一个目录中

```
[root@xuegod63 ~]# mysqlhotcopy -u root -p 123456 book /opt/ # -p和密
```

码之前要有空格

Locked 2 tables in 0 seconds.

Flushed tables (`book`.`books`, `book`.`category`) in 0 seconds.

Copying 7 files...

Copying indices for 0 files...

Unlocked tables.

mysqlhotcopy copied 2 tables (7 files) in 0 seconds (0 seconds overall).

```
[root@xuegod63 ~]# ls /opt/book/
```

```
books.frm  books.MYI      category.MYD  db.opt
```

```
books.MYD  category.frm  category.MYI
```

对比下大小

```
[root@xuegod63 ~]# du -h /opt/book/  /var/lib/mysql/book/
```

```
48K  /opt/book/
```

```
48K  /var/lib/mysql/book/
```

4.备份多个数据库:book和mysql库到一个目录中

```
[root@xuegod63 ~]# mkdir /opt/book-mysql
```

```
[root@xuegod63 ~]# mysqlhotcopy -u root -p 123456 book mysql
```

```
/opt/book-mysql/
```

```
Locked 23 tables in 0 seconds.  #锁住23个表
```

```
Flushed tables ( 刷新表 ) ('book`.`books`, `book`.`category`,
```

```
`mysql`.`columns_priv`, `mysql`.`db`, `mysql`.`event`, `mysql`.`func`,
```

```
`mysql`.`help_category`, `mysql`.`help_keyword`, `mysql`.`help_relation`,
```

```
`mysql`.`help_topic`, `mysql`.`host`, `mysql`.`ndb_binlog_index`, `mysql`.`plugin`,
```

```
`mysql`.`proc`, `mysql`.`procs_priv`, `mysql`.`servers`, `mysql`.`tables_priv`,
```

```
`mysql`.`time_zone`, `mysql`.`time_zone_leap_second`, `mysql`.`time_zone_name`,  
`mysql`.`time_zone_transition`, `mysql`.`time_zone_transition_type`, `mysql`.`user`) in  
0 seconds.
```

```
Copying 7 files... #复制7个文件
```

```
Copying indices for 0 files...
```

```
Copying 69 files...
```

```
Copying indices for 0 files...
```

```
Unlocked tables.
```

```
mysqlhotcopy copied 23 tables (76 files) in 1 second (1 seconds overall). #整体
```

花费1秒钟

5.备份数据库中某一个表

语法：mysqlhotcopy -u 用户 -p 密码 数据库名./要备份的表名/ 要备份的路径

```
[root@xuegod63 ~]# mkdir /opt/books
```

```
[root@xuegod63 ~]# mysqlhotcopy -u root -p 123456 book./books/
```

```
/opt/books/
```

```
Locked 1 tables in 0 seconds.
```

```
Flushed tables (`book`.`books`) in 0 seconds.
```

```
Copying 3 files...
```

Copying indices for 0 files...

Unlocked tables.

mysqlhotcopy copied 1 tables (3 files) in 0 seconds (0 seconds overall).

#实际上是把对应的表文件复制到/opt目录下

```
[root@tong2 ~]# ll /var/lib/mysql/mysql/user.*
```

```
-rw-r--r--. 1 mysql mysql 10684 Jan  4 16:49 /var/lib/mysql/mysql/user.frm
```

```
-rw-r--r--. 1 mysql mysql  784 Jan  4 16:49 /var/lib/mysql/mysql/user.MYD
```

```
-rw-r--r--. 1 mysql mysql  2048 Jan  4 16:49 /var/lib/mysql/mysql/user.MYI
```

```
[root@tong2 ~]#
```

总结：

mysqldump和mysqlhotcopy的比较：

1、mysqldump 是采用SQL级别的备份机制，它将数据表导成 SQL 脚本文件，数据库大时，占用系统资源较多，支持常用的MyISAM，innodb

2、mysqlhotcopy只是简单的缓存写入和文件复制的过程，占用资源和备份速度比mysqldump快很多很多。特别适合大的数据库，但需要注意的是：mysqlhotcopy只支持MyISAM 引擎

