**[MySQL基础](http://www.cnblogs.com/xinsiwei18/p/5846940.html)**

**一、概述**

* MySQL是一个软件，主要为两个功能：

1. 将数据保存到文件或内存
2. 接收特定的命令，然后对文件进行相应的操作

　　有了MySQL软件，无须自己再去创建文件和文件夹，而是直接传递命令给上述软件，让其来进行文件操作

* MySQL接收的操作命令就是SQL语句（结构化语言），是一种专门用来与数据库通信的语言
* **数据库主要优化点是查询优化和存储资源节省，比如选择innodb的引擎，查询速度是稍次的，但是innodb支持事务，支持原子性操作，还有定长char和变长varchar，虽然varchar节省存储资源，但是却是降低了搜索速度**

**二、下载安装**

想要使用MySQL来存储并操作数据，则需要做几件事情：  
*a. 安装MySQL服务端*  
*b. 安装MySQL客户端*  
*b. 【客户端】连接【服务端】*  
*c. 【客户端】发送命令给【服务端MySQL】服务的接受命令并执行相应操作(增删改查等)*

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | 下载          http://dev.mysql.com/downloads/mysql/  安装          windows:              点点点          Linux:              yum install mysql-server          Mac:              点点点 |

 服务端启动

|  |  |
| --- | --- |
| 1 | mysql.server start |

 客户端连接

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | 连接：      mysql -h host -u user -p        常见错误：          ERROR 2002 (HY000): Can't connect to local MySQL server through socket '/tmp/mysql.sock' (2), it means that the MySQL server daemon (Unix) or service (Windows) is not running.  退出：      QUIT 或者 Control+D |

**三、数据库操作**

**1、显示数据库**

|  |  |
| --- | --- |
| 1 | show databases; |

 默认数据库：  
　　mysql - 用户权限相关数据  
　　test - 用于用户测试数据  
　　information\_schema - MySQL本身架构相关数据

**2、创建数据库**

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | #utf-8  create database db1(数据库名) default charset utf8 collate utf8\_general\_ci;      #gbk  create database db2 default character set gbk collate gbk\_chinese\_ci; |

**3、使用数据库**

|  |  |
| --- | --- |
| 1 | use db\_name; |

　　显示当前使用的数据库中所有表：show tables；

**4、用户管理**

* **创建用户**

|  |  |
| --- | --- |
| 1 | create user "用户名"@“IP地址” identified by "密码"; |

* **删除用户**

|  |  |
| --- | --- |
| 1 | drop user "用户名"@"IP地址"; |

* **修改用户**

|  |  |
| --- | --- |
| 1 | rename user “用户名”@“IP地址” to “新用户名”@“IP地址”; |

* **修改密码**

|  |  |
| --- | --- |
| 1 | set password for "用户名"@“Ip地址” = Password（“新密码”）; |

**5、授权管理**

|  |  |
| --- | --- |
| 1  2  3 | show grants for “用户”@“IP地址”     ---查看权限  grant 权限 on 数据库.表 to ‘用户’@‘IP地址’     ---授权  revoke 权限 on 数据库.表 from ‘用户’@‘IP地址’    ---取消权限 |

https://images.cnblogs.com/OutliningIndicators/ContractedBlock.gif权限

https://images.cnblogs.com/OutliningIndicators/ContractedBlock.gif数据库

https://images.cnblogs.com/OutliningIndicators/ContractedBlock.gif用户和Ip

https://images.cnblogs.com/OutliningIndicators/ContractedBlock.gif实例

**四、数据表基本**

**1、创建表**

|  |  |
| --- | --- |
| 1  2  3  4 | create table 表名（      列名  类型  是否可以为空，      列名  类型  是否可以为空  ）engine = innodb default charset=utf-8 |

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

是否可空，null表示空，非字符串

not null - 不可空

null - 可空

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

默认值，创建列时可以指定默认值，当插入数据时如果未主动设置，则自动添加默认值

create table tb1(

nid int not null defalut 2,

num int not null

)

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

自增，如果为某列设置自增列，插入数据时无需设置此列，默认将自增（表中只能有一个自增列）

create table tb1(

nid int not null auto\_increment primary key,

num int null

)

