MySQL

基本命令

```
net start mysql
mysql -u root -p #如果我们要登录本机的MySQL数据库，只需要该命令即可
mysql -h 主机名 -u 用户名 -p
# -h: 指定客户端所要登录的MySQL主机名，登录本机（localhost或127.0.0.1）该参数可以忽略
# -u: 登录的用户名
# -p: 告诉服务器将使用一个密码来登录，如果所要登录的用户名密码为空，可以忽略此选项

use 数据库名; # 选择要操作的mysql数据库，使用该命令后所有的mysql命令都针对该数据库
show databases; # 列出mysql数据管理系统的数据库列表
show tables; # 显示指定数据库的所有表，使用该命令前需要使用use命令来选择要操作的数据库
show columns from 数据表; # 显示数据表的属性，属性类型，主键信息，是否为NULL，默认值等其他信息
show index from 数据表; # 显示数据表的详细索引信息，包括PRIMARY KEY(主键)
show table status like [from db_name] [like 'pattern'] \g; # 输出mysql数据库管理系统的性能及统计信息
# show table status from db;
# show table status from db like 'runoob%'; # 表名以runoob开头的表的信息
# show table status from db like 'runoob%'\g; # 加上\g，查询结果按列打印
```

```
create database 数据库名; # create 创建数据库
mysqladmin -u root -p create db # 命令终端执行
drop database 数据库名; # 删除数据库
mysqladmin -u root -p drop db # 命令终端执行
```

数值类型

- 数值、日期/时间和字符串（字符）类型
- 数值类型：INTEGER, SMALLINT, DECIMAL, NUMERIC, 以及近似值数据类型(FLOAT, REAL, DOUBLE PRECISION)

日期和时间类型

- 表示时间值的日期和时间类型为DATETIME, DATE, TIMESTAMP, TIME, YEAR
- 每个时间类型有一个有效值范围和一个“零”值，当指定不合法的MySQL不能表示的值时使用“零”值
- TIMESTAMP类型有专有的自动更新特性

字符串类型

- 字符串类型指CHAR, VARCHAR, BINARY, BLOB, TEXT, ENUM和SET
- char(n), varchar(n)中括号中n代表字符的个数，并不代表字节个数

MySQL创建数据表

创建MySQL数据表需要以下信息：

- 表名

- 表字段名
- 定义每个表字段

```
create table table_name (column_name column_type); # 创建MySQL数据表的通用语法

create table if not exists runoob_tb2(
    runoob_id INT NOT NULL AUTO_INCREMENT,
    runoob_title VARCHAR(100) NOT NULL,
    runoob_author VARCHAR(40) NOT NULL,
    submission_date DATE,
    PRIMARY KEY (runoob_id)
)ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

- 如果不想字段为null可以设置字段的属性为not null， 在操作数据库时如果输入该字段的数据为null，就会报错。
- auto_increment定义列为自增的属性，一般用于主键，值的数目会自动加1
- primary key关键字用于定义列为主键，可以使用多列来定义主键，列间以逗号分隔
- engine设置存储引擎，charset设置编码

```
mysql> CREATE TABLE runoob_tbl(
   -> runoob_id INT NOT NULL AUTO_INCREMENT,
   -> runoob_title VARCHAR(100) NOT NULL,
   -> runoob_author VARCHAR(40) NOT NULL,
   -> submission_date DATE,
   -> PRIMARY KEY ( runoob_id )
   -> )ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE if not exists runoob_tbl(
runoob_id INT NOT NULL AUTO_INCREMENT,
runoob_title VARCHAR(100) NOT NULL,
runoob_author VARCHAR(40) NOT NULL,
submission_date DATE,
PRIMARY KEY ( runoob_id )
)ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

MySQL删除数据表

```
drop table table_name;
```

MySQL插入数据

```
insert into table_name (field1, field2, ...fieldN)
                       values
                       (value1, value2, ...valueN);
```

如果数据是字符型，必须使用单引号或者双引号，如：“value"。

```
mysql>insert into db
    ->(runoob_title, runoob_author, submission_date)
    ->values
    ->("学习PHP", "菜鸟教程", now());


insert into runoob_tbl
(runoob_title, runoob_author, submission_date)
values
("学习PHP", "菜鸟教程", now());
```

上例，我们没有提供 'runoob_id' 的数据，因为该字段我们在创建表的时候已经设置它为 auto_increment(自动增加)属性，所以，该字段会自动递增而不需要我们去设置，实例中now()是一个 MySQL函数，该函数返回日期和时间。

```
select * from runoob_tbl; # 读取数据表
```

MySQL查询数据

语法

```
select column_name, column_name
FROM table_name
[WHERE Clause]
[LIMIT N][OFFSET M]
```

- 查询语句中可以使用一个或多个表，表之间使用逗号(,)分割, 并使用WHERE语句来设定查询条件
- SELECT命令可以读取一条或多条记录
- 可以使用*来代替其他字段， SELECT语句会返回表的所有字段数据
- 可以用WHERE语句来包含任何条件
- 可以使用LIMT属性来设定返回的记录数
- 可以通过OFFSET指定SELECT语句开始查询的数据偏移量，默认情况下偏移量为0

```
select * from runoob_tbl;
```

