# **mysql基本语法大全**

## 1.备份数据库：

### 1.1备份数据库中的表:

mysqldump -u root -p test a b >d:\bank\_a.sql

//分别备份数据库test下a和b表

### 1.2备份一个数据库

mysqldump -u root -p test > d:\testbk.sql

### 1.3备份多个数据库

mysqldump -u root -p --databases test mysql > D:\data.sql

### 1.4备份所有的数据库

mysqldump -u -root -p --all-databases > D:\all.sql

### 1.5直接复制整个数据库目录（物理备份）

前提条件是停止mysql服务，然后复制mysql下的data目录下数据库目录。

### 2.还原数据：

### 2.1使用mysql命令恢复

还原数据库文件:

mysql -u root -p < d:\backup.sql

还原数据库表文件：

mysql -u root -p test <d:\backup.sql

### 2.2使用source命令恢复数据

还原数据库文件：

mysql>source d:\testbk.sql

还原数据库表文件：

mysql>use test

mysql>source d:\testbk.sql

### 2.3先停止mysql服务，然后拷贝备份的整个test数据库目录到目标目录。

### 2.4网络上远程还原数据可以用

mysqldump -h x.x.x.x -u root -p test >tesbk.sql

### 2.5mysql忘记密码：

1.停止mysql服务

net stop mysql或者相应的进程

2.进入mysql下bin目录：

mysqld --skip-grant-tables //跳过验证登录

3.另外窗口打开：

mysql //直接可用进入系统

4.更改密码

use mysql;

update user set password=password('123456') where user='root' and host='localhost';

5.注销系统，再进入，开MySQL，使用用户名root和刚才设置的新密码123456登陆

### 2.6修改密码：

1.你的root用户现在没有密码，你希望的密码修改为123456，那么命令是：

mysqladmin -u root password 123456

2.如果你的root现在有密码了（123456），那么修改密码为abcdef的命令是：

mysqladmin -u root -p password 123456

### 2.7添加用户：

grant select,insert,update,delete,create,drop on stud.\* to user1@localhost identified by "user1"; //添加用户名user1密码为user1具有插入，更新，删除，创建，删除对于数据库所有表

GRANT ALL PRIVILEGES ON \*.\* TO 'backlion'@'%' IDENTIFIED BY 'backlion123' WITH GRANT OPTION;

grant all privileges on \*.\* to test@loclhost identified by "test";

创建主键：

Alter table test add primary key(code,curlum); //用code和curlum作为一组联合主键来约束

### 3.数据库操作：

### 3.1显示数据库

show databases;

### 3.2选择数据库

use examples;

### 3.3创建数据库并设置编码utf8 多语言

create database bk default character set utf8 collate utf8\_general\_ci;

### 3.4修改数据库字符集为utf8

use mysql;

alter database character set utf8;

### 3.5删除数据库

drop database bk;

### 3.6查看数据库状态：

status;

### 4.数据表的操作：

### 4.1显示数据表

show tables;

### 4.2查看数据表的结构属性（字段，类型）

describe test;

desc test;

### 4.3复制表结构（里面没有数据，结构一样）

create table newtest like oldtest; //创建新表newtest和旧表oldtest数据表结构一样

### 4.4复制表中的数据

insert into nettest select \* from oldtest; //将旧表oldtest的数据复制到新表newtest里面

### 4.6重命名表名

alter table old\_name rename new\_name

### 4.7显示当前mysql版本和当前日期

select version(),current\_date;

### 4.8创建表：

create table bk(

id int(10) unsigned zerofill not null auto\_increment,

email varchar(40) not null,

ip varchar(15) not null,

state int(10) not null default '-1',

primary key (id)

);

/\*

1.常用的数据类型为int,varchar,date,text这4个数据类型

2.字段（行）的属性有：数据类型（数据长度） 是否为空 是否为主键 是否为自动增加 默认值

如：int(25) not null primary key auto\_increment default 12

3.每个字段之间用逗号分开，最后那个字段不需要用逗号

4.以分号结束

5.可以设置简单数据表结构如：

字段名数据类型 数据长度 是否为空 是否主键是否自动增加 默认值

id int 12 NOT NULL primary key auto\_increment

name varchar 30NOT NULL

password varchar30 NOT NULL

time date 30 NOT NULL

jianyi text 400

\*/

### 4.9删除数据表：

drop table bk; //包括结构和数据都删除

### 10.数据库字段的操作：

### 11.1添加表字段

alter table test add bk varchar(32) not null; //向表test中添加bk字段（列）

### 11.2修改表字段

alter table test change id id1 varchar(10) not null; //将表test中字段（列）id更改为id1

### 11.3删除字段（列）

alter table test drop cn;

