# MySQL基础操作

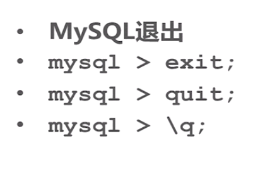


## MySQL登录

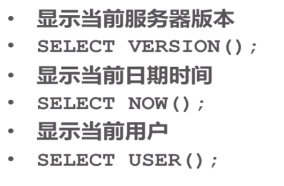
C:\Users\ai>mysql -uroot -p -P3306 -h127.0.0.1

Enter password: \*\*\*\*\*\*\*\*\*

MySQL退出：



## MySQL常用命令

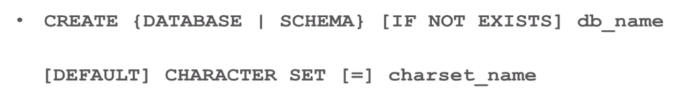


关键字与函数名称全部大写

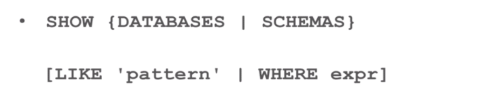
数据库名称、表名称、字段名称全部小写

SQL语句必须以分号结束“;”

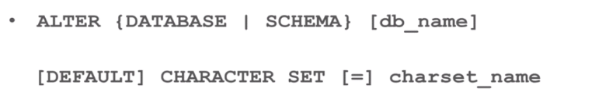
## 创建数据库



## 查看当前服务器下的所有数据库



## 修改数据库



## 删除数据库



# 数据类型

## 整型

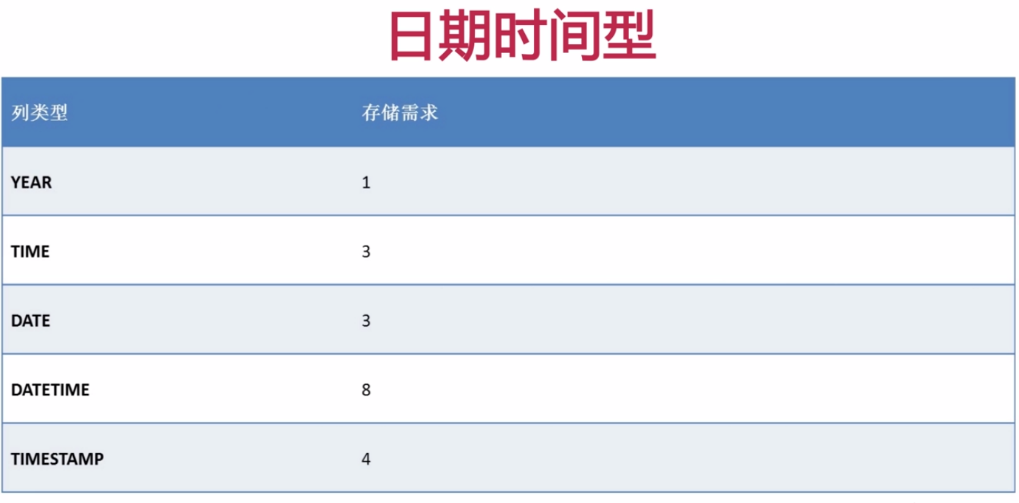


数据优化时选择最合理最合适的数据类型存储。

## 浮点型



## 日期时间型



## 字符型



# 数据表的操作

## 创建数据库 create

CREATE DATABASE test;

## 查看已存在的数据库 show

SHOW DATABASES;

## 打开数据库 use

USE test;

## 查看当前已打开的数据库

SELECT DATABASE();

## 创建表 create

mysql> CREATE TABLE tb1(

-> username VARCHAR(20),

-> age TINYINT UNSIGNED,

-> salary FLOAT(8,2) UNSIGNED

-> );

Query OK, 0 rows affected (0.43 sec)

## 查看（指定）数据库下的所有表 show

查看当前已打开数据库下的表：mysql> SHOW TABLES;

查看指定数据库下的表：mysql> SHOW TABLES FROM mysql;

## 查看数据表结构 show

mysql> SHOW COLUMNS FROM tb1;

mysql> desc tb\_samples;

…………..

3 rows in set (0.08 sec)

查看表创建命令

## 向数据表中插入记录 insert

省略列名时默认是（必须）给所有列都插入值

mysql> INSERT tb1 VALUES('Tom',25,7894.25);

Query OK, 1 row affected (0.07 sec)

指定要插入到哪些列中

mysql> INSERT tb1 (username,salary) VALUES('John',4500.23);

Query OK, 1 row affected (0.11 sec)

## 空值与非空 not null

NULL：字段值可以为空

NOT NULL：字段值不能为空

## 主键

PRIMARY KEY

主键约束

每张数据表只能存在一个主键

主键保存记录的唯一性

主键自动为NOT NULL

## 自动编号

AUTO\_INCREMENT

自动编号必须为**数值型**，若为浮点数小数后应为0

自动编号应与主键组合使用（但主键不一定有此属性）

默认情况下起始值为1，每次增量为1

mysql> CREATE TABLE tb3(

-> id SMALLINT UNSIGNED AUTO\_INCREMENT PRIMARY KEY,

-> username VARCHAR(30) NOT NULL

-> );

Query OK, 0 rows affected (0.22 sec)

插入姓名时主键id自动从1开始递增。当然主键也允许赋值但不能出现相同的值。

mysql> SELECT \* FROM tb3;

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

| id | username |

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

| 1 | oen |

| 2 | TWO |

| 3 | three |

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

3 rows in set (0.00 sec)

## 唯一约束

UNIQUE KEY

唯一约束

唯一约束可以保证记录的唯一性（类似主键）

唯一字段的值可以为空（NULL）--多个记录值可以同时为空

每张数据表可以存在多个唯一约束

mysql> CREATE TABLE tb4(

-> id SMALLINT AUTO\_INCREMENT PRIMARY KEY,

-> username VARCHAR(20) NOT NULL UNIQUE KEY,

-> age TINYINT UNSIGNED

-> );

Query OK, 0 rows affected (0.25 sec)

## 默认约束

DEFAULT

当插入记录时，如果没有明确为字段赋值，则自动赋予默认值。

mysql> CREATE TABLE tb5(

-> id SMALLINT UNSIGNED AUTO\_INCREMENT PRIMARY KEY,

-> sex ENUM('1','2','3') DEFAULT '3'

-> );

Query OK, 0 rows affected (0.25 sec)