MySQL WHERE 子句

```
SELECT field1, field2, ...fieldN FROM table_name1, table_name2...
[WHERE condition1 [AND[OR]] condition2...
```

- 可以在WHERE子句中指定任何条件
- 可以使用AND或OR指定一个或多个条件
- WHERE子句也可以运用于SQL的DELETE或者UPDATE命令
- WHERE子句类似于程序语言中的if条件，根据MySQL表中的字段值来读取指定的数据。

MySQL的WHERE子句的字符串比较是不区分大小写的。可以使用BINARY关键字来设定WHERE子句的字符串比较是区分大小写的。

```
SELECT * from runoob_tbl WHERE runoob_author="菜鸟教程";

SELECT * from runoob_tbl WHERE BINARY runoob_author='runoob.com';

SELECT * from runoob_tbl WHERE BINARY runoob_author='RUNOOB.COM';
```

MySQL UPDATE 更新

```
UPDATE table_name SET field1=new-value1, field2=new-value2
[WHERE Clause]
```

- 可以同时更新一个或多个字段
- 可以在WHERE子句中指定任何条件
- 可以在一个单独表中同时更新数据

```
UPDATE runoob_tbl SET runoob_title='学习JAVA' WHERE runoob_id=3;
# 更新表中runoob_id为3的runoob_title字段值
```

MySQL DELETE 语句

可以使用SQL的DELETE FROM命令来删除MySQL数据表中的记录

```
DELETE FROM table_name [WHERE Clause]
```

- 如果没有指定WHERE字句，MySQL表中所有记录将被删除
- 可以在WHERE子句中指定任何条件
- 可以在单个表中一次性删除记录

```
DELETE FROM runoob_tbl WHERE runoob_id=3;
```

MySQL LIKE语句

获取含有某个字段的记录，LIKE语句

SQL LIKE子句中使用百分号**%**字符表示任意字符，类似于UNIX或正则表达式中的星号*

如果没有使用百分号**%**，LIKE子句与等号**=**效果是一致的

```
SELECT field1, field2, ...fieldN
FROM table_name
WHERE field1 LIKE condition1 [AND[OR]] field2 = 'somevalue'
```

- 可以在WHRER子句中指定任何条件
- 可以在WHERE子句中使用LIKE子句
- 可以使用LIKE子句代替等号**=**
- LIKE通常与**%**一同使用，类似于一个原字符的搜索
- 可以使用**AND**或者**OR**指定一个或多个条件
- 可以在DELETE或UPDATE命令中使用WHERE..LIKE子句来指定条件

```
SELECT * from runoob_tbl WHERE runoob_author LIKE '%COM';
```

MySQL UNION 操作符

用于连接两个以上的SELECT语句的结果组合到一个结果集合中，多个SELECT语句会删除重复的数据

```
SELECT expression1, expression2, ...expression_n
FROM tables
[WHERE conditions]
UNION [ALL | DISTINCT]
SELECT expression1, expression2, ...expression_n
FROM tables
[WHERE conditions];
```

- expression1, ... expression_n: 要检索的列
- tables：要检索的数据表
- WHERE conditions: 可选，检索条件
- DISTINCT: 可选，删除结果集中重复的数据。默认情况下UNION操作符已经删除了重复数据，所以DISTINCT修饰符对结果没啥影响
- ALL：可选，返回所有结果集，包含重复数据

```
SELECT * FROM Websites;

SELECT * FROM apps;

SELECT country FROM Websites
UNION
SELECT country FROM apps
ORDER BY country;
```

UNION 不能用于列出两个表中所有的country，如果一些网站和APP来自同一个国家，每个国家只会列出一次。UNION只会选取不同的值。使用UNION ALL 来选取重复的值

下面的SQL语句使用UNION ALL从websites和apps表中选取所有的country（也有重复值）

```
SELECT country FROM Websites
UNION ALL
SELECT country FROM apps
ORDER BY country;
```

下面的SQL语句使用UNION ALL从websites和apps表中选取所有的中国的数据（也有重复值）

```
SELECT country, name FROM Websites
WHERE country='CN'
UNION ALL
SELECT country, app_name FROM apps
WHERE country='CN'
ORDER BY country;
```

MySQL 排序

使用SQL SELECT 语句来读取数据， 使用MySQL的ORDER BY子句来设定你想按哪个字段那种方式来进行排序，再返回搜索结果。

```
SELECT field1, field2, ...fieldN FROM table_name1, table_name2...
ORDER BY field1 [ASC [DESC][默认 ASC]], [field2...] [ASC [DESC][默认 ASC]]
```

- 可以使用任何字段来作为排序的条件，从而返回排序后的查询结果
- 可以设定多个字段来排序
- 可以使用ASC或DESC关键字来设置查询结果是按升序或降序排列。默认升序排列
- 可以添加WHERE...LIKE子句来设置条件

```
SELECT * from runoob_tbl ORDER BY submission_data ASC;

SELECT * from runoob_tbl ORDER BY submission_data DESC;
```