### 11.4插入表数据

insert into test (id11,email,ip,state,bk)value(2,'601462930@qq.com','10.192.16.12',1314,567);

//如果是字符型对应的值需要用单引号引起来，数字型不需要

### 11.5删除数据

delete from test //删除整个表test的数据，结构保留

delete from test where id11=2; 删除数据表来自某个主键字段。就等于删除整条数据

### 11.6修改表字段数据信息(数据）

Update table\_name set 字段名=’新值’ [, 字段2 =’新值’ , …..][where id=id\_num] [order by 字段 顺序]

update test set email='895098355@qq.com' where id11=2;

### 11.7修改字段的属性（数据类型）

alter table test modify class varchar(30) not null; //修改表test中字段class的属性为varchar(30)

### 12.查询数据：

### 12.1查询所有数据：

select \* form test;

### 12.2查询两个字段的数据

select id,number from test;

### 12.3查询前2行数据：

select \* from test limit 0,2;

### 12.4按增序排列查询

select \* from test order by id11 asc;

select \* from test order by id11 //默认为增序查询

### 12.5按降序排列查询

select \* from test order by id11 desc;

### 12.6模糊查询

select \* from test where email '%qq%'; //查询test表中，条件是email的数据中包含qq的数据

### 12.7查询某个字段下面的数据

select email as emaildata from test ; //选择email字段作为emalidata统计显示出的数据

### 12.8.条件查询

select \* form test where id=12;

### 13.多表查询：

### 13.1用法一：where条件联合查询

select 表1.字段 [as 别名],表n.字段 from 表1 [别名],表n where 条件;

select testA.username as username,testB.id from testA,testB where testA.uid=testB.uid

### 13.2用法二：inner join on 条件联合查询

select 表1.字段 [as 别名],表n.字段 from 表1 INNER JOIN 表n on 条件;

select testA.username as uername ,testB.id from testA inner join testB on testA.uid=testB.uid

### 13.3记录联合:

select语句1 union[all] select语句2

select \* from testA union select id from testB;

1. #创建数据库#   **create**  **database** 数据库名;
2. #查看数据库#     show databases;
4. #选择数据库#      use 数据库名;
6. #删除数据库#      **drop** **database** 数据库名;


10. #创建表#  **create** **table** 表名(属性名1  数据类型 ，属性名2  数据类型。。。。);
12. #查看表结构# **desc** 表名;
14. #查看建表语言#  show **create** **table** 表名;
16. #表中新增字段#  **alter** **table** 表名 **add**( 属性名1 数据类型，属性名2 数据类型.....);
18. #在表的第一个位置增加字段#  **alter** **table** 表名 **add** 属性名 数据类型 **first**;
20. #在指定字段后面增加字段#   **alter** **table**  表名 **add**  新增属性名  数据类型  **after**  属性名;
22. #删除表中字段#  **alter** **table** 表名 **drop**  属性名;
24. #删除表#   **drop** **table** 表名;


28. #修改表名# **alter** **table** 旧表名 rename  新表名; 或**alter** **table** 旧表名 rename **to** 新表名;
30. #修改表中字段的数据类型#  **alter** **table** 表名 **modify**  需要修改的属性名   想要修改成的数据类型;
32. #修改表中字段名称 **alter**  **table** 表名 change 旧属性名  新属性名 旧数据类型;
34. #修改表中字段名称和数据类型  **alter**  **table** 表名 change 旧属性名 新属性名 新数据类型;
36. #修改表中字段为头字段#  **alter**  **table**  表名 **modify**  属性名 数据类型   **first**;
38. #修改字段1为顺序，在字段2后面#   **alter** **table** 表名 **modify**  属性名1  数据类型 **after** 属性名2;