# 约束及修改数据表

## 约束

约束保证数据的完整性和一致性

约束分为表级约束（一个字段以上）和列级约束（仅一个字段）

约束类型包括：

非空约束：NOT NULL

主键约束：PRIMARY KEY

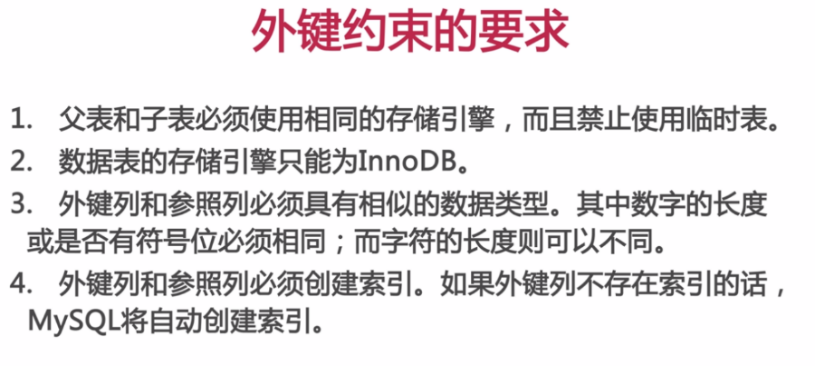
唯一约束：UNIQUE KEY

默认约束：DEFAULT

外键约束：FOREIGN KEY

## 外键约束 foreign

实现一对一或一对多关系



编辑数据表的默认存储引擎

配置文件：My.ini

编辑：default-storage-engine=INNODB

创建

mysql> CREATE TABLE provinces(

-> id SMALLINT UNSIGNED PRIMARY KEY AUTO\_INCREMENT,

-> pname VARCHAR(20) NOT NULL

-> );

Query OK, 0 rows affected (2.21 sec)

mysql> CREATE TABLE users(

-> username VARCHAR(10) NOT NULL,

-> pid SMALLINT UNSIGNED,

-> FOREIGN KEY (pid)REFERENCES provinces (id)

-> );

Query OK, 0 rows affected (0.34 sec)

pid对应的表users为子表，子表参照的表provinces为父表。

pid和id类型为整形所以必须相同

## 查看索引 show index

mysql> SHOW INDEXES FROM users\G;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Table: users

Non\_unique: 1

Key\_name: pid

Seq\_in\_index: 1

Column\_name: pid

Collation: A

Cardinality: 0

Sub\_part: NULL

Packed: NULL

Null: YES

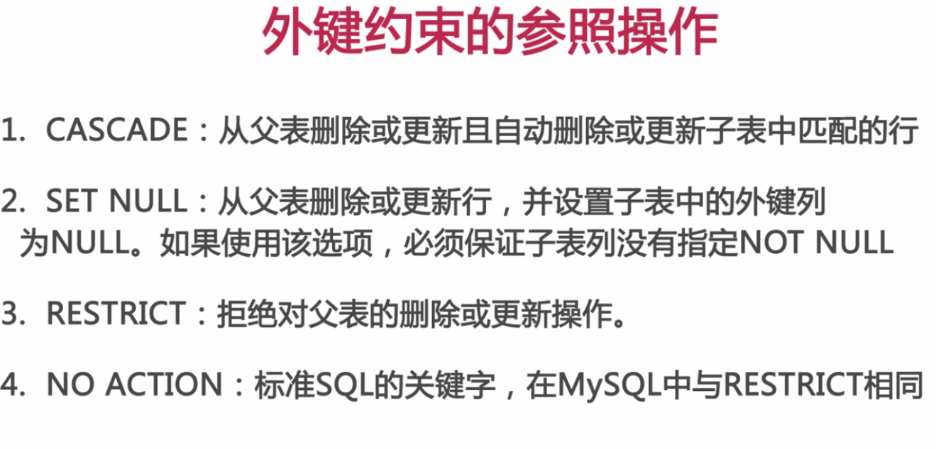
Index\_type: BTREE

Comment:

Index\_comment:

1 row in set (0.00 sec)

## 外键约束的参照条件



mysql> CREATE TABLE users1(

-> id SMALLINT UNSIGNED PRIMARY KEY AUTO\_INCREMENT,

-> username VARCHAR(20) NOT NULL,

-> pid SMALLINT UNSIGNED,

-> FOREIGN KEY (pid) REFERENCES provinces (id) **ON DELETE CASCADE**

-> );

Query OK, 0 rows affected (0.29 sec)

查看创建命令

mysql> SHOW CREATE TABLE users1;

在两张表中插入信息验证外键约束的查找条件CASCADE

1. 在两张表中插入信息（必须先在父表中插入）
2. 在父表中删除值，同时子表中对应的值也会被删除

## 表级约束和列级约束

对一个数据**列**建立的约束称为列级约束

对**多个**数据列建立的约束称为表级约束

列级约束既可以在列定义时声明，也可以在列定义后声明，表级约束只能在列定义后声明。

## 添加列 alter add

1. 添加一列：

ALTER TABLE tb\_name **ADD** [CLOUMN] col\_name column\_definition [FIRST|AFTER col\_name]

mysql> ALTER TABLE users1 **ADD** age TINYINT UNSIGNED NOT NULL DEFAULT 10 **AFTER** username;

若不加FIRST或AFTER…默认添加到尾部

添加多列：

1. ALTER TABLE tb\_name ADD [COLUMN] （col\_name column\_definition,………）

**不能指定位置关系**，只能添加到数据表末尾。

## 删除列 alter drop

ALTER TABLE tb\_name DROP [COLUMN] col\_name

mysql> ALTER TABLE users1 DROP username;

删除多列只需在DROP后添加多个列名用逗号分隔

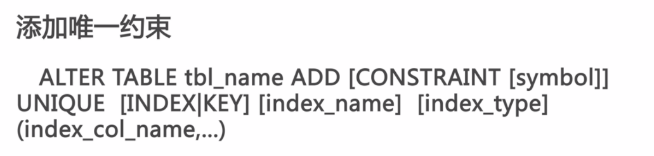
## 添加主键约束 alter add constraint

ALTER TABLE tb\_name ADD [CONSTRAINT [symbol]] PRIMARY EKY [index\_type] (index\_col\_name,…)