3、mysqlhotcopy只能运行在数据库目录所在的机器上，mysqldump可以用在远程客户端。

4、相同的地方都是在线执行LOCK TABLES 以及 UNLOCK TABLES

5、mysqlhotcopy恢复只需要COPY备份文件到源目录覆盖即可，mysqldump需要倒入SQL文件到原来库中。

拓展：

🔴 实战：写个自动备份 MySQL 数据库 shell 脚本

```
[root@xuegod63 ~]# cat mysql-autoback.sh
```

```
#!/bin/bash
```

```
export LANG=en_US.UTF-8
```

```
savedir=/database_back/
```

```
cd "$savedir"
```

```
time="$(date +%Y-%m-%d)"
```

```
mysqldump -u root -p123456 book > book-"$time".sql
```

然后添加一个计划任务就可以。

数据库迁移

数据库迁移是指将数据库从一个系统移动到另外一个系统上

数据库迁移场景：

1. 升级了计算机，部署开发的管理系统

2. 升级 mysql 数据库

3. 换用其他数据库

数据库分类：

1. 在相同版本的 mysql 数据库之间的迁移

2. 迁移到其他版本的 mysql 数据库当中

3. 迁移到其他类型的数据库中。

相同版本的 mysql 数据库之间的迁移

相同版本的数据库迁移的原因有很多，通常的原因是换了新的机器或者装了新的操作系统。还有一种常见的原因就是将开放的管理系统部署到工作的机器上，因为迁移后 mysql 数据库的主版本号相同，所以可以通过复制数据库目录来实现数据库迁移。但是只有数据库表都是 mysqlISAM 类型才能使用这种方式。

不同版本的数据库迁移通常是 mysql 升级的原因。例如原来的 4.0 版本要升级到 5.0 版本。这样就需要进行不同版本的数据库之间进行迁移。

高版本的 mysql 数据库通常都会兼容低版本，因此从低版本的 mysql 数据库迁移到高版本，对于 mysiam 类型的表可以直接复制，也可以使用 mysqlhotcopy 工具。

但是对于 innodb 类型的表不可以使用这两种方法。最常用的是使用 mysqldump 命令来备份，然后通过 mysql 命令将备份文件还原到目标的 mysql 当中去。迁移过程中一定要小心避免数据丢失。

学员拓展：

MySQL 管理之使用 XtraBackup 进行热备

<http://www.linuxidc.com/Linux/2014-04/99671.htm>

412

数据库选择备份数据库的方法？

根据数据库表的存储引擎的类型不同，备份表的方法也不一样，对于 myisam 类型表，可以直接复制 mysql 数据库文件或者使用 msyqlhotcopy 命令进行快速备份。复制 mysql 数据文件夹时将 mysql 服务停止。否则可能出现异常。而 mysqlhotcoyp 命令则不需要停止 mysql 服务器。mysqldump 命令是最安全的备份方法，它既适合 myisam 类型的表，也适合用于 innodb 类型的表。

面试必备考题：

如何升级 mysql 数据库：

1.先使用 `mysqldump` 命令备份 `mysql` 数据库中的数据。这样做的目的是为了避免误操作引起的 `mysql` 数据库中数据丢失。

2.停止 `mysql` 服务，可以直接终止 `mysql` 进程，但最好还是用安全的方法停止 `mysql` 服务，这样可以避免缓存中的数据丢失。

3.卸载旧版的 `mysql` 数据库，通常情况下，卸载 `mysql` 数据库软件时候，系统会继续保留 `mysql` 数据库中的数据文件

4.安装新版本的 `mysql` 数据库，并进行相应的配置

5.启动 `mysql` 服务，登陆 `mysql` 数据库查看数据是否完整，如果数据不完整使用之前的备份进行恢复。

拓展：看时间是否充足（若时间不够，学员可自由拓展）

实战，迁移数据

背景：公司业务数据 `book`,由于之前建表没注意字符集的问题，导致之前写入的数据出现乱码。现在要将之前的数据和现在数据的字符集一致，不出现乱码情况

将字符集为 `latin1` 已有记录的数据转成 `utf8`，并且已经存在的记录不乱码

步骤

1：建库及建表的语句导出，`sed` 批量修改为 `utf8`

2：导出之前所有的数据

3：修改 `mysql` 服务端和客户端编码为 `utf8`

4 : 删除原有的库表及数据

5 : 导入新的建库及建表语句

6 : 导入之前的数据

1): 导出表结构

```
mysqldump -uroot -p123456 --default-character-set=latin1 -d book >  
book.sql --default-character-set=utf8
```