42. #插入数据记录#  **insert** **into** 表名(字段1,字段2,.....或不写)   **values**(一条数据),(一条数据);
44. #插入查询结果# **insert**  **into**  要插入表表名 (插入表中字段1，插入表中字段2,.....)   **select** (查询表中字段1,查询表中字段2......)  **from** 要查询表表名    查询语句;
46. #更新数据记录#  **update** 表名  **set** 字段1=新值， 字段2=新值.........**where**  查询语句；
48. #删除数据记录#   **delete** **from**表名 **where**  查询语句；
50. #查询部分数据记录#  **select**    字段1，字段2。。。。 **from**   要查询的表；
52. #避免重复查询#  **select**   **distinct**    字段1，字段2。。。。 **from**   要查询的表；
54. #为查询结果起别名# **select**    字段1  **as**或不写    别名1，字段2  **as**或不写  别名2。。。。 **from**   要查询的表；
56. #设置显示格式数据查询#  **select**   concat（字段1，‘提示语句’，字段2。。。。） **from**  要查询的表；
58. #**where**条件查询#  **select**  \* **from**   表名  **where** 字段=值；
60. #between and 关键字范围查询#   **select** \*  **from**  表名  **where**  字段   between  值1  and  值2；
62. #between and  关键字不再范围内查询#  **select**  \*  **from**   表名 **where** 字段 not  between 值1  and  值2 ；
64. #带null关键字空查询#    **select**  \*  **from** 表名   **where**  字段  **is**  not   null；    或   **select**  \*  **from** 表名   **where**  字段  **is**  not   null；
66. #带in关键字的集合查询#   **select**  字段1  ，字段2.。。。。。**from**  表名    **where**  字段n     in（值1，指2。。。。。）；
68. #带like关键字的模糊查询#   **select** \*  **from** 表名  **where**   字段 like  ‘字段串%’ ；       或**select** \*  **from** 表名  **where**   字段 like  ‘字段串\_’ ； 或    **select** \*  **from** 表名  **where**    not  字段 like  ‘字段串\_’ ；
70. #排序数据查询#   **select** \*  **from** 表名 **order**  **by**  依照排序的字段名   **asc**（升序）  或 **desc** （降序）；
72. #多字段排序#   **select** \*  **from** 表名 **order** **by**  字段1 **asc** ，字段2  **desc** 。。。。；
74. #带limit关键字的限制查询数量#  **select** \*  **from** 表名  **where**     limit  开始条数位置 ，显示条数；
76. #分组数据查询#   **select**   \*   **from**    表名  **group**  **by**  字段；  #随机显示出每个分组的一条数据，一般讲分组与统计合起来使用才有意义#
78. #带**having**的分组限定查询#   **select** \*  **from** 表名 **group**   **by**   字段  **having**   条件；

81. #**inner**  join  **on**内连接#  **select** \* **from**   表1   **inner**   join   表2   **on** 条件；
83. #自连接# **select**  e.字段   **as**  别名1，f.字段 **as** 别名2.。。。。**from**     表名    **as** e  **inner** join 表  **as** f   **on**  条件；
85. #**from**多表连接#  **select** \*  **from** 表1 ，  表 2 。。。    **where**   条件；
87. #左外连接# **select** \*  **from**   表1   left    join    表2   **on** 条件；
89. #右外连接#   **select** \*  **from**    表1  right    join  表2    **on**    条件；
91. #允许重复值的合并查询#  **select** \* **from**    表1   **union** all    表2    **on**   条件；
93. #不含重复值的合并查询#  **select** \*  **from**  表1   **union**  表2  **on**  条件；
95. #**where**型子查询#   **select** \*  **from** 表名   **where** （字段1，字段2）=（**select**   字段1，  字段2  **from**  表名  **where** 。。。。）；
97. #in关键字的子查询#   **select**  \*  **from**  表名  **where**  字段  in （**select** 。。。。查询语句）；
99. #any关键字的子查询#   **select**  \*  **from**  表名  **where**  字段  >=any （**select** 。。。。查询语句）；
101. #all关键字的子查询#   **select**  \*  **from**  表名  **where**  字段   <=  all（**select** 。。。。查询语句）；
103. #exists关键字的子查询#   **select**  \*  **from**  表名  **where**   not exists （**select** 。。。。查询语句）；
105. #regexp正则表达式运算符#  **select**  ‘chshs’   rehexp  ‘c.’  ；