MySQL GROUP BY 语句

GROUP BY语句根据一个或多个列结果进行分组

在分组的列上我们可以使用COUNT, SUM, AVG等函数

```
SELECT column_name, function(column_name)
FROM table_name
WHERE column_name operator value
GROUP BY column_name;
```

```
set names utf8;
SELECT * FROM employee_tbl;
SELECT name, COUNT(*) FROM employee_tbl GROUP BY name;
# 利用GROUP BY语句将数据表按名字进行分组，并统计每个人有多少条记录：


# WITH ROLLUP 可以实现在分组统计数据基础上再进行相同的统计（SUM, AVG, COUNT...）
SELECT name, SUM(singin) as singin_count FROM employee_tbl GROUP BY name WITH
ROLLUP;


# coalesce语法
select coalesce(a,b,c);
# 如果a==null,则选择b；如果b==null，则选择c；如果a!=null,则选择a;如果a，b,c都为null，则返
回null
SELECT coalesce(name, '总数'), SUM(singin) as singin_count FROM employee_tbl
GROUP BY name WITH ROLLUP;
```

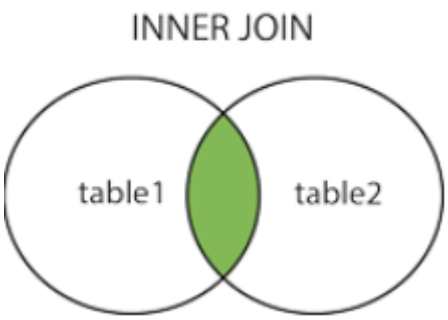
MySQL 连接的使用

JOIN在两个或多个表中查询数据

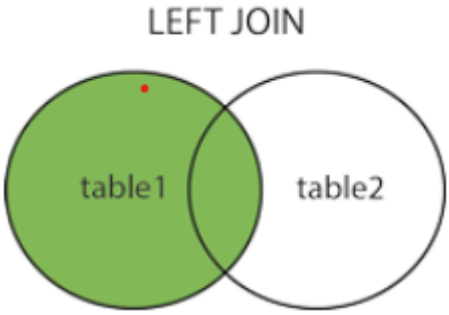
JOIN按照功能大致分为3类

- INNER JOIN(内连接，或等值连接)：获取两个表中字段匹配关系的记录。
- 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
- RIGHT JOIN(右连接)： 与LEFT JOIN相反，用于获取右表所有记录，即使左表没有对应匹配的记录

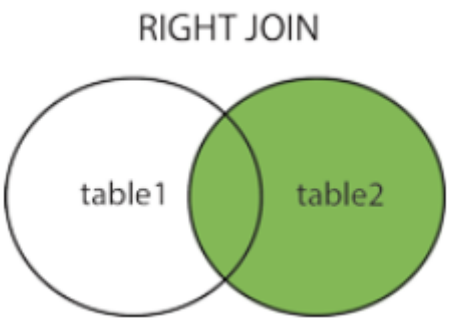
```
SELECT a.runoob_id, a.runoob_author, b.runoob_count FROM runoob_tbl a INNER JOIN
tcount_tbl b ON a.runoob_author = b.runoob_author;
```




```
SELECT a.runoob_id, a.runoob_author, b.runoob_count FROM runoob_tbl a LEFT JOIN
tcount_tbl b ON a.runoob_author = b.runoob_author;
```




```
SELECT a.runoob_id, a.runoob_author, b.runoob_count FROM runoob_tbl a RIGHT JOIN
tcount_tbl b ON a.runoob_author = b.runoob_author;
```



MySQL NULL值处理

MySQL使用SQL SELECT 命令及WHERE子句来读取表中的数据，但是当提供的查询条件字段为NULL时，该命令可能就无法正常工作

为了处理这种情况，MySQL提供了三大运算符：

- IS NULL：当前值是NULL, 此运算符返回true
- IS NOT NULL：当前值不为NULL, 运算符返回true
- <=>：比较操作符（不同于 = 运算符），当比较的两个值相等或者都为NULL时返回true

```
SELECT * , columnName1+ifnull(colunmName2,0) from tableName;
```

columnName1, columnName2为int型，当columnName2中有值为null时，columnName1+columnName2=null, **ifnull**(columnName2, 0)把columnName2中null值转为0

```
INSERT INTO runoob_test_tbl (runoob_author, runoob_count) values ('菜鸟教程',
NULL);
```

= 和 != 运算符在这里不起作用

MySQL 正则表达式

MySQL可以通过LIKE ...%来进行模糊匹配

MySQL使用REGEXP操作符来进行正则表达式匹配

- ^：匹配输入字符串的开始位置，如果设置了RegExp对象的Multiline属性，^也匹配'\n'或'\r'之后的位置
- $：匹配输入字符串的结束位置，如果设置了RegExp对象的Multiline属性，$也匹配'\n', '\r'之前的位置

  匹配除'\n'之外的任何单个字符，要匹配包括'\n'在内的任何字符，请使用像'.\n'的模式
- [...]：字符集合，匹配所包含的任意一个字符，例如，['abc']可以匹配"plain"中的'a'
- [^...]：负值字符集合。匹配未包含的任意字符，

  ```
  [^abc] 可以匹配"plain"中的'p'
  ```
- p1|p2|p3：匹配p1或p2或p3。例如：'z|food'能匹配'z'或'food'。'(z|f)food'则匹配"zood"或"food"。
- *：匹配前面的子表达式零次或多次。例如

```
zo*能匹配"z"以及"zoo"， 等价于{0,}
```