或

create table tb1(

nid int not null auto\_increment,

num int null,

index(nid)

)

注意：1、对于自增列，必须是索引（含主键）。

2、对于自增可以设置步长和起始值

show session variables like 'auto\_inc%';

set session auto\_increment\_increment=2;

set session auto\_increment\_offset=10;

shwo global variables like 'auto\_inc%';

set global auto\_increment\_increment=2;

set global auto\_increment\_offset=10;

[复制代码](javascript:void(0);)

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

主键，一种特殊的唯一索引，不允许有空值，如果主键使用单个列，则它的值必须唯一，如果是多列，则其组合必须唯一。

create table tb1(

nid int not null auto\_increment primary key,

num int null

)

或

create table tb1(

nid int not null,

num int not null,

primary key(nid,num)

)

[复制代码](javascript:void(0);)

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

外键，一个特殊的索引，只能是指定内容

creat table color(

nid int not null primary key,

name char(16) not null

)

create table fruit(

nid int not null primary key,

smt char(32) null ,

color\_id int not null,

constraint fk\_cc foreign key (color\_id) references color(nid)

)

[复制代码](javascript:void(0);)

**2、删除表**

|  |  |
| --- | --- |
| 1 | drop table 表名 |

**3、清空表**

|  |  |
| --- | --- |
| 1  2 | delete from 表名  truncate table 表名（自增列回到原点） |

**4、修改表**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33 | 添加列：alter table 表名 add 列名 类型  alter table db1 add name varchar(32) not null;    删除列：alter table 表名 drop column 列名  alter table db1 drop column name;    修改列：          alter table 表名 modify column 列名 类型;  --修改 类型          alter table db1 modify column num char(10);            alter table 表名 change 原列名 新列名 类型;  --修改 列名，类型          alter table db1 change num name varchar(32);    添加主键：alter table 表名 add primary key(列名);               alter table hobby\_id add primary key(hobbies);    删除主键：          alter table 表名 drop primary key;          ALTER TABLE hobby\_id DROP PRIMARY KEY;            alter table 表名  modify  列名 int, drop primary key;          alter table hobby\_id modify hobbies char(20),drop primary key;    添加外键：alter table 从表 add constraint 外键名称（形如：FK\_从表\_主表） foreign key (从表-外键字段) references 主表(主键字段);    alter table hobby add constraint fk\_hobby\_id foreign key (hobb\_id) references hobby\_id(idc);      删除外键：alter table 表名 drop foreign key 外键名称  alter table hobby drop foreign key fk\_hobby\_id;    修改默认值：ALTER TABLE testalter\_tbl ALTER i SET DEFAULT 1000;  删除默认值：ALTER TABLE testalter\_tbl ALTER i DROP DEFAULT; |

**5、基本数据类型**

　　Mysql的数据类型大致分为：数值、时间、字符串

* char 字符串定长：易于查询，处理速度快；不省硬盘
* varchar  字符串不定长：使用灵活，省硬盘；处理速度变慢

https://images.cnblogs.com/OutliningIndicators/ContractedBlock.gif详细

**五、表内容操作★★★**

**1、增**

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | insert into 表 (列名,列名...) values (值,值,值...)  insert into 表 (列名,列名...) values (值,值,值...),(值,值,值...)  insert into 表 (列名,列名...) select (列名,列名...) from 表    insert into hobby\_id (idc,hobbies) values(4,'高尔夫');  insert into hobby\_id (idc,hobbies) values(5,"台球"),(6,'排球'); |

**2、删**

|  |  |
| --- | --- |
| 1  2  3  4 | delete from 表  delete from 表 where id＝1 and name＝'alex'    delete from hobby where name='alex' and num = 9; |

**3、改**

|  |  |
| --- | --- |
| 1  2  3 | update 表 set name ＝ 'alex' where id>1    update hobby set name='alex' where num>10; |

**4、查**

|  |  |
| --- | --- |
| 1  2  3 | select \* from 表  select \* from 表 where id > 1  select nid,name,gender as gg from 表 where id > 1 |