112. #计数函数#  count（）；
114. #求平均函数#  avg（）；
116. #求和函数#   sum（）；
118. #取最大函数#   **max**（）；
120. #取最小函数# **min**（）；
122. #取绝对值# abs（）；
124. #取大于x的最大整数# cell（x）；
126. #取小于x的最大整数# floor（x）；
128. #取数值x的四舍五入后有y为小数# round（x，y）；
130. #直接截取x为有y位的小数# **truncate**（x，y）；
132. #返回0~1之间的一个随机数#rand（）；
134. #获取第n组返回相同值得随机数#  rand（n）；

137. #对字符串进行加密#   **password**（）；
139. #字符串连接函数#  concat（字符串1，字符串2.。。）；
141. #带分隔符字符串合并#   concat（分隔符，字符串1，字符串2.。。。）；
143. #返回在字符串str2。。。中与str1相匹配的字符串位置#   find\_in\_set（‘str1’，‘str2，str3.。。。。’）；
145. #返回字符串str1第一次出现位置#   field（‘str1’，‘str2’，‘str3’。。。。）；
147. #返回子字符串str在字符串str1中匹配开始的位置#  locate（str，str1）；或   position（str  in str1）；  或   instr（str1  ，str）；
149. #返回第n'个字符串#  elt（n，str1，str2.。。strn）；
151. #截取指定位置和长度的字符串#  substring（str，num，length）；  或  mid（str，num，length）；
153. #将字符串str从第x位置开始，y个字符长的字串替换为字符串str2#     **insert**（str ， x ，y，str2）；
155. #将字符变为小写#  lower（str）；或        lcase（str）
157. #将字符变为大写#   upper（str）；或       ucase（str）
159. #获取字符串长度# length（str）；
161. #获取字符数函数# char\_length（str）；
163. #返回字符串str中最左边的x个字符#   left（str，x）；
165. #返回字符串str中最右边的x个字符#    right（str，x）；
167. #使用字符串pad对字符串str最左边进行填充，知道长度为n个字符长度#    lpad（str，n，pad）；
169. #使用字符串pad对字符串str最右边进行填充，知道长度为n个字符长度#    rpad（str，n，pad）；
171. #去掉字符串str左边的空格#   ltrim（str）；
173. #去掉字符串str右边的空格#   rtrim（str）；
175. #返回字符串str重复x次的结果#   repeat（str，x）；
177. #使用字符串b代替字符串str中的所有字符串a#    replace（str，a，b）；
179. #比较字符串str1和str2#  strcmp（str1，str2）；
181. #去掉字符串str行头和行尾的空格#   trim（str）；
183. #返回字符串str中从x位置起y个长度的字符串#   substring（str，x，y）；

186. #获取当前日期# curdate（）； 或  **current\_date**（）；
188. #获取当前时间# curtime（）； 或  **current\_time**（）；
190. #获取当前日期和时间# now（）；或 current\_timestamp（）  或  localtime（）  或  systemdate（）；
192. #获取日期**date**的UNIX时间戳#   unix\_timestamp（**date**）；
194. #获取unix时间戳的日期值#   from\_unixtime（）；
196. #返回日期**date**为一年中的第几周#    week（**date**）； 或 weekofyear（**time**）；
198. #返回日期的英文周几#  dayname（**time**）；
200. #返回日期和时间中周几（1：周日，2：周一）#  dayofweek（）；
202. #返回日期和时间中周几（0：周一，1：周二）#  weekday（）；
204. #返回年中第几天# dayofyear（**time**）；
206. #返回月中第几天# dayofmonth（**time**）；
208. #返回日期**date**的年份#  year（**date**）；
210. #返回时间**time**的小时值#   **hour**（**time**）；
212. #返回时间**time**的分钟值#  **minute**（**time**）；
214. #返回时间**time**的月份值#  monthname（**date**）； 或 month（**time**）
216. #截取日期中的各部分值# extrcat（年或月或日或时或分或秒      **from**  **time**）；
218. #计算date1与date2之间相隔天数#   datediff（date1，date2）；
220. #计算**date**加上n天后的日期#    adddate（**date**，n）；
222. #计算**date**减去n天后的日期#   subdate（**date**，n）；
224. #计算**time**加上n秒后的时间#    adddate（**time**，n）；
226. #计算**time**减去n秒后的时间#    subdate（**time**，n）；