- +：匹配前面的子表达式一次或多次，例如，"zo+"能匹配"zo"以及"zoo", 但不能匹配"z",+等价于{1,}
- {n}：n是一个非负整数，匹配确定的n次，例如，o{2}不能匹配"Bob"中的"o",但是能匹配"food"中的两个o
- {n,m}：m和n均为非负整数，其中n <= m, 最少匹配n次且最多匹配m次

```
# 查找name字段中以'st'为开头的所有数据
select name from person_tbl where name regexp '^st';
# 查找name字段中以'ok'为结尾的所有数据
select name from person_tbl where name regexp 'ok$';
# 查找name字段中包含'mar'字符串的所有数据
select name from person_tbl where name regexp 'mar';
# 查找name字段中以元音字符开头或以'ok'字符串结尾的所有数据
select name from person_tbl where name regexp '^[aeiou]|ok$';
```

MySQL 事务

MySQL事务主要用于处理操作量大，复杂度高的数据，比如说，在人员管理系统中，你删除一个人员，既需要删除人员的基本资料，也要删除和该人员相关的信息，如信箱，文章等等，这样，这些数据库操作系统语句就构成了一个事务

- 在MySQL中只有使用了Innodb数据库引擎的数据库或表才支持事务
- 事务处理可以用来维护数据库的完整性，保证成批的SQL语句要么全部执行，要么全部不执行
- 事务用来管理insert，update， delete语句

一般来说，事务必须满足4个条件（ACID）：原子性（Atomicity，或称不可分割性）、一致性（Consistency）、隔离性（Lsolation, 又称独立性）、持久性（Durability）

- **原子性**：一个事务（transaction）中的所有操作，要么全部完成，要么全部不完成，不会结束在中间某个环节。事务在执行过程中发生错误，会被回滚（Rollback）到事务开始前的状态，就像这个事务从来没有执行过一样
- **一致性**：在事务开始和之前和事务结束以后，数据库的完整性没有被破坏。这表示写入的资料必须完全符合所有的预设规则，这包含资料的精确度、串联性以及后续数据库可以自发性地完成预定的工作。
- **隔离性**：数据库允许多个并发事务同时对其数据进行读写和修改的能力，隔离性可以防止多个事务并发执行时由于交叉执行而导致数据的不一致。事务隔离分为不同级别，包括读未提交（Read uncommitted）、读提交（read committed）、可重复读（repeatable read）和串行化（Seriazlizable）。
- **持久性**：事务处理结束后，对数据的修改就是永久的，即便系统故障也不会丢失

在MySQL命令行的默认设置下，事务都是自动提交的，即执行SQL语句后就会马上执行COMMIT操作。因此要显式地开启一个事务务须使用命令BEGIN或START TRANSACTION，或者执行命令SET AUTOCOMMIT=0，用来禁止使用当前会话的自动提交。

事务控制语句

- BEGIN或START TRANSACTION显式地开启一个事务；
- COMMIT也可以使用COMMIT WORK, 不过二者是等价的。COMMIT会提交事务，并使已对数据库进行的所有修改成为永久性的；
- ROLLNACL也可以使用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；**
- 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;
- RELESE 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；
- ROLLBACK TO identifier把事务回滚到标记点；
- 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

MySQL 事务处理主要有两种方法：

1、用BEGIN, ROLLBACK, COMMIT来实现

- **BEGIN** 开始一个事务
- **ROLLBACK** 事务回滚
- **COMMIT** 事务确认

2、直接使用 SET 来改变MySQL的自动提交模式

- SET AUTOCOMMIT=0 禁止自动提交
- SET AUTOCOMMIT=1 开启自动提交

```
use runoob;
create table runoob_transaction_test(id int(5)) engine=innodb;# 创建数据表
select * from runoob_transaction_test;
begin;# 开始事务
insert into runoob_transaction_test value(5);
insert into runoob_transaction_test value(6);
commit;# 提交事务
select * from runoob_transaction_test;
begin;# 开始事务
insert into runoob_transaction_test values(7);
rollback;# 回滚
select * from runoob_transaction_test;# 因为回滚所以数据没有插入
```

MySQL ALTER命令

用于修改数据表名或者修改数据表字段。

```
mysql -u root -p password;
use runoob;
create table testaler_tbl
(
i int,
c char(1),
);
show columns from testalter_tbl;

alter table testalter_tbl drop i;# 使用alter命令及drop子句来删除以上创建的i字段。如果表中只剩余一个字段则无法使用drop来删除字段
alter table testalter_tbl add i int;# 使用add子句来向数据表中添加列，在表中添加i字段，并定义数据类型：
alter table testalter_tbl drop i;
alter table testalter_tb1 add i int first;# 使用关键字first（设定位第一列）
alter table testalter_tbl drop i;
alter table testalter_tbl add i int after c;# 使用关键字after（设定位于某个字段之后）
# first和after关键字可用于add与modify子句，所以如果你想重置数据表字段的位置就需要先使用drop删除字段然后使用add来添加字段并设置位置
```