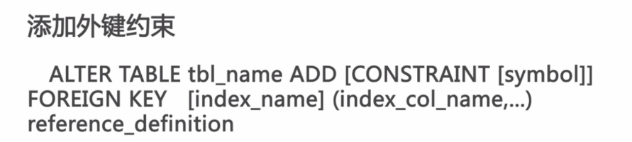
mysql> ALTER TABLE users2 ADD CONSTRAINT PK\_users2\_id PRIMARY KEY (id);

PK\_users2\_id为主键的名字

## 添加唯一约束 alter add unique



## 添加外键约束 alter add foreign



## [添加联合唯一约束](http://blog.csdn.net/yumushui/article/details/38960619)

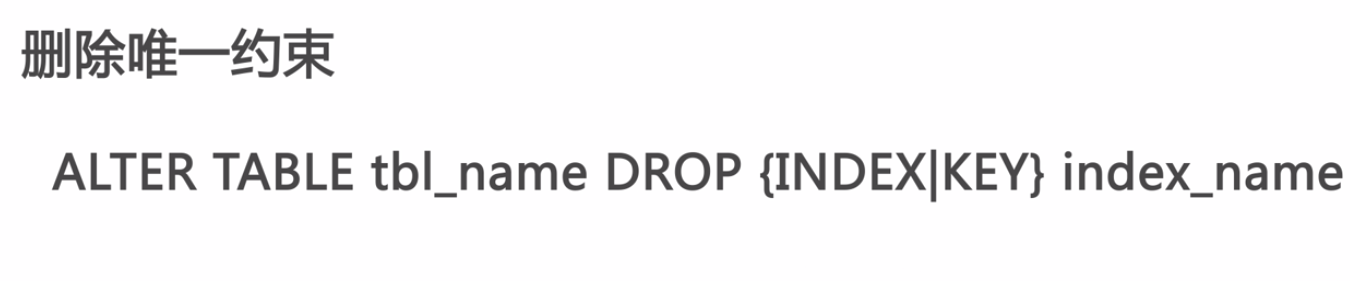
mysql> ALTER TABLE jw\_resource ADD **UNIQUE KEY(**resource\_name, resource\_type**)**;

## 删除主键约束 alter drop primary



## 删除唯一约束 alter drop

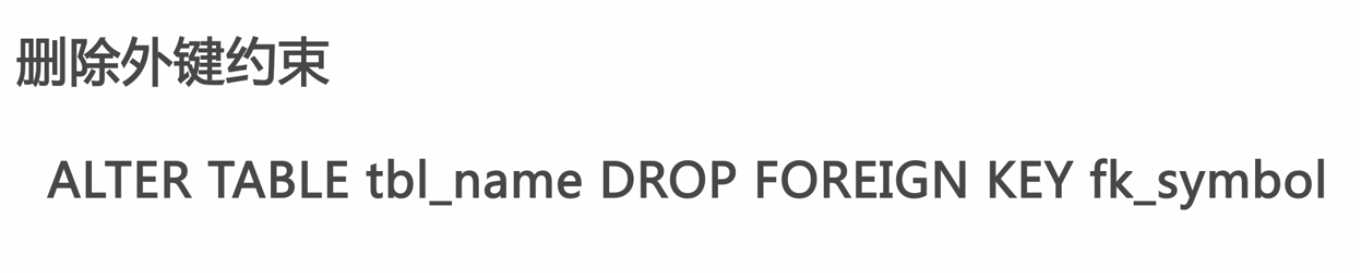
通过SHOW INDEX FROM tb\_name;查看索引（Key\_name对应项）。