229. #返回数据库版本号#  version（）；
231. #返回当前数据库名#  **database**（）；
233. #返回当前用户#  user（）；
235. #将IP地址转化为数字#  inet\_aton（ip）；
237. #将数字转化为IP地址#   inet\_ntoa（x）；
239. #创建一个持续时间为**time**的名为**name**的锁#     cet\_loct（**name**，**time**）；
241. #为名为**name**的锁解锁# release\_loct（**name**）；
243. #将表达式重复执行count次#  benchmark（count，表达式）；
245. #将x变为type形式#  convert（x，type）；





252. #设置字段的非空约束# **create** **table** 表名 (属性名 数据类型  not null);
254. #设置字段的默认值#  **create** **table** 表名 (属性名  数据类型  **default**  默认值);
256. #设置字段的唯一约束# **create** **table** 表名(属性名  数据类型  **unique** );
258. #设置字段的唯一约束并未约束命名#    **create**  **table** 表名(属性名1 数据类型 ， 属性名2     数据类型  ..........        **constraint**   约束名  **unique** (属性名1，属性名2......));
260. #设置单字段为主键约束#  **create**  **table** 表名(属性名1  数据类型   **primary**  **key**....);
262. #设置多字段为主键约束#   **create**  **table**  表名(属性名1   数据类型  ,  属性名2  数据类型........**constraint**   约束名  **primary**  **key** (属性名1，属性名2......));
264. #设置字段自动增加值#      **create** **table** 表名 (属性名   数据类型    auto\_increment.........);
266. #设置外键约束#    **create** **table** 表名 (属性名1    数据类型  ，   属性名2   数据类型........      **constraint**   外键约束名  foreing  **key** (外键属性名1)    **references**    表名 (主键属性名2));


270. #创建普通索引#  **create** **table** 表名(属性名  数据类型 ，属性名  数据类型.....   **index**或 **key** 索引名（可省略）(属性名  (长度（可省略）)    **asc**或**desc**);
272. #在已存在表创建普通索引# **create** **index** 索引名  **on** 表名  ( 属性名 (长度或不写)   **asc**或**desc**或不写);  或
274. **alter**   **table**  表名  **add**  **index**或**key**   索引名(属性名 (长度或不写)   **asc**或**desc**或不写);
276. #创建唯一索引#     **create** **table** 表名(属性名  数据类型 ，属性名  数据类型..... **unique**    **index**或 **key** 索引名（可省略）(属性名  (长度（可省略）)    **asc**或**desc**);
278. #在已存在表创建唯一索引# **create**  **unique**   **index** 索引名  **on** 表名  ( 属性名 (长度或不写)   **asc**或**desc**或不写);  或
279. **alter**   **table**  表名  **add**    **unique**  **index**或**key**   索引名(属性名 (长度或不写)   **asc**或**desc**或不写);
281. #创建全文索引#  **create** **table** 表名(属性名  数据类型 ，属性名  数据类型..... fulltext    **index**或 **key** 索引名（可省略）(属性名  (长度（可省略）)    **asc**或**desc**);
283. #在已存在表创建全文索引# **create**  fulltext   **index** 索引名  **on** 表名  ( 属性名 (长度或不写)   **asc**或**desc**或不写);  或
284. **alter**   **table**  表名  **add**    fulltext  **index**或**key**   索引名(属性名 (长度或不写)   **asc**或**desc**或不写);
286. #创建多列索引#  **create** **table** 表名(属性名  数据类型 ，属性名  数据类型.....   **index**或 **key** 索引名（可省略）(属性名1  (长度（可省略）)    **asc**或**desc** ,属性名2  (长度（可省略）)    **asc**或**desc**.........);
288. #在已存在表创建多列索引# **create**   **index** 索引名  **on** 表名 (属性名1  (长度（可省略）)    **asc**或**desc** ,属性名2  (长度（可省略）)    **asc**或**desc**.........);  或
289. **alter**   **table**  表名  **add**     **index**或**key**   索引名(属性名1  (长度（可省略）)    **asc**或**desc** ,属性名2  (长度（可省略）)    **asc**或**desc**.........);;
291. #查看索引是否用到#  explain  **select** \* **from** 表名  **where**  .......;
293. #删除索引#   **drop** **index**   索引名 **on** 表名;


297. #创建视图#  **create** **view**  视图名 **as**   查询语句;
299. #查看视图详细信息#  show    **table**   status   **from**  数据库名  like '视图名';       或    show  **table**   status
301. #查看视图定义信息#   show  **create**  **view**  视图名;
303. #查看视图设计信息#   **desc**  视图名;
305. #通过系统表查看视图信息#   use  information\_schema ;    **select** \* **from** views **where** table\_name='视图名'\G;
307. #删除视图#  **drop** **view**  视图名;
309. #修改视图#  **create** or replace **view** 视图名 **as** 查询语句;   或    **alter**  **view**   视图名  **as**  查询语句 ;