修改字段类型及名称

需要修改字段类型及名称，可以在alter命令中使用modify或change子句

```
alter table testalter_tbl modify c char(10);# 把字段c的类型从char(1)改为char(10);
alter table testalter_tbl change i j bigint;# 在change关键字之后，紧跟着要修改的字段名，然后指定新字段名及类型
alter table testalter_tbl change j j int;
```

ALTER TABLE 对Null值和默认值的影响

当修改字段时，可以指定是否包含值或者是设置默认值。

```
alter table testalter_tbl
modify j bigint not null default 100;# 指定字段j为not null且默认值为100；如果不设置默认值，MySQL会自动设置该字段默认为null
```

修改字段默认值

```
alter table testalter_tbl alter i set default 1000;# 使用alter字段来修改字段的默认值
show columns from testalter_tbl;
alter table testalter_tbl alter i drop default;# 使用alter命令及drop子句来删除字段的默认值
show columns from testalter_tbl;

# 修改数据表类型，可以使用alter命令及type子句来完成。以下我们将表testalter_tbl的类型修改为MYISAM;查看数据表类型可以使用show table status语句
alter table testalter_tbl engine = myisam;
show table status like 'testalter_tbl'\G;
```

修改表名

如果需要修改数据表的名称，可以在alter table 语句中使用rename子句来实现。

```
alter table testalter_tbl tename to alter_tbl;
```

MySQL 索引

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，即一个索引包含多个列。

创建索引时，需要确保该索引是应用在SQL查询语句的条件（一般作为WHERE子句的条件）

实际上，索引也是一张表，该表保存了主键与索引字段，并指向实体表的记录。

但是过多的索引将会造成滥用，因此索引也会有它的缺点：虽然索引大大提高了查询速度，同时却会降低更新表的速度，如对表insert， undate和delete。因为更新表时，MySQL不仅要保存数据，还要保存以下索引文件。

建立索引会占用磁盘空间的索引文件。

普通索引

```
CREATE INDEX indexName on mytable(username(length));
# 如果是CHAR， VARCHAR类型，length可以小于字段实际长度；如果是blob和text类型，必须指定length。
```

修改表结构（添加索引）

```
alter table tableName add index indexName(columnName);

# 创建表的时候直接指定
create table mytable(
id int not null,
username varchar(16) not null,
index [indexName] (username(length))
);
# 删除索引的语法
drop index [indexName] on mytable;
```

唯一索引

它与前面的普通索引类似，不同的是：索引列的值必须唯一，但允许有空值。如果是组合索引，则列值的组合必须唯一。

```
# 创建索引
create unique index indexName on mytable(username(length))
# 修改表结构
alter table mytable add unique [indexName] (username(length))
# 创建表的时候直接指定
create table mytable(
id int not null,
username varchar(16) not null,
unique [indexName] (userName(length))
);
```

使用alter命令添加和删除索引

- alter table tbl_name add primary key (column_list)：该语句添加一个主键，这意味着索引值必须是唯一的，且不能为NULL
- ALTER TABLE tbl_name add unique index_name(column_list): 该语句创建索引的值必须是唯一的（除了null外，null可能会出现多次）
- alter table tbl_name add index index_name(column_list): 添加普通索引，索引值可出现多次
- alter table tbl_name add fulltext index_name(column_list): 该语句指定了索引为fulltext, 用于全文索引

```
alter table testalter_tbl add index(c);# 在表中添加索引

alter table testalter_tbl drop index c;# drop子句来删除索引
```

使用alter命令添加和删除主键

```
alter table testalter_tb1 modify i int not null;# 主键只能作用与一个列上，添加主键索引时，需要确保该主键默认不为空（not null）
alter table testalter_tbl add primary key(i);

alter table testalter_tbl drop primary key;# alter命令删除主键
# 删除主键只需要指定primary key，但在删除索引时，必须知道索引名
```

显示索引名

```
show index from table_name;\G # 可以使用show index命令来列出表中相关的索引信息，可以通过添加\G来格式化输出信息
```

MySQL 临时表

MySQL临时表在我们需要保存一些临时数据时是非常有用的。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。

```
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 all default 0
);
```

```
insert into SalesSummary
(product_name, total_sales, avg_unit_price, total_units_sold)
values
('cucumber', 100.25, 90, 2);


select * from SalesSummary;

# 当你使用show tables命令显示数据列表时，将无法看到SalesSummary表
# 如果退出当前mysql会话，再使用select命令来读取原先创建的临时表数据，发现数据库中没有该表，因为退出时该表已经被销毁了



# 删除MySQL临时表
# 默认请况下，断开与数据库的连接后，临时表就会自动销毁。当然可以使用drop table命令来删除临时表
drop table SalesSummary;
# 检查
select * from salessummary;
```

MySQL 复制表

如何完整复制MySQL的数据表：

- 使用show create table 命令获取创建数据表（create table）语句，该语句包含了原数据表的结构，索引等。
- 复制以下命令显示的sql语句，修改数据表名，并执行sql语句，通过以上命令将完全的复制数据表结构。
- 如果你想复制表的内容，就可以使用insert into ... select语句来实现

```
show create table runoob_tbl \G;
# 创建新的克隆表
create table 'clone_tbl'(
'runoob_id' int(11) not null auto_increment,
'runoob_title' varchar(100) not null default '',
'runoob_author' varchar(40) not null default '',
'submission_date' date default null,
primary key ('runoob_id'),
unique key 'author_index' ('runoob_author')
)engine=innodb;
# 拷贝数据表的数据
insert into clone_tbl(runoob_id,
                      runoob_title,
                      runoob_author,
                      submission_date)
select runoob_id, runoob_title, runoob_author, submission_date)
from runoob_tbl;
# 在数据库中创建新的克隆表clone_tbl，然后使用insert into ... select来拷贝数据表

# 执行完上述步骤后，将完整的复制表，包括表数据及表结构
```