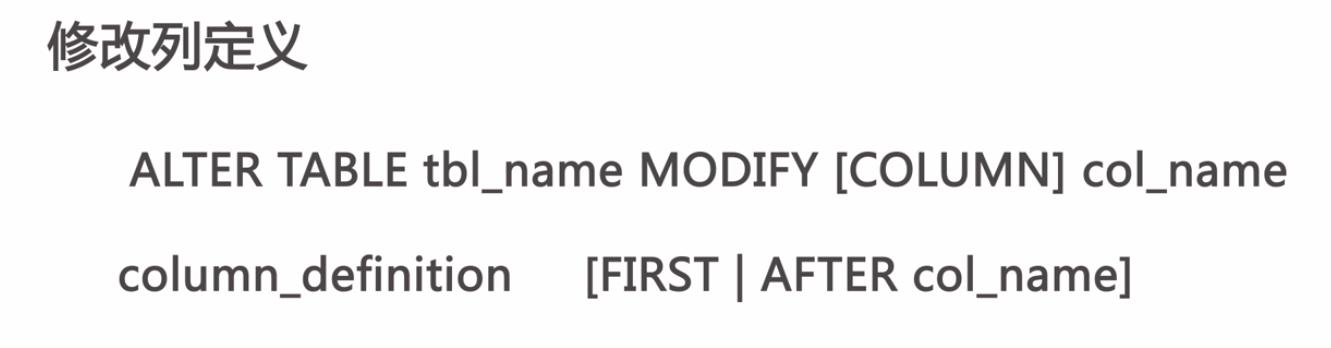
删除的是字段上的约束，而并非字段。

## 删除外键约束 alter drop foreign

通过SHOW CREATE TABLE tab\_name;查看外键约束。

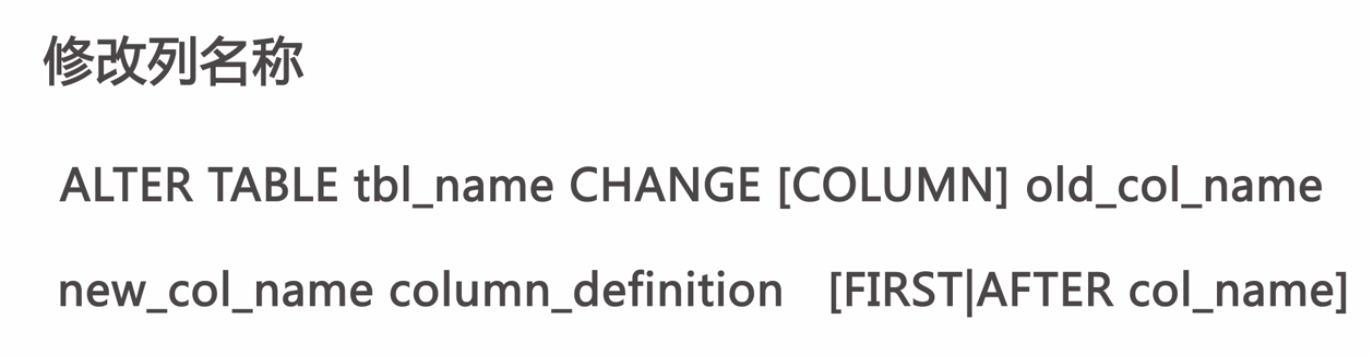


## 修改列定义 alter modify

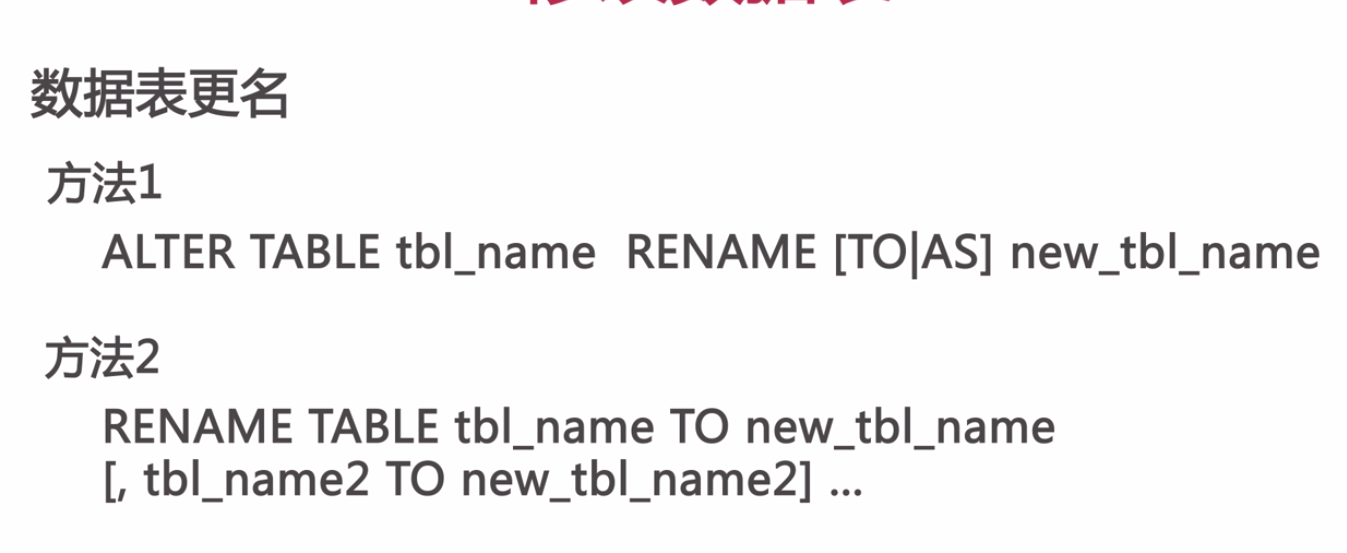


ALTER TABLE tb5 MODIFY sex ENUM('1','2','3','4','5') DEFAULT 5 FIRST;

## 修改列名及定义 alter change



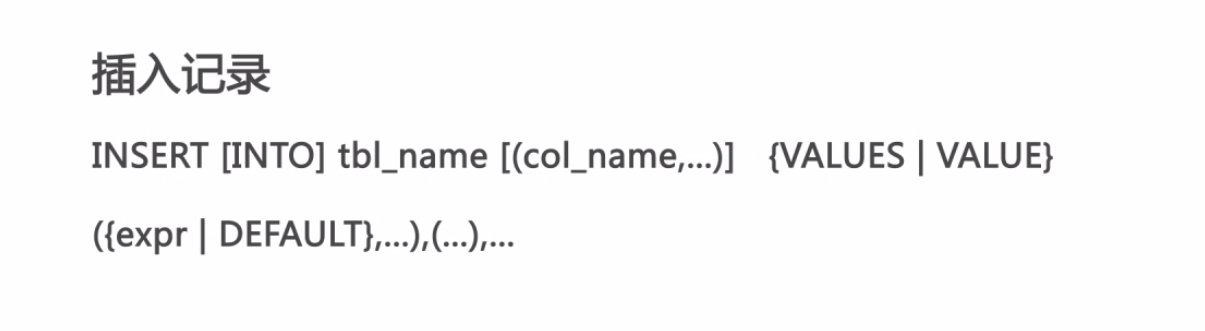
## 数据表重命名 alter rename



## 去重：distinct

# 操作数据

## 插入——INSERT



mysql> CREATE TABLE five01(

-> id SMALLINT UNSIGNED PRIMARY KEY AUTO\_INCREMENT,

-> username VARCHAR(20) NOT NULL,

-> password VARCHAR(32) NOT NULL,

-> age TINYINT UNSIGNED NOT NULL DEFAULT 13,

-> sex BOOLEAN

-> );

Query OK, 0 rows affected (0.29 sec)

### 单值，多值插入

一次插入一个值

mysql> INSERT fivee01 VALUES(***NULL***,'Tom','123',23,1);

mysql> INSERT fivee01 VALUES(***DEFAULT***,'Tom','123',23,1);

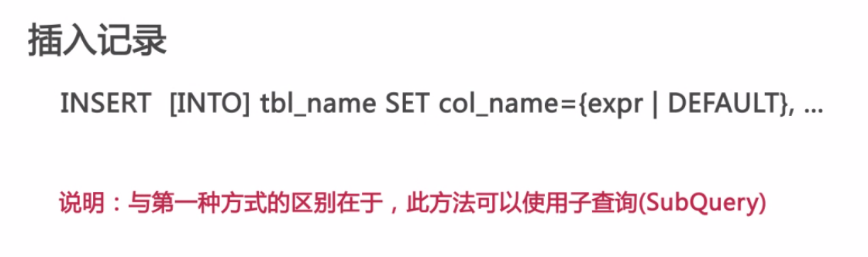
DEFAULT和NULL对应自增的主键id，不为其赋值而使之按默认的自增可赋予这两个值。