313. #创建触发器# **create** **trigger**   触发器名 before或**after**   触发条件(**delete**、**insert**、**update**) **on**  触发条件的操作表表名  **for**   each   row  触发语句;
315. #创建含多条语句的触发器# delimiter $$ **create** **trigger**   触发器名 before或**after**   触发条件(**delete**、**insert**、**update**) **on**  触发条件的操作表表名  **for**   each   row  **begin** 触发语句1; 触发语句2；......；**end** $$  delimiter;
317. #查看触发器#  show triggers\G
319. #通过查看系统表查看触发器信息# use information\_schema;  **select** \* **from** triggers\G
321. #删除触发器# **drop** **trigger** 触发器名字


325. #查看错误信息#   show warnings;
327. #查看支持的存储引擎#   show engines;   或 show    variables   like 'have%';
329. #查看默认存储引擎#    show      variables    like   'storage\_engine% ';
331. #查看MySQL的帮助文档目录列表# help  contents;


335. #查看数据类型# help data  types;
337. #显示当前年月日# **select** curdate();
339. #显示当前年月日和时间#  **select** now();
341. #显示当前时间# **select** **time**(now());
343. #显示当前年月日#  **select** year (now()) ;


347. #创建存储过程#  **create** **procedure**   存储过程名字  （存储过程参数：输入/输出类型，参数名，参数类型）  存储过程特性或不写     存储过程的语句；
349. #创建函数过程#   **create** **function** 函数名字 （函数的参数：参数名，参数类型）   函数特性或不写   函数过程语句；
351. #查看存储过程#  show **procedure** status like '存储过程名' \G   或  use information\_schema；  **select** \* **from** routines **where** specific\_name='存储过程名'\G
353. #查看函数过程#    show **function** status like '函数过程' \G或  use information\_schema；  **select** \* **from** routines **where** specific\_name='函数过程名'\G
355. #查看存储过程定义信息#  show   **create**    **procedure**  存储过程名\G
357. #查看函数过程定义信息#   show crate  **function**  函数名\G
359. #删除存储过程#   **drop**  **procedure**    存储过程名\G
361. # 删除函数#   **drop**   **function**  函数名\G
363. #创建普通用户账户#  **create** user 用户名 identified    **by**  ‘密码’；
365. #创建带权限的普通用户账户#   **grant**  权限参数：**select**、**create**、**drop**等  **on**  库.表（权限范围）   **to**  用户名 identified    **by**  ‘密码’；
367. #更改超级用户root的密码#    **set**  **password**=**password**（“新密码”）；
369. #利用root用户修改带权限的普通用户密码# **grant**  权限参数：**select**、**create**、**drop**等  **on**  库.表（权限范围）   **to**  用户名 identified    **by**  ‘新密码’；
371. #利用root用户修改普通用户密码#**set**  **password**    **for**   用户名=**password**（“新密码”）；
373. #普通用户更改自身密码#**set**  **password**=**password**（“新密码”）；
375. #删除普通用户#   **drop**  user 用户名；  或  **delete** **from**  user   **where**  user=“用户名”；
377. #对普通用户进行授权#  **grant**  权限参数：**select**、**create**、**drop**等  **on**  库.表（权限范围）   **to**  用户名；
379. #收回权限#   **revoke**  权限参数：**select**、**create**、**drop**等  **on**  库.表（权限范围）   **from**  用户名；
381. #收回所有权限#   **revoke** all **privileges**,**grant** **option** **from**  用户名;