MySQL 元数据

MySQL 三种信息：

- **查询结果信息**： select，update或delete语句影响的记录数
- **数据库和数据表的信息**：包含里数据库及数据表的结构信息
- **MySQL服务器信息**：包含了数据库服务器的当前状态，版本号等

在MySQL的命令提示符中，我们可以很容易的获取以上服务器信息。

命令	描述
select version()	服务器版本信息
select database()	当前数据库名（或者返回空）
select user()	当前用户名
show status	服务器状态
show variables	服务器配置变量

MySQL 序列使用

MySQL 序列是一组整数：1，2，3，... 由于一张数据表只能有一个字段自增主键，如果像实现其他字段也实现自动增加，就可以使用MySQL序列来实现。

使用AUTO_INCREMENT

MySQL 中最简单使用序列的方法就是使用MySQL AUTO_INCREMENT来定义列。

```
create table insect(
id int unsigned not null auto_increment,# 重点
primary key(id),
name varchar(30) not null, # type of insect
date DATE not null# where collected
);

insert into insect (id, name, date, origin)
values
(null, 'housefly', '2001-09-10', 'kitchen'),
(null, 'millipede', '2001-09-10', 'driveway'),
(null, 'grasshopper', '2001-09-10', 'front yard');

select * from insect order by id;

# 重置序列
# 如果删除了数据表中的多条记录，并希望对剩下数据的auto_increment列进行重新排列，那么可以通过删除自增的列，然后重新添加来实现，不过操作也要非常小心，如果在删除的同时又有新记录添加，有可能会出现数据混乱
alter table insect drop id;
alter table insect
add id int unsigned not null auto_increment first,
add primary key(id);

# 设置序列的开始值
# 一般情况下序列的开始值未1，但如果需要指定一个开始值100，那可以通过以下语句来实现：
create table insect(
id int unsigned not null auto_increment,
primary key(id),
name varchar(30) not null,
```

```
date DATE not null,
origin varchar(30) not null
)engine=innodb auto_increment=100 charset=utf-8;
# 或者可以在表创建成功后，通过以下语句来实现
alter table t auto_increment = 100;
```

MySQL 处理重复数据

有些M有SQL数据表中可能存在重复的记录，有些情况我们允许数据的存在，但有时候我们也需要删除这些重复的数据。

防止表中出现重复的数据

可以在MySQL数据表中设置指定的字段未PRIMAYR KEY（主键）或者UNIQUE（唯一）索引来保证数据的唯一性，下表无索引及主键，所以该表允许出现多条重复记录。

```
create table person_tbl(
first_name char(20),
last_name char(20),
sex char(10)
);

# 如果像设置表中字段first_name, last_name数据不能重复，可以设置双主键模式来设置数据的唯一性，如果设置了双主键，那么那个键的默认值不能未为null，可以设置为not null
create table person_tbl(
first_name char(20) not null,
last_name char(20) not null,
sex char(10),
primary key(last_name, first_name)
);

# 如果我们设置了唯一索引，那么在插入重复数据时，sql语句将无法执行成功，并抛出错

# insert ignore into 与 insert into的区别就是insert ignore会忽略数据库中已经存在的数据，如果数据库没有数据，就插入新的数据，如果有数据的话就跳过这条数据。这样可以保留数据库中已经存在的数据，达到在间隙中插入数据的目的。
# 以下实例使用了insert ignore into，执行后不会出错，也不会向表中插入重复数据：
insert ignore into person_tbl(last_name, first_name)
values('Jay', 'Thomas');

insert ignore into person_tbl(last_name, first_name)
values('Jay', 'Thomas');
# insert ignore into 当插入数据时，在设置了记录的唯一性后，如果插入重复数据，将不返回错误，只以警告形式返回。而replace into如果存在primary或unique相同的记录，则先删除掉，再插入新记录。
# 另一种设置数据的唯一性方法是添加一个unique索引
create table person_tbl(
first_name char(20) not null,
last_name char(20) not null,
sex char(10),
unique (last_name ,first_name)
);
```

统计重复数据

```
select count(*) as repetitions, last_name, first_name
from person_tbl
group by last_name, first_name
having repetitions > 1;
```

以上查询语句将返回person_tbl表中重复的记录数，一般情况下，查询重复的值，执行以下操作：

- 确定哪一列包含的值可能会重复
- 在列选择列表使用count(*)列出的那些列
- 在group by句子中列出的列
- having子句设置重复数大于1

过滤重复数据

如果需要读不重复的数据可以在select语句中使用distinct关键字来过滤重复数据

```
select distinct last_name, first_name
from person_tbl;
# 也可以使用group by来读取数据表中不重复的数据：
select last_name, first_name
from person_tbl
group by (last_name, first_name);
```