一次插入多个值

mysql> INSERT five01 VALUES(***NULL***,'Tom','123',23,1),(***DEFAULT,***'Join',***md5(123)***,***DEFAULT***,0);

md5()：为PHP函数，将传入参数转换为哈希值。

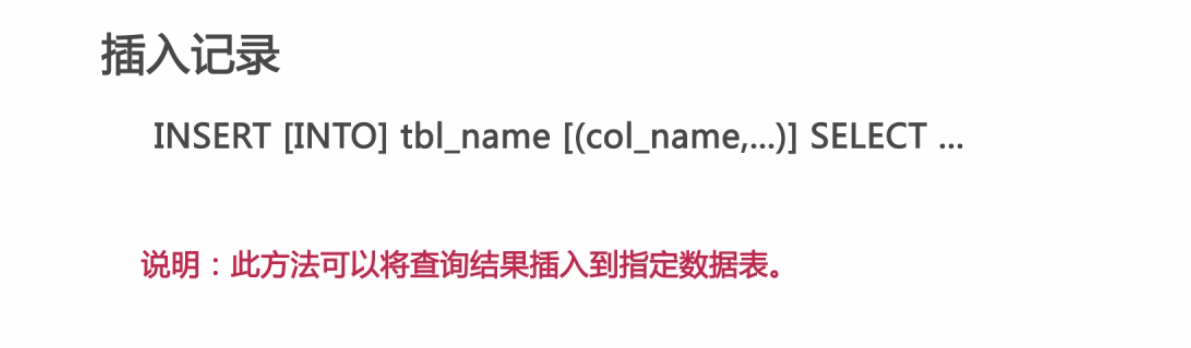
### set可使用子查询



一次只能插入一个记录

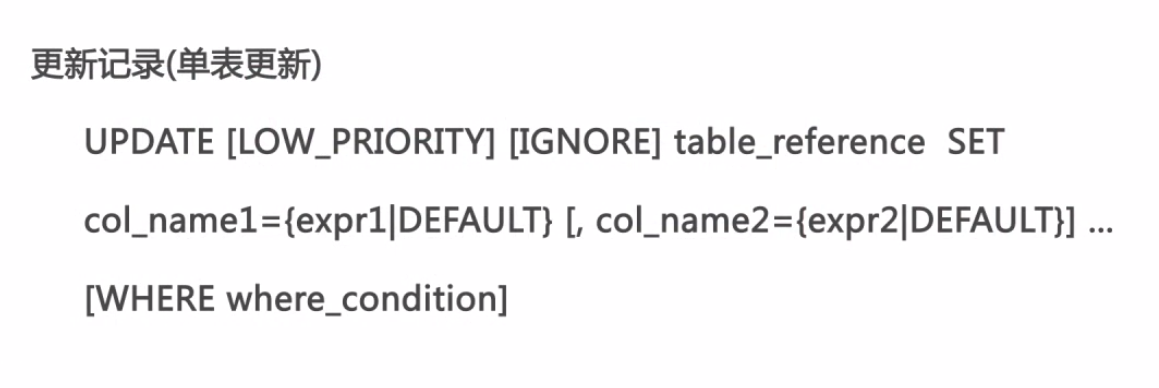
mysql> INSERT five01 **SET** username='Bean',password='4567';

### 插入查询结果



mysql> INSERT tb4\_a(username,age) SELECT username,age FROM tb4 WHERE age>=25;

## 更新记录——UPDATE



### **只更新一列**

mysql> UPDATE five01 SET age=age+3;

此命令将five01表中所有记录的age值更改为age+3；

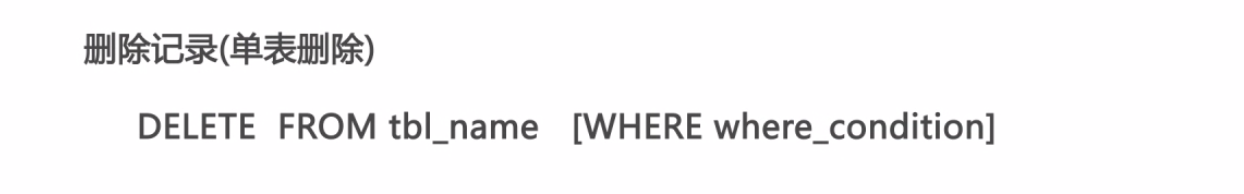
### **更新多列**

mysql> UPDATE five01 SET age=age+2,sex=0;

### **更新符合指定条件的列**

mysql> UPDATE five01 SET age=age+10 WHERE id%2=0;