## [mySql 基本语法学习笔记](http://www.cnblogs.com/yxt520/p/4754961.html)

 create database if not exists yang;

 drop database if exists yang;

 show databases;

 show databases like 'yang%';

  create table yang\_tableOne(name varchar(50),age int(20),address varchar(50));

show tables from yang;

show tables from yang like 'yang\_%';

use yang;

create table yang\_tableTwo(id int null default '444',age int(40) null default '20');

desc yang\_tableOne;

describe yang\_tableOne;

show columns from yang\_tableOne;

insert into yang\_tableOne values('yanger','29','beijing'), ('wangwu','23',' nanjing');

insert yang\_tableOne set name='zhaosi', age='22',address='wuhan'; //只能设置一条数据

update yang\_tableOne set age='24' where name='yanger';

delete from yang\_tableOne where name='yanger';

select \* from yang\_tableOne;

select  id, age  from yang\_tableOne where age='40';

select (20+2\*10)/4; //求值

create table yang\_tableTwo (id int not null auto\_increment primary key, weight int null default '250');

insert into yang\_tableTwo values('1','23');

insert  yang\_tableTwo values();//会插入一个（2，null）

insert  yang\_tableTwo values();//会插入一个（3，null）

insert  yang\_tableTwo values();//会插入一个（4，null）

create table yang\_tableThree  select \* from yang\_tableTwo;//复制一个表结构

第三种语法表示：查看某个数据库表中的索引；例如

1. show index from yang\_tableTwo from yang;

## **alter修改表结构**

有时你可能需要改变一下现有表的结构，那么alter table语句将是你的合适选择。

增加列

1. alter table tbl\_name add col\_name type

例如增加一个weight列:

1. alter table test01\_01 add weight int;

修改列

1. alter table test01\_01 modify weight varchar(50);

删除列

1. alter table test01\_01 drop weight;

另外一种常用方式，例如:

给列更名

1. alter table test01\_01 change weight wei int;

给表更名

1. alter table test01\_01 rename test01\_04;

## **drop删除数据库表**

drop table删除一个或多个数据库表。所有表中的数据和表定义均被删除，故小心 使用这个命令！  
在MySQL 3.22或以后版本，你可以使用关键词 if exists 类避免不存在表的一个错 误发生。语法如下:

1. drop table [if exists] tbl\_name [, tbl\_name,...]

例如删除test01\_04这个表,要删除多个表，在后面用逗号隔开即可。例句：

1. drop table if exists test01\_04;

## **insert添加数据**

前面我们学习了数据库，数据库表，接下来我们学习一下操作表中的数据。

让我们开始利用 insert 语句来增加记录，这是一个 SQL 语句，需要为它指定希望插入数据行的表或将值按行放入的表。如果你设置了主键并且自动增长，添加的时候需要把对应的主键值设置为null或者数字“0”。

基本语法：

1. insert [into] tbl\_name [(col\_name,...)] VALUES (pression,...)
2. insert [into] tbl\_name SET col\_name=expression

“into”一词自 MySQL 3.22.5 以来是可选的。（这一点对其他形式的insert 语句也成立。）values表必须包含表中每列的值，并且按表中列的存放次序给出。（一般， 这就是创建表时列的定义次序。如果不能肯定的话，可使用describe tbl\_name来查看这个次序。）。 看一下实例：

可指定所有列的值：

1. insert test01\_01 values(null,'jack','23','beij');

使用多个值表，可以一次提供多行数据：

1. insert test01\_01 values('jack','23','beij'),('grey','34','ah');

可以给出要赋值的那个列，然后再列出值。这对于希望建立只有几个列需要初始设置 的记录是很有用的：

1. insert test01\_01(name) values('gery');
2. insert test01\_01(name) values('mack'),('rose');
3. insert test01\_01 set name='sara';//使用这种形式的 insert 语句不能插入多行

## **insert...select添加数据**

当我们在上一节学习创建表时，知道可以使用select从其它表来直接创建表，甚至可以同时复制数据记录。如果你已经拥有了一个表，你同样可以从select语句的配合中获益。

基本语法：

1. insert into tb\_name select \* from tb\_name1;//表结构完全一致
2. insert into tb\_name(col1) select col1 from tb\_name1;//取某个值添加

insert into ... select语句满足下列条件:

1:查询不能包含一个order by子句

2:insert语句的目的表不能出现在select查询部分的from子句，因为这在ANSI SQL中被禁止让从你正在插入的表中select.问题是select将可能发 现在同一个运行期间内先前被插入的记录。当使用子选择子句时，情况能很容易混淆

看下面实例：

1. insert into test01\_03 select \* from test01\_01;
2. insert into test01\_03(name) select name from test01\_01;

  eate table yang1\_three (id int not null auto\_increment pr

## **select查询**

除非最终检索它们并利用它们来做点事情，否则将记录放入数据库没什么好处。这就 是select语句的用途，即帮助取出数据。select大概是SQL语言中最常用的语句，而且怎样使用它也最为讲究；用它来选择记录可能相当复杂，可能会涉及许多表中列之间 的比较。本节介绍select语句关于查询的最基本功能。 select语句的语法如下：