414

2): 编辑 booktable.sql 将 latin1 修改成 utf8

```
vim booktable.sql
```

```
DEFAULT CHARSET=utf8
```

3): 确保数据库不再更新, 导出所有数据

```
mysqldump -usystem -p123456 --quick --no-create-info --extended-insert  
--default-character-set=latin1 book>bookdata.sql
```

参数说明 :

--quick: 用于转储大的表, 强制 mysqldump 从服务器一次一行的检索数据而不是检索所有行, 并输出当前 cache 到内存中

--no-create-info: 不要创建 create table 语句

--extended-insert: 使用包括几个 values 列表的多行 insert 语法, 这样文件更小, IO

也小，导入数据时会非常快

--default-character-set=latin1:按照原有字符集导出数据，这样导出的文件中，所有中文都是可见的，不会保存成乱码

4): 打开 bookdata.sql 将 set names latin1 修改成 set names utf8

```
vim bookdata.sql
```

```
/*!40101 SET NAMES utf8 */;
```

5): 重新建库

```
mysql> create database book2 default charset utf8;
```

6): 建立表，导入我们之前导出的表的数据库

```
mysql -usystem -p123456 book2 <booktable.sql
```

7): 导入数据

```
mysql -usystem -p123456 book2 <bookdata.sql
```

注意：选择目标字符集时，要注意最好大于等于原字符集（字库更大），否则可能会丢失不被支持的数据

一：字段修饰符

1、 null 和 not null 修饰符

我们通过例子来看一下：

```
mysql> create database bb;
```

```
mysql> use bb
```

```
mysql> create table kusers (id int not null,name varchar(8) not null,pass  
varchar(20) not null);
```

```
mysql> insert into kusers values(1,"mm",'123456');
```

Query OK, 1 row affected (0.00 sec)

```
mysql> insert into kusers values(1,"mm",NULL);
```

ERROR 1048 (23000): Column 'pass' cannot be null

416

此处插入 NULL 报错，意味着不能为 NULL

```
mysql> insert into kusers values(1,"mm","");
```

Query OK, 1 row affected (0.03 sec)

插入空值没有报错，可以为空。

```
mysql> select * from kusers;
```

```
+----+-----+-----+
```

```
| id | name | pass  |
```

```
+----+-----+-----+
```

```
| 1 | mm   | 123456 |
```

```
| 1 | mm   | NULL   |
```

```
| 1 | mm   |        |
```

```
+----+-----+-----+
```

```
3 rows in set (0.00 sec)
```

“空值” 和 “NULL” 有什么不一样？ null 和 not null 到底有什么不一样？

问题：

- 1、我字段类型是 not null，为什么我可以插入空值
- 2、为什么 not null 的效率比 null 高
- 3、判断字段不为空的时候，到底要 select * from table where column <> '' 还是要用 select * from table where column is not null 呢。

“空值” 和 “NULL” 有什么不一样？

- 1、空值是不占用空间的
- 2、mysql 中的 NULL 其实是占用空间的，下面是来自于 MYSQL 官方的解释

“NULL columns require additional space in the row to record whether their values are NULL. For MyISAM tables, each NULL column takes one bit extra, rounded up to the nearest byte.”