**5、其他**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68 | a、条件      select \* from 表 where id > 1 and name != 'alex' and num = 12;        select \* from 表 where id between 5 and 16;        select \* from 表 where id in (11,22,33)      select \* from 表 where id not in (11,22,33)      select \* from 表 where id in (select nid from 表)    b、通配符      select \* from 表 where name like 'ale%'  - ale开头的所有（多个字符串）      select \* from 表 where name like 'ale\_'  - ale开头的所有（一个字符）    c、限制      select \* from 表 limit 5;            - 前5行      select \* from 表 limit 4,5;          - 从第4行开始的5行      select \* from 表 limit 5 offset 4    - 从第4行开始的5行    d、排序      select \* from 表 order by 列 asc              - 根据 “列” 从小到大排列      select \* from 表 order by 列 desc             - 根据 “列” 从大到小排列      select \* from 表 order by 列1 desc,列2 asc    - 根据 “列1” 从大到小排列，如果相同则按列2从小到大排序    e、分组      select num from 表 group by num      select num,nid from 表 group by num,nid      select num,nid from 表  where nid > 10 group by num,nid order nid desc      select num,nid,count(\*),sum(score),max(score),min(score) from 表 group by num,nid        select num from 表 group by num having max(id) > 10        特别的：group by 必须在where之后，order by之前    f、连表      无对应关系则不显示      select A.num, A.name, B.name      from A,B      Where A.nid = B.nid        无对应关系则不显示      select A.num, A.name, B.name      from A inner join B      on A.nid = B.nid        A表所有显示，如果B中无对应关系，则值为null      select A.num, A.name, B.name      from A left join B      on A.nid = B.nid        B表所有显示，如果B中无对应关系，则值为null      select A.num, A.name, B.name      from A right join B      on A.nid = B.nid    g、组合      组合，自动处理重合      select nickname      from A      union      select name      from B        组合，不处理重合      select nickname      from A      union all      select name      from B |

# [MySQL进阶](http://www.cnblogs.com/xinsiwei18/p/5853398.html)

# 一、索引

## 1、索引概述

* 索引是表的索引目录，在查找内容之前先到目录中查找索引位置，从而快速定位查询数据；可以理解为新华字典中的索引；
* 索引会保存在额外的文件中

## 2、索引的种类

* 普通索引：仅加速查询
* 唯一索引：加速查询，列值唯一（可以有null）
* 主键索引：加速查询，列值唯一，表中只有一个，不可以有null
* 组合索引：多列值组成一个索引，对于组合搜索，组合索引的效率要大于索引合并
* 全文索引：对文本的内容进行分词，进行搜索

　　索引合并：使用多个单列索引组合查询搜索

　　覆盖索引：select的数据列只用从索引中就能够取到，不必读取数据行，换句话说查询列要被所建的索引覆盖

### 普通索引

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | #创建表+索引  CREATE TABLE person (      nid INT NOT NULL auto\_increment PRIMARY KEY,      NAME varchar (32) NOT NULL,      email VARCHAR (64) NOT NULL,      extra text,      INDEX ix\_name (NAME)  ); |
| 1  2  3  4 | #创建索引  create index index\_name on table\_name(column\_name)    create index inx\_name on person(email); |

|  |  |
| --- | --- |
| 1  2 | 删除索引  drop index\_name on table\_name; |
| 1  2 | 查看索引  show index from table\_name; |

 注意：对于创建索引时如果是BLOB 和 TEXT 类型，必须指定length。

### 唯一索引

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | #创建表+唯一索引    CREATE TABLE in1 (      nid INT NOT NULL auto\_increment PRIMARY KEY,      NAME VARCHAR (32) NOT NULL,      email VARCHAR (64) NOT NULL,      extra text,      UNIQUE ix\_name (NAME)  ) |
| 1  2 | # 创建唯一索引  create unique index 索引名 on 表名(列名) |

|  |  |
| --- | --- |
| 1  2 | # 删除唯一索引  drop unique index 索引名 on 表名 |

### 主键索引

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | # 创建表 + 创建主键  create table in1(      nid int not null auto\_increment primary key,      name varchar(32) not null,      email varchar(64) not null,      extra text,      index ix\_name (name)  )    OR    create table in1(      nid int not null auto\_increment,      name varchar(32) not null,      email varchar(64) not null,      extra text,      primary key(ni1),      index ix\_name (name)  ) | |
| 1  2 | # 创建主键  alter table 表名 add primary key(列名); |

|  |  |
| --- | --- |
| 1  2  3 | # 删除主键  alter table 表名 drop primary key;  alter table 表名  modify  列名 int, drop primary key; |