删除重复数据

```
create table tmp select last_name, first_name, sex from person_tbl group
by(last_name, first_name, sex);
drop table person_tbl;
alter table tmp rename to person_tbl;

# 当然也可以在数据表中添加index（索引）和primary key（主键）这种简单的方法来删除表中的重复记
录
alter ignore table person_tbl
add primary key(last_name, first_name);
```

MySQL 及 SQL 注入

如果通过网页获取用户输入的数据并将其插入一个MySQL数据库，那么就可能发生SQL注入安全的问题。

所谓的sql注入，就是通过把sql命令插入到web表单提交或输入域名或页面请求的查询字符串，最终达到欺骗服务器执行恶意的sql命令。

我们永远不要信任用户的输入，我们必须认定用户输入的数据都是不安全的，我们都需要对用户输入进行过滤处理。

防止SQL注入，我们需要注意以下几个要点：

- 永远不要新人用户的输入。对用户的输入进行检验，可以通过正则表达式，或限制长度；对单引号和双"-"进行转换等。
- 永远不要使用动态拼装sql，可以使用参数化的sql或者直接使用存储过程进行数据查询存取。
- 永远不要使用管理员权限的数据库连接，为每个应用使用单独的权限有限的数据库连接。
- 不要把机密信息直接存放，加密或者hash掉密码和敏感的信息。
- 应用的异常信息应该给出尽可能少的提示，最好使用自定义的错误信息进行包装

- sql注入的检测方法一般采用辅助软件或网站平台来检测，软件一般采用sql注入检测工具jsky，网站平台就有亿思网站安全平台检测工具。MDCSOFT SCAN等。采用MDCSOFT-IPS可以有效的防御sql注入，XSS攻击等。

MySQL 导出数据

MySQL可以使用select..into outfile语句来见到那的导出数据到文本文件上。

```
select * from runoob_tbl
into outfile '/tmp/runoob.txt';

# 可以通过命令选项来设置数据输出的指定格式,以下实例为导出csv格式
select * from passwd into outfile './tmp/runoob.txt'
fields terminated by ',' enclosed by '"'
likes terminated by '\r\n';
# 生成一个文件，各值用逗号隔开。这种格式可以被许多程序使用
select a,b,a+b, into outfile './tmp/result.txt'
fields terminated by ',' optionally enclosed by'"'
likes terminated by '\n'
from test_table;
```

- load data infile 是select ... into outfile的逆操作，select句法，为了将一个数据库的数据写入一个文件，使用select ... into outfile, 为了将文件读回数据库， 使用load data infile
- select ... into outfile "file_name" 形式的select可以把被选择的行写入一个文件中，该文件被创建到服务器主机上，因此必须有file权限，才能使用此语法
- 输出不能是一个已经存在的文件，防止文件数据被篡改
- 需要一个登录服务器的账号来检索文件，否则select ... into outfile不会起任何作用
- 在unix中，该文件被创建后是可读的，权限由MySQL服务器所拥有。这意味着，虽然可以读取该文件，但可能无法将其删除。

导出表作为原始数据

```
mysqldump -u root -p --no-create-info --tab=/tmp runoob runoob_tbl
# mysqldump是mysql用于转存数据库的程序，它主要产生一个sql脚本，其中包含从头重新创建数据库所必须的命令create table insert等。
# 使用mysqldump导出数据需要使用--tab选项来指定导出文件指定的目录，该目标必须是可写的。
#以上实例将数据表runoob_tbl导出到/tmp目录中；
```

导出SQL格式的数据

```
mysqldump -u root -p runoob_tbl > dump.txt

# 如果需要导出整个数据库的数据，可以使用以下命令：
mysqldump -u root -p runoob > database_dump.txt

# 如果需要备份所有的数据库，可以使用以下命令：
mysqldump -u root -p --all-databases > database_dump.txt

# 将数据拷贝到其他主机，可以在mysqldump命令中指定数据库及数据表
mysqldump -u root -p database_name table_name > dunmp.txt
# 如果完整备份数据库，则无需使用特定的表名称。
# 如果徐奥将备份的数据库导入到MySQL服务器中，可以使用以下命令，使用以下命令你需要确认数据库已经创建：
mysql -u root -p database_name < dump.txt
```

```
# 也可以使用以下命令将导出的数据直接导入到远程的服务器上，但请确保两台服务器是相同的，是可以访问的：
mysqldump -u root -p database_name | mysql -h other-host.com database_name
# 以上使用了管道来将导出的数据导入到指定的远程主机上。
```