“空列需要行中的额外空间来记录其值是否为空。对于 MyISAM 表，每个 NULL 列需要一个额外的位，四舍五入到最接近的字节。”

比如：一个杯子，空值''代表杯子是真空的，NULL 代表杯子中装满了空气，虽然杯子看起来都是空的，但是里面是有空气的。

SELECT * FROM my_table WHERE phone = NULL; 表示有电话号码未知，但是有电话

SELECT * FROM my_table WHERE Phone='' 表示该人员没有电话，所以没有电话号码

举例：

```
CREATE TABLE `test` (  
  `col1` VARCHAR( 10 ) NOT NULL ,  
  `col2` VARCHAR( 10 ) NULL  
) ENGINE = MYISAM ;
```

插入数据：

```
INSERT INTO `test` VALUES (null,1);
```

mysql 发生错误：

```
#1048 - Column 'col1' cannot be null
```

再来一条

```
INSERT INTO `test` VALUES ('',1);
```

成功插入。

注：NOT NULL 的字段是不能插入 “NULL” 的，只能插入 “空值”。

对于问题 2，为什么 not null 的效率比 null 高？

NULL 其实并不是空值，而是要占用空间，所以 mysql 在进行比较的时候，NULL 会参与字段比较，所以对效率有一部分影响。

418 而且索引时不会存储 NULL 值的，所以如果索引的字段可以为 NULL，索引的效率会下降很多。

-Mysql 难以优化引用可空列查询，它会使索引、索引统计和值更加复杂。可空列需要更多的存储空间，还需要 mysql 内部进行特殊处理。可空列被索引后，每条记录都需要一个额外的字节，还能导致 MYisam

中固定大小的索引变成可变大小的索引-----这也是《高性能 mysql 第二版》介绍的解读：“可空列需要更多的存储空间”：需要一个额外字节作为判断是否为 NULL 的标志位“需要 mysql 内部进行特殊处理”

所有使用 not null 比 null 效率高

3、判断字段不为空的时候，到底要 `select * from table where column <> ''` 还是要用 `select * from table where column is not null` 呢。

```
INSERT INTO `test` VALUES ('', NULL);
```

```
INSERT INTO `test` VALUES ('1', '2');
```

现在表中数据：


```
mysql> select * from test ;
```

col1	col2
	1
	NULL
1	2

现在根据需求，我要统计 test 表中 col1 不为空的所有数据，我是该用 "<> ''" 还是 "IS NOT NULL" 呢，让我们来看一下结果的区别。

```
SELECT * FROM `test` WHERE col1 IS NOT NULL
```

```
mysql> select * from test where col1 is not null;
```

col1	col2
	1
	NULL
1	2

```
SELECT * FROM `test` WHERE col1 <> ''
```

```
mysql> select * from test where col1 <> '';
```

col1	col2
1	2

注：两次查询结果是不一样的。所以一定要根据业务需求，选择搜索条件。

为空表示不占空间，null 占用空间

2、default ： 设定字段的默认值

例：

```
mysql> create table kuser1 ( name varchar(8) not null ,dept varchar(25) default 'SNS');
Query OK, 0 rows affected (0.08 sec)
```

```
mysql> insert into kuser1 (name) values ('yiye');
Query OK, 1 row affected (0.00 sec)
```

```
mysql> select * from kuser1;
```

```
+-----+-----+
| name | dept |
+-----+-----+
| yiye | SNS  |
+-----+-----+
```

420 总结：如果字段没有设定 default ,mysql 依据这个字段是 null 还是 not null ,如果为可以为 null ,则为 null。如果不可以为 null ,报错。。

如果时间字段，默认为当前时间，插入 0 时，默认为当前时间。

如果是 enum 类型，默认为第一个元素。

3、auto_increment 字段约束:

increment[英][ˈɪŋkrəmənt] 增量

只能修饰 int 字段。表明 mysql 应该自动为该字段生成一个唯一没有用过的数（每次在最大 ID 值的基础上加 1。特例：如果目前最大 ID 是 34，然后删除 34，新添加的会是 35.）。对于主键，这是非常 有用的。可以为每条记录创建一个惟一的标识符。

```
mysql> create table items ( id int not null auto_increment primary key , label varchar(20) not null);
```

```
mysql> insert into items (label) values ('aaba');
```

```
mysql> insert into items values (9,'aaba');
```

互动：插入下面这条语句，id 将为多少？

```
mysql> insert into items (label) values ('aaccccaaba');
```

```
mysql> select * from items;
```

```

+----+-----+
| id | label      |
+----+-----+
|  1 | aa         |
|  9 | aaba       |
| 10 | aacccaaba  |
+----+-----+
3 rows in set (0.00 sec)