### 组合索引

　　组合索引是多个列组合成一个索引来查询

　　应用场景：频繁使用多列来进行查询，如：where name=‘alex’ and age=22

|  |  |
| --- | --- |
| 1  2 | 创建组合索引  create index ix\_name\_age on tbb（name，age）; |

如上创建组合索引之后，查询一定要注意：

* name and email  -- >使用索引，name一定要放前面
* name                 -- >使用索引
* email                 -- >不使用索引

注意：同时搜索多个条件时，组合索引的性能效率好过于多个单一索引合并。

### 其他

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

# 查看索引

show index from 表名

# 查看执行时间

set profiling = 1; # 开启profiling

SQL... # 执行SQL语句

show profiles; # 查看结果

[复制代码](javascript:void(0);)

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

# like '%xx'，避免%\_写在开头

select \* from tb1 where name like '%n';

# 使用函数

select \* from tb1 where reverse(name) = 'nick';

# or

select \* from tb1 where nid = 1 or email = '630571017@qq.com';

注：当or条件中有未建立索引的列才失效，否则会走索引

# 类型不一致

如果列是字符串类型，传入条件是必须用引号引起来。

select \* from tb1 where name = 999;

# !=，不等于

select \* from tb1 where name != 'nick'

注：如果是主键，则还是会走索引

select \* from tb1 where nid != 123

# >，大于

select \* from tb1 where name > 'nick'

注：如果是主键或索引是整数类型，则还是会走索引

select \* from tb1 where nid > 123

select \* from tb1 where num > 123

# order by

select email from tb1 order by name desc;

当根据索引排序时候，选择的映射如果不是索引，则不走索引

注：如果对主键排序，则还是走索引：

select \* from tb1 order by nid desc;

# 组合索引最左前缀

如果组合索引为：(name,email)，查询使用：

name and email -- 使用索引

name -- 使用索引

email -- 不使用索引

[复制代码](javascript:void(0);)

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

# 避免使用select \*

# count(1)或count(列) 代替 count(\*)

# 创建表时尽量时 char 代替 varchar

# 表的字段顺序固定长度的字段优先

# 组合索引代替多个单列索引（经常使用多个条件查询时）

# 尽量使用短索引

# 使用连接（JOIN）来代替子查询(Sub-Queries)

# 连表时注意条件类型需一致

# 索引散列值（重复少）不适合建索引，例：性别不适合

[复制代码](javascript:void(0);)

* explain 查询sql语句-----用于显示sql执行信息参数，根据参考信息可以进行相应的优化

# 二、视图

* 虚拟表（非真实存在）
* 本质：根据SQL语句获取动态的数据集，并为其命名，用户使用时只需使用名称获取结果集，并可以将其当做表来使用

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

SELECT

\*

FROM

(

SELECT

nid,

NAME

FROM

person

WHERE

nid > 2

) AS A

WHERE

A. NAME = 'seven';

[复制代码](javascript:void(0);)

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

--格式：CREATE VIEW 视图名称 AS SQL语句

CREATE VIEW v1 AS SELECT

nid,

NAME,

email

FROM

person

WHERE

nid > 2;

SELECT

\*

FROM

v1;

[复制代码](javascript:void(0);)

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

--格式：DROP VIEW 视图名称

DROP VIEW v1

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

-- 格式：ALTER VIEW 视图名称 AS SQL语句

ALTER VIEW v1 AS

SELET A.nid,

B. NAME

FROM

A

LEFT JOIN B ON A.id = B.nid

LEFT JOIN C ON A.id = C.nid

WHERE

A.id > 2

AND C.nid < 5

[复制代码](javascript:void(0);)

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

select \* from v1

　　使用视图时，将其当做表进行操作即可，由于视图时虚拟表，所以无法使用其对真实表进行创建、更新和删除操作，仅能做查询用。