MySQL 导入数据

```
mysql -u 用户名 -p 密码 < 要导入的数据库数据（runoob.sql)# mysql 命令导入

# source 命令导入
# source 命令导入数据库需要先登录到数据库终端
create database abc;# 创建数据库
use abc;# 使用自己创建的数据库
set names utf8;# 设置密码
source /home/abc/abc.sql# 导入备份数据库

# 使用LOAD DATA 导入数据
load data local infile 'dump.txt' into table_mytbl;# 将从当前目录中读取文件dump.txt，将该文件中的数据插入到当前数据库的mytbl表中。
# 如果指定loacl关键词，则表明从客户主机上按路径读取文件，如果没有指定，则文件在服务器上按路径读取文件。
# 可以明确在load data语句中指出列值的风格符和行尾标记，但是默认标记是定位符和换行符
# 两个命令的fields和lines子句的语法是一样的，两个子句都是可选的，但是如果两个同时被指定，fields子句必须出现在lines之前。
# 如果用户指定一个fields子句，它的子句(TRRMINATED BY. [OPTIONALLY] ENCLOSED BY和ESCAPED BY)也是可选的，不过，用户必须至少指定他们中的一个。
load data local infile 'dump.txt' into table mytbl
fields terminated by ':'
lines terminated by '\r\n';

# LOAD DATA默认情况下是按照文件中列的顺序插入数据的，如果数据文件中的列与插入表中的列不一致，则需要指定列的顺序。
# 如，在数据文件中的列顺序是a,b,c,但在插入表的列顺序是b,c,a则数据导入语法如下：
load data local infile 'dump.txt'
into table mytbl (b,c,a);

# 使用mysqlimport导入数据
mysqlimport -u root -p --local myttbl dump.txt
# mysqlimport可以指定选项来设置指定格式，命令语句格式如下：
mysqlimport -u root -p --local --field-terminated-by=":" --lines-terminated-by="\r\n" mytbl dump.txt
# mysqlimport 语句中使用columns选项来设置列的顺序：
mysqlimport -u root -p --local --columns=b,c,a mytbl dump.txt
```

选项	功能
-d or --delete	新数据导入数据表中之前删除数据数据表中的所有信息
-f or --force	不管是否遇到错误，mysqlimporot将强制继续插入数据
-i or --ignore	mysqlimport跳过或者忽略那些有相同唯一关键字的行，导入文件中的数据将被忽略
-l or -lock-tables	数据被插入到之前锁住表，这样就放置了，你在更新数据库时，用户的查询和更新收到影响
-r or -replace	这个选项与-i选项的作用相反；此选项将替代表中有相同唯一关键字的记录
-field-enclosed-by=char	指定文本文件中数据的记录时以什么括起的，很多情况下数据以双引号括起，默认情况下数据是没有被字符括起的
--field-terminated-by=char	指定各个数据之间的分割符，在句号分隔的文件中，分隔符是句号，可以用此选项指定数据之间的分隔符。默认的分隔符是跳格符（Tab）
--lines-terminated-by=str	此选项指定文本文件中行与行之间数据的分隔字符串或者字符。默认情况下mysqlimport以newline为行分隔符。可以选择用一个字符串来替代一个单个的字符：一个新行或者一个回车。

mysqlimport命令常用的选项还有-v显示版本（version），-p提示输入密码（password）等。

MySQL函数

MySQL 字符串函数

函数	描述	实例
ASCII(s)	返回字符串s的第一个字符的ASCII码	select ascii(CustomerName) as NumCodeOfFirstChar from Customers;
char_length(s)	返回字符串s的字符数	select char_length('runoob') as lengthofstring;
character_length(s)	返回字符串s的字符数	select character_length('runoob') as lengthofstring;
concat(s1,s2..sn)	字符串s1,s2等多个字符串合并为一个字符串	select concat('s','q','l') as concatenatedString;
field(s,s1,s2)	返回第一个字符串s在字符串列表（s1,s2）中的位置	select field('a', 'av', 'ab','cad')
format(x,n)	函数可以将数字x进行格式化"#,###.##",将x保留到小数点后n位，最后一位四舍五入	select format(250500.5634, 2);# 输出250,500.56
find_in_set(s1,s2)	返回字符串s2中与s1匹配的字符串的位置	select find_in_set("c", "a,b,c,d,e");
insert(s1,x,len,s2)	字符串s2替换s1的x位置开始长度为len的字符串	select insert("google.com", 1, 6, 'runoob');# 输出结果 runoob.com

函数	描述	实例

MySQL 数字函数

函数名	描述	实例
abs(x)	返回x的绝对值	select abs(-1)
acos(x)	求x的反余弦值	select acos(0.25)`

函数名	描述	实例

MySQL 日期函数

函数名	描述	实例
adddate(d,n)	计算起始日期d加上n天的日期	select adddate("2017-06-15", interval 10 day);# 输出 2017-06-25

函数名	描述	实例

MySQL 高级函数

函数名	描述	实例
bin(x)	返回x的二进制编码	select bin(15); # 输出 1111

函数名	描述	实例

MySQL 运算符

算数运算符

运算符	作用
+	加法
-	减法
*	乘法
/或div	除法
% 或 MOD	取余

比较运算符

符号	描述
=	等于
<>, !=	不等于
>	大于
<	小于
<=	小于等于
>=	大于等于
between	在两值之间
not between	不在两值之间
in	在集合中
not in	不在集合中
<=>	严格比较两个null值是否相等
like	模糊匹配
regexp或rlike	正则式匹配
is null	为空

逻辑运算符

运算符号	作用
not 或 !	逻辑非
and	逻辑与
or	逻辑或
xor	逻辑异或

异或运算相同值结果为 0， 不同值结果为 1；

位运算符

运算符号	作用
&	按位与
|	按位或
^	按位异或
!	取反
<<	左移
>>	右移

运算符优先级

最低优先级为：**:=**

最高优先级为：**!**、**BINARY**、**COLLATE**。