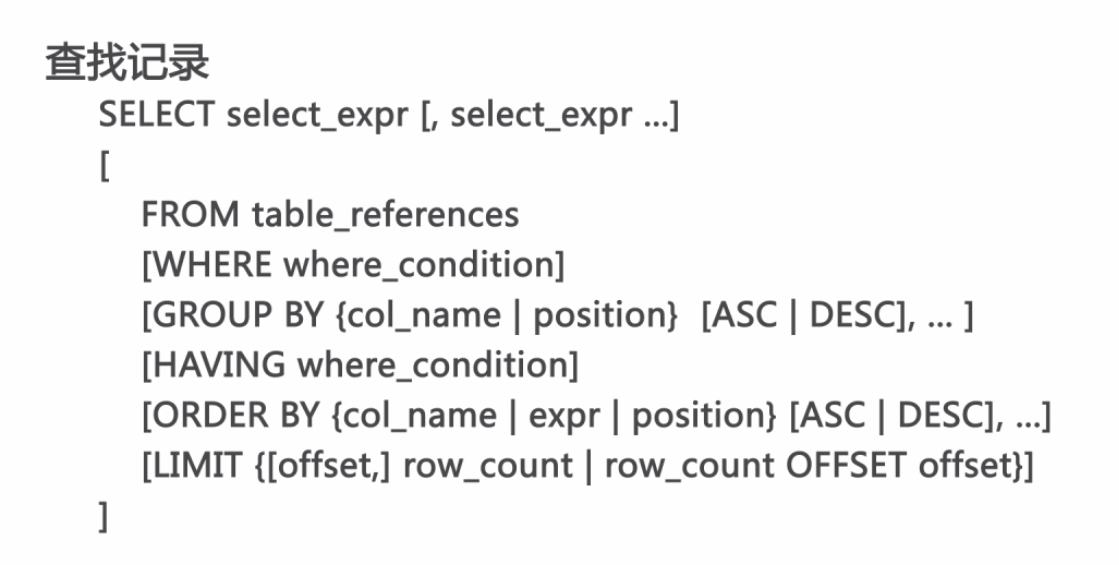
## 删除记录——DELETE



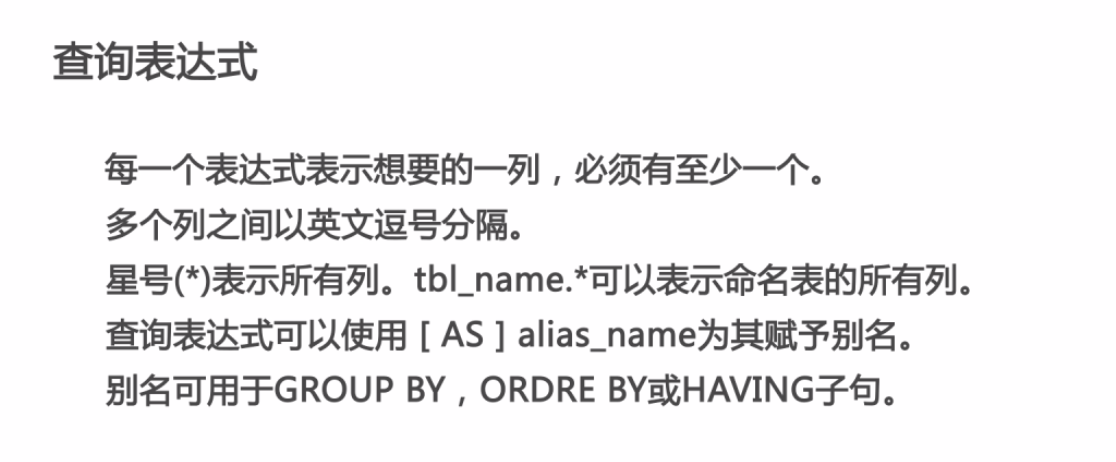
mysql> DELETE FROM five01 WHERE id=2;

## 清空表——TRUNCATE

## 查找记录——SELECT



### 查询表达式



mysql> SELECT id AS uid,username AS uname FROM five01;

mysql> SELECT 78-34+34\*122;

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

| 78-34+34\*122 |

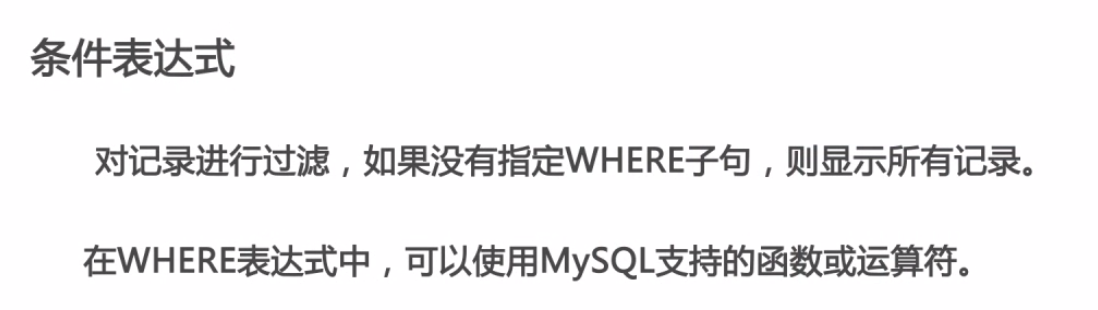
+-----------------------+

| 4192 |

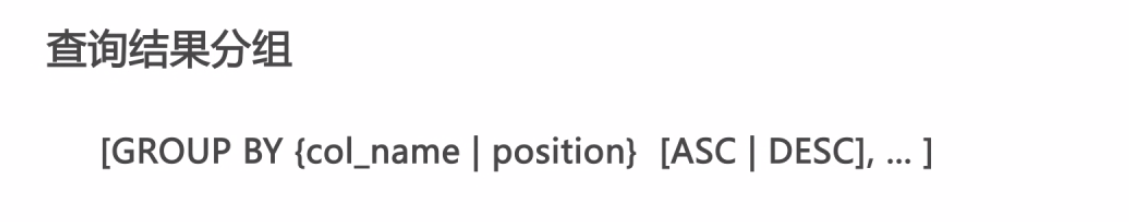
+------------------------+

1 row in set (0.04 sec)

### 查询条件

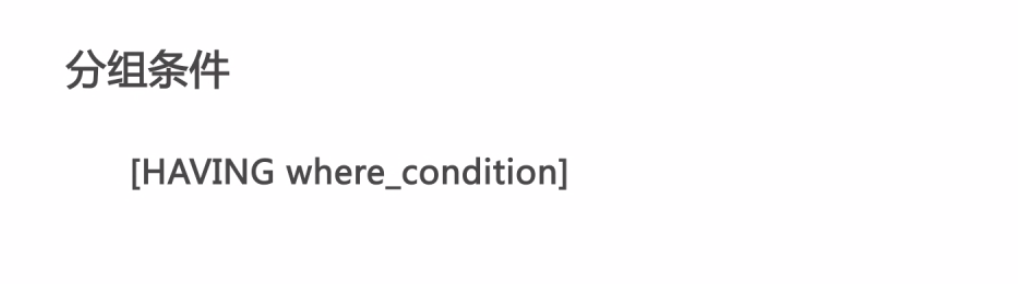


### 查询结果分组——GROUP BY

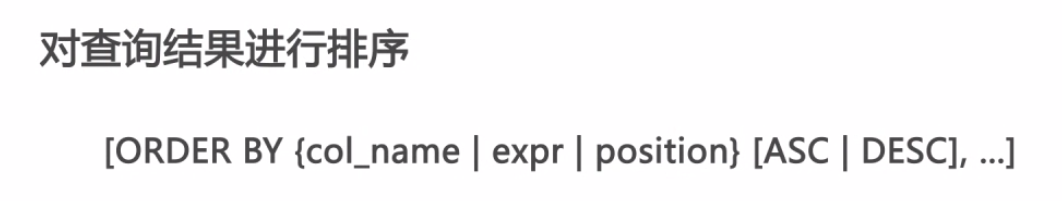


mysql> SELECT sex FROM five01 GROUP BY sex;