```

id 将为 10.

```
mysql> insert into items values (9,'ccc');    #再插入相同的记录，会报错
ERROR 1062 (23000): Duplicate entry '9' for key 'PRIMARY'
Duplicate    ['dupli.keit'] 重复的

```

清空表中所有记录：

方法一：delete 不加 where 条件，清空所有表记。但是 delete 不会清零 auto_increment 值

```
mysql> delete from items;
Query OK, 4 rows affected (0.00 sec)

```

```
mysql> select * from items;
Empty set (0.00 sec)

```

```
mysql> insert into items (label) values ("aaaa");
Query OK, 1 row affected (0.00 sec)

```

```
mysql> select * from items;
+----+-----+
| id | label |
+----+-----+
| 11 | aaa a |
+----+-----+
1 row in set (0.00 sec)

```

方法二：删除表中所有记录，清 auto_increment 值。

truncate

作用：删除表的所有记录，并清零 auto_increment 值。新插入的记录从 1 开始。

语法：truncate table name;

```
mysql> truncate table items;

```

Query OK, 0 rows affected (0.01 sec)

```
mysql> select * from items;
```

Empty set (0.03 sec)

```
mysql> insert into items values (null,'accaa');    #插入 null 时，会自动填充值
```

Query OK, 1 row affected (0.00 sec)

```
mysql> select * from items;
```

```
+----+-----+
```

```
| id | label |
```

```
+----+-----+
```

```
|  1 | accaa |
```

```
+----+-----+
```

1 row in set (0.00 sec)

```
mysql> insert into items (label) values ("aaaa");
```

Query OK, 1 row affected (0.00 sec)

```
mysql> select * from items;
```

```
+----+-----+
```

```
| id | label |
```

```
+----+-----+
```

```
|  1 | accaa |
```

```
|  2 | aaaa  |
```

```
+----+-----+
```

2 rows in set (0.01 sec)

二 索引

索引由数据库表中一列或者多列组合而成，其作用是提高对表中数据的查询速度。

快速查找记录出现的。

mysql 中的索引是以文件的形式存放的。如果对表进行增删改操作时，索引文件将会同步更新，使索引文件和表保持一致。

优点：可以提高检索数据的速度，这是创建索引的最主要的原因；对于有依赖关系的子表和父表之间的联合查询时，可以提高查询速度，使用分组和排序子句进行数据查询时，同样可以显著节省查询中

的分组和排序时间。

缺点：

1 索引是以文件存储的。如果索引过多，占磁盘空间较大。而且他影响：insert,update,delete 执行时间。

2：索引中数据必须与数据表数据同步：如果索引过多，当表中数据更新的时候后，索引也要同步更新，这就降低了效率。

3.每一个索引都要站一定的物理空间，增加删除和修改数据的时候，要动态的维护索引，造成数据的维护速度降低。

总结：凡是常用在 where 的字段一般加索引。

索引类型：

- 1、普通索引
- 2、唯一性索引
- 3、主键索引（主索引）
- 4、复合索引

索引的创建

1.创建索引是指在某个表的一列或者多列上建立一个索引，以便提高对表的访问速度。创建索引有三种方式

2.创建表的时候创建索引

3.在已经存在的表上创建索引

4.使用 ALTER TABLE 语句来创建索引。

普通索引：最基本的索引，不具备唯一性。

在创建普通索引时候，不附加任何限制条件，这类索引可以创建在任何数据类型上，其值是否唯一和非空由字段本身的完整性约束条件决定，建立索引后，查询时候可以通过索引进行查询。

索引的设计原则：

为了使索引的使用效率更高，在创建索引的时候必须考虑在哪些字段上创建索引和创建什么类型的索引。

选择唯一性索引

为经常需要排序，分组和联合操作的字段建立索引

为常作为查询条件的字段建立索引

限制索引的数目

尽量使用数据量少的索引

尽量使用前缀来索引

删除不再使用或者很少使用的索引

创建普通索引：

方法一：创建表时添加索引

```
create table 表名 (
    列定义
    index 索引名称 ( 字段 )
    index 索引名称 ( 字段 )
)
```