1. select selection\_list //选择哪些列
2. from table\_list //从何处选择行
3. where primary\_constraint //行必须满足什么条件
4. group by grouping\_columns // 怎样对结果分组
5. having secondary\_constraint //行必须满足的第二条件
6. order by sorting\_columns //怎样对结果排序
7. limit count //结果限定

注意：所有使用的关键词必须精确地以上面的顺序给出。例如，一个having子句必 须跟在group by子句之后和order by子句之前

上述语法中除了select关键字之外，其他每样东西都是可选的。有些数据库需要用from关键字，但是MySQL允许对表达式求值而不引用任何表im

## **普通查询**

select最简单的形式是从一张表中检索每样东西

基本语法：

1. select columns from tb\_name;

columns是列名称，可以写成\*号，表示查询所有列；也可以指定某个列。tb\_name是你的目标数据库表。

看两个例子：

1. select <em> from test01\_01;//查询当前表里所有的列的数据
2. select name,age from test01\_01;//查询当前表里name列和age列的所有数据
3. select (2+3</em>4.5)/2.5;//表达式求值而不引用任何表

不必每次查询都返回所有的行记录，你能从你的表中只选择特定的行。你可以使用where或者having，这里我们只讲where语句，因为where能实现having绝大部分功能

基本语法：

1. select columns from tb\_name where expression..;

expression的表现形式很多，例如name='jack',age>=20,name='jack' or name ='tom'等等，都可以写在where表达式后面。

看一个例子：

1. select \* from test01\_01 where name ="tom";//筛选出当前表中name的值是tom的所有数据。

## **查询排序**

使用order by子句对查询返回的结果按一列或多列排序。order by子句的语法 格式为：

1. order by column\_name [asc|desc] [,…]

其中asc表示升序，为默认值，desc为降序。order by不能按text、text和image数据类型进行排序。另外，可以根据表达式进行排序

看下面实例：

1. select \* from test01\_01 order by age asc;//按照年龄升序输出

注意desc/asc关键词仅适用于紧跟在它之前的列名字

从选用test01数据库，test01\_01表中查询结果，并按照年龄降序显示

## **分组和行计数**

group by从句根据所给的列名返回分组的查询结果，可用于查询具有相同值的列

基本语法：

1. group by col\_name，….

单独使用group by没有任何意义，它的真正作用在于与各种组合函数配合，用于行计数

看下面实例：

1. select name,count(\*) from test01\_01 group by name;//按name分组，并且统计数量

如果你除了计数还返回一个列的值，那么必须使用group by语句，否则无法计算记录,产生错误。

## **多表查询**

查询多个表，from子句列出表名，并用逗号分隔，因为查询需要从他们两个拉出信息

基本语法：

1. select t1.name,t2.name from tb\_name1 as t1,tb\_name2 as t2 where t1.name=t2.name;

上述代码表示，从t1和t2中取出两张表name相同的值。as表示为这个表取别名。

实例如下：

1. select t1.name as t1name,t1.age as t1age,t2.name as t2name,t2.age as t2age
2. from test01\_01 as t1,test01\_03 as t2 where t1.name=t2.name;

选用test01数据库，查询出test01\_01表中与test01\_02表中地址相同的数据

## **update更新数据**

修改数据我们用update更新，这是最常用的更新数据语句，我们看下语法

1. update tbl\_name SET 要更改的列
2. where 要更新的记录

这里的 where 子句是可选的，因此如果不指定的话，表中的每个记录都被更新。

实例如下:

1. update test01\_03 set name='tom' where age= '20';

选用test01数据库，修改test01\_01表中地址是“USA”的修改成地址是“CHINA”

## **delete删除数据**

删除一条语句，我们常用delete关键字。语法定义：

1. delete from tbl\_name where 要删除的记录

where子句指定哪些记录应该删除。它是可选的，但是如果不选的话，将会删除所有的记录。这意味着最简单的delete语句也是最危险的

实例如下:

1. delete from test01\_01 where age=10;//删除年龄为10的数据

请选用test01数据库，删除test01\_01表中名字是jery的数据

ary key,weigh

t int);

te table yang1\_three (id int not null auto\_increment primary key,weigh

t int);