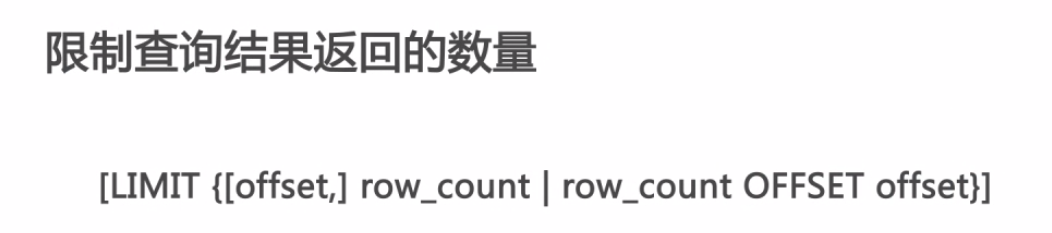
#### **分组条件——HAVING**



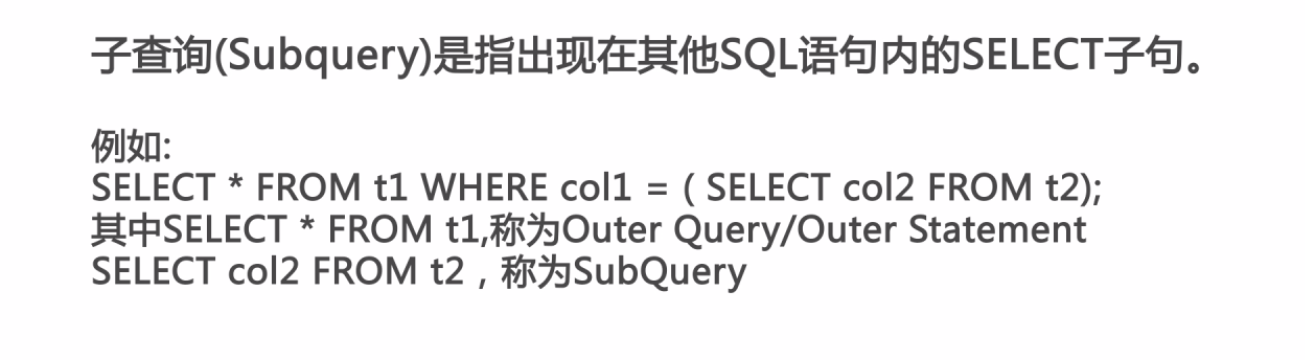
#### **对查询结果排序——ORDER BY**

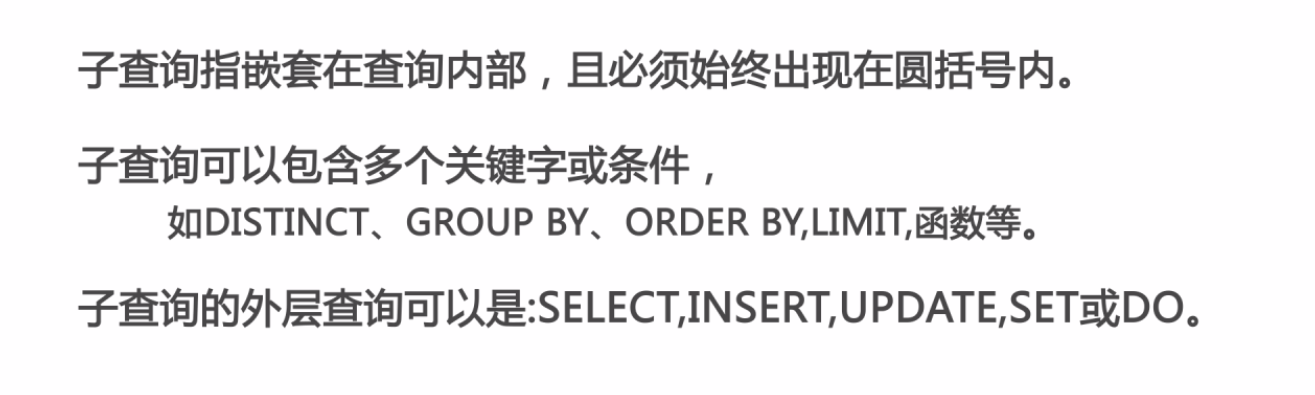


#### **限制查询结果返回数量——LIMIT**



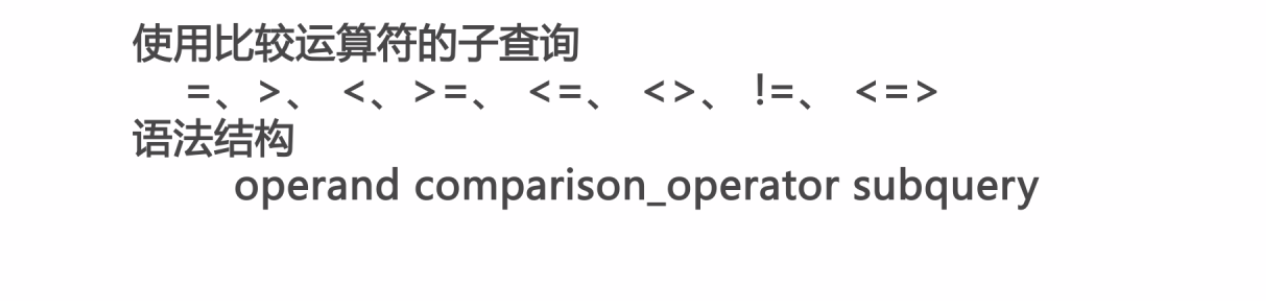
# 子查询与连接





子查询可以返回标量、一行、一列或子查询

## 使用比较运算符的子查询



mysql> SELECT ROUND(AVG(goods\_price),2) FROM tdb\_goods;

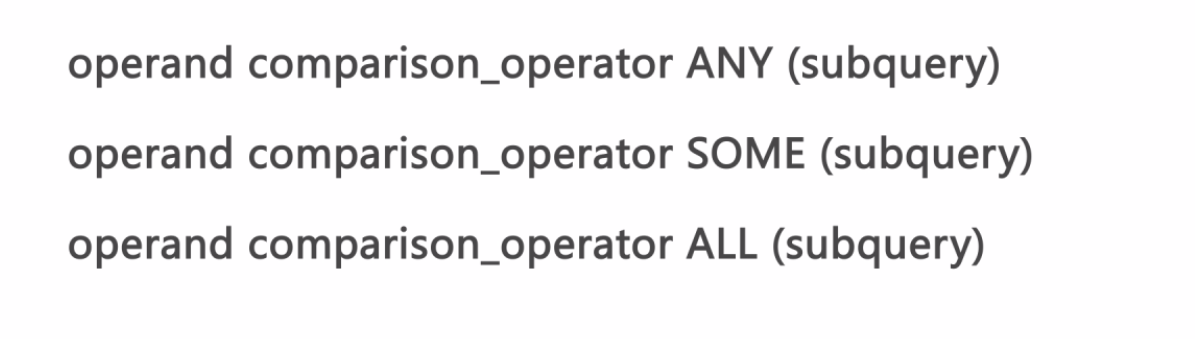
ROUND(r,2)对r进行四舍五入并保留小数点后两位

AVG(a)对a字段的所有值求平均值

mysql> SELECT goods\_id,goods\_name,goods\_price FROM tdb\_goods WHERE goods\_price>=(SELE