注：可以使用 key，也可以使用 index。 index 索引名称 (字段)，索引名称，可以加也可以不加，不加使用字段名作为索引名。。

424

例：

#使用密码字段创建一个索引

```
mysql> create table demo( id int(4),uName varchar(20),uPwd varchar(20),index (uPwd));
Query OK, 0 rows affected (0.28 sec)
```

或者这样写

```
create table demo(
    id int(4) ,
    uName varchar(20),
    uPwd varchar(20),
    index (uPwd)
);
```

注意：index 和 key 是相同的

```
mysql> create table demo1( id int(4) auto_increment primary key, uName varchar(20),
uPwd varchar(20), key (uPwd) );
```

```
mysql> create table demo2( id int(4), name varchar(20), pwd varchar(20), key(pwd) );
```

```
mysql> create table demo3( id int(4), name varchar(20), pwd varchar(20), key
index_pwd(pwd) ); #加上名称
```

方法二：当表创建完成后，使用 alter 为表添加索引：

alter [!o:ltə(r)] 改变

alter table 表名 add index 索引名称 (字段 1, 字段 2.....);

show create table 表名;查看建表信息的。

查看索引：

mysql> desc demo;

Field	Type	Null	Key	Default	Extra
id	int(11)	YES		NULL	
uName	varchar(20)	YES		NULL	
uPwd	varchar(20)	YES	MUL	NULL	

注：如果 Key 是 MUL，那么该列的值可以重复，该列是一个非唯一索引的前导列(第一列)或者是一个唯一性索引的组成部分但是可以含有空值 NULL。就是表示是一个普通索引。

mysql> show create table demo;

Table	Create
demo	CREATE TABLE `demo` (`id` int(11) DEFAULT NULL, `uName` varchar(20) DEFAULT NULL, `uPwd` varchar(20) DEFAULT NULL, KEY `uPwd` (`uPwd`)) ENGINE=MyISAM DEFAULT CHARSET=latin1

我们先删除索引

mysql> alter table demo drop key uPwd; 注意此处的 pwd 指的是索引的名称,而不是表中 pwd 的那个字段

再用 alter 添加

mysql> alter table demo add key(uPwd);

2 唯一性索引

与普通索引基本相同，但有一个区别：索引列的所有值都只能出现一次，即必须唯一，用来约束内容，字段值只能出现一次。应该加唯一索引。唯一性允许有 NULL 值<允许为空>。

使用 UNIQUE 参数可以设置索引为唯一性索引，在创建唯一性索引时，限制该索引的值必须是唯一的。通过唯一性索引，可以更快速的确定某条记录。主键就是一种特殊唯一性索引。

unique 美 [ju'nik]

426

方法一：创建表时加唯一索引

```
create table 表名(  
    列定义：  
    unique key 索引名 ( 字段 );  
)
```

注意：常用在值不能重复的字段上，比如说用户名，电话号码，身份证号。

例：

```
create table demo3(  
    id int(4) auto_increment primary key,  
    uName varchar(20),  
    uPwd varchar(20),  
    unique index (uName)  
);
```

查看索引：

```
mysql> desc demo3; //
```

Field	Type	Null	Key	Default	Extra
id	int(4)	NO	PRI	NULL	auto_increment
uName	varchar(20)	YES	UNI	NULL	
uPwd	varchar(20)	YES		NULL	

+-----+-----+-----+-----+-----+-----+

查看创建表的语句：

```
mysql> show create table demo3;
```

-----+

```
| demo3 | CREATE TABLE `demo3` (  
    `id` int(4) NOT NULL AUTO_INCREMENT,  
    `uName` varchar(20) DEFAULT NULL,  
    `uPwd` varchar(20) DEFAULT NULL,  
    PRIMARY KEY (`id`),  
    UNIQUE KEY `uName` (`uName`)  
  ) ENGINE=MyISAM DEFAULT CHARSET=latin1 |
```

+-----+-----+-----+-----+-----+-----+
-----+

修改表时加唯一索引

方法二：alter table 表名 add unique 索引名 (字段);

```
mysql> alter table demo3 drop key uName;
```

```
mysql> alter table demo3 add unique(uName);
```

6. 空间索引

使用 SPATIAL 参数可以设置索引为空间索引。空间索引只能建立在空间数据类型上，这样可以提高系统获取空间数据的效率。空间数据类型包括 GEOMETRY、POINT、LINESTRING、POLYGON 等。目前只有 MyISAM 存储引擎支持空间检索。这类索引很少会用到。

3 主索引（主键）

primary key 约束。

查询数据库，按主键查询是最快的，每个表只能有一个主键列，可以有多个普通索引列。主键列要求列的所有内容必须唯一，而索引列不要求内容必须唯一，不允许为空

主键特点：此字段的记录是唯一，主键字段很少被修改。一般主键都约束为 :auto_increment 或 not null 和 unique。 不能为空，不能重复。 主键可以用于一个字段，也可以多字段。

创建主键索引

方法一：创建表创建主键索引

方法一：语法：

```
create table 表名 (  
    列定义 primary key,  
    ...  
)
```

```
mysql> create table demo4 (id int(4) not null auto_increment primary key,name  
varchar(4) not null);
```


```
mysql> create table demo5( id int(4) not null auto_increment, name varchar(20)  
default null,primary key(id));
```

428

```
+-----+-----+-----+-----+-----+-----+  
| Field | Type          | Null | Key | Default | Extra          |  
+-----+-----+-----+-----+-----+-----+  
| id    | int(4)        | NO   | PRI | NULL    | auto_increment |  
| name  | varchar(20)   | YES  |     | NULL    |                |  
+-----+-----+-----+-----+-----+-----+  
2 rows in set (0.00 sec)
```

```
mysql> show create table demo5;
```

```
show index from demo5 \G
```

```
mysql> show index from demo5\G  
***** 1. row *****  
      Table: demo5  
      Non_unique: 0  
      Key_name: PRIMARY   
Seq_in_index: 1  
Column_name: id  
Collation: A  
Cardinality: 0  
Sub_part: NULL  
Packed: NULL  
Null:  
Index_type: BTREE  
Comment:
```

方法二：创建表后添加 <不推荐>

先删除测试

```
mysql> alter table demo5 drop primary key;
ERROR 1075 (42000): Incorrect table definition; there can be only one auto
column and it must be defined as a key
```

删除遇到这种情况是 auto_increment 的原因

```
mysql> alter table demo5 change id id int(4) not null;
```

```
mysql> desc demo5;
```

```
+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| id    | int(4)        | NO   | PRI | NULL    |       |
| name  | varchar(20)   | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+
```

2 rows in set (0.00 sec)

```
mysql> alter table demo5 drop primary key;
```

再添加

```
mysql> alter table demo5 change id id int(4) not null primary key auto_increment;
```

总结：主键索引，唯一性索引区别：主键索引不能有 NULL，唯一性索引可以有空值。

4、复合索引

索引可以包含一个、两个或更多个列。两个或更多个列上的索引被称作复合索引

例：创建一个表存放，服务器允许或拒绝的 IP 和 port。要记录中 IP 和 port 要唯一。

```
mysql> create table firewall ( host varchar(15) not null ,port smallint(4) not
null ,access enum('deny','allow') not null, primary key (host,port));
```

Query OK, 0 rows affected (0.07 sec)

```
mysql> desc firewall;
+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| host  | varchar(15)   | NO   | PRI | NULL    |       |
| port  | smallint(4)   | NO   | PRI | NULL    |       |
| access | enum('deny','allow') | NO   |     | NULL    |       |
+-----+-----+-----+-----+-----+
```

varchar(11) : 192.168.168.198 12+3 =15 #应该定义长度为 15

mysql> insert into firewall values ('111.1.11.1',21,'deny');

Query OK, 1 row affected (0.00 sec)

mysql> insert into firewall values ('111.1.11.1',21,'allow'); #插入的 IP 和 prot 一样时 ,
报错。

ERROR 1062 (23000): Duplicate entry '111.1.11.1-21' for key 'PRIMARY'

mysql> insert into firewall values ('111.1.11.1',80,'allow');

Query OK, 1 row affected (0.00 sec)

mysql> insert into firewall values ('111.1.11.2',80,'allow');

Query OK, 1 row affected (0.01 sec)

430

插入一样就报错，唯一