CT ROUND(AVG(goods\_price),2) FROM tdb\_goods);

当子查询返回多个结果时选用ANY/SOME/ALL来选出其中唯一一个。

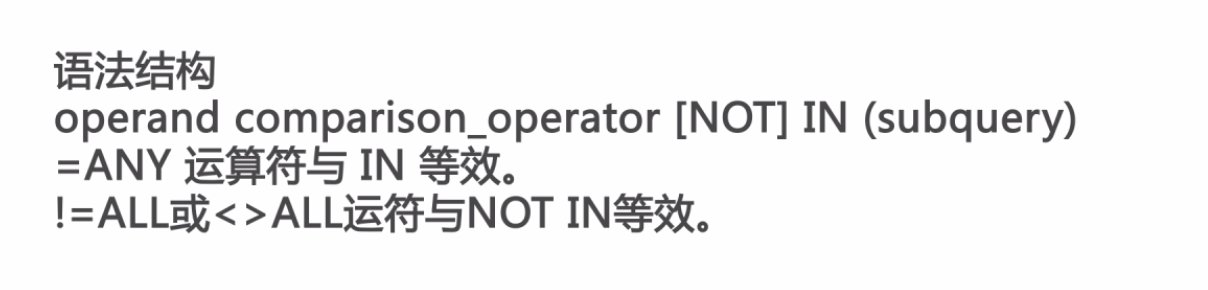




mysql> SELECT goods\_id,goods\_name,goods\_price FROM tdb\_goods WHERE goods\_price>=***ALL*** (SELECT goods\_p

rice FROM tdb\_goods WHERE goods\_cate='超级本');

## 使用[NOT]IN的子查询



# 从数据库导出数据库文件

1. 将数据库mydb导出到e:/mysql/mydb.sql文件中：  
   打开开始->运行->输入cmd 进入命令行模式  
   c:/>mysqldump -h localhost -u root -p mydb >e:/mysql/mydb.sql  
   然后输入密码，等待一会导出就成功了，可以到目标文件中检查是否成功。  
   2.将数据库mydb中的mytable导出到e:/mysql/mytable.sql文件中：  
   c:/>mysqldump -h localhost -u root -p mydb mytable>e:/mysql/mytable.sql   
   3.将数据库mydb的结构导出到e:/mysql/mydb\_stru.sql文件中：  
   c:/>mysqldump -h localhost -u root -p mydb --add-drop-table >e:/mysql/mydb\_stru.sql

# 从外部文件导入数据到数据库中

从e:/mysql/mydb2.sql中将文件中的SQL语句导入数据库中：  
1.从命令行进入mysql，然后用命令CREATE DATABASE mydb2;创建数据库mydb2。  
2.退出mysql 可以输入命令exit；或者quit；  
3.在CMD中输入下列命令：  
c:/>mysql -h localhost -u root -p mydb2 < e:/mysql/mydb2.sql  
然后输入密码，就OK了。  
五.下面谈一下关于导入文件大小限制问题的解决：  
默认情况下：mysql 对导入文件大小有限制的，最大为2M，所以当文件很大时候，直接无法导入，下面就这个问题的解决列举如下：  
1.在php.ini中修改相关参数：  
影响mysql导入文件大小的参数有三个：  
memory\_limit=128M,upload\_max\_filesize=2M,post\_max\_size=8M  
修改upload\_max\_filesize=200 M 这里修改满足你需要的大小，  
可以同时修改其他两项memory\_limit=250M post\_max\_size=200M  
这样就可以导入200M以下的.sql文件了。

# 数据库索引（MySQL）

<https://www.cnblogs.com/chenshishuo/p/5030029.html>

mysql的索引分为单列索引(主键索引,唯索引,普通索引)和组合索引.

1. **单列索引**:一个索引只包含一个列,一个表可以有多个单列索引.
2. **组合索引**:一个组合索引包含两个或两个以上的列,

## 索引的创建

### 单列索引

1. **普通索引,**这个是最基本的索引

其sql格式是：

CREATE INDEX IndexName ON `TableName`(`字段名`(length)) 或

ALTER TABLE TableName ADD INDEX IndexName(`字段名`(length))。

示例：

mysql> CREATE INDEX account\_index ON award(account);

ALTER TABLE award ADD INDEX account\_index(account);

1. **唯一索引**,与普通索引类似,但是不同的是唯一索引要求所在的列的值是唯一的,这一点和主键索引一样.但是他允许有空值。

其sql格式是：

CREATE UNIQUE INDEX IndexName ON `TableName`(`字段名`(length)); 或者

ALTER TABLE TableName ADD UNIQUE (column\_list)

示例：

CREATE UNIQUE INDEX account\_UNIQUE\_index ON award(account);

1. **主键索引**,不允许有空值,(在B+TREE中的InnoDB引擎中,主键索引起到了至关重要的地位)

主键索引建立的规则是 int优于varchar,一般在建表的时候创建,最好是与表的其他字段不相关的列或者是业务不相关的列.一般会设为 int 而且是 AUTO\_INCREMENT自增类型的。

### 组合索引

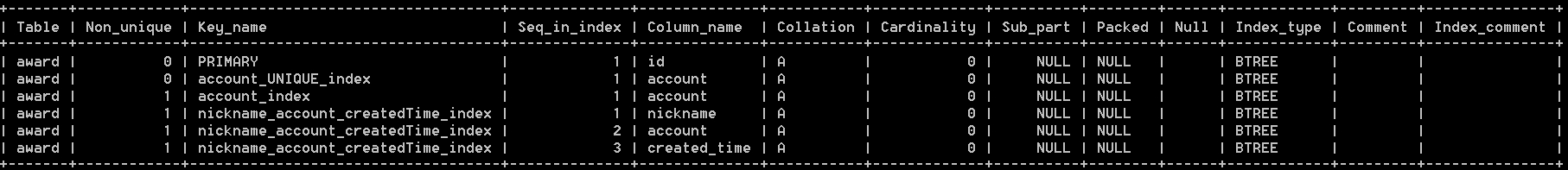
一个表中含有多个单列索引不代表是组合索引,通俗一点讲 组合索引是:包含多个字段但是只有索引名称。

其sql格式是：

CREATE INDEX IndexName On `TableName`(`字段名`(length),`字段名`(length),...);

示例：

mysql> CREATE INDEX nickname\_account\_createdTime\_index ON award(nickname,account,created\_time)



如果你建立了组合索引(nickname\_account\_createdTime\_Index) 那么他实际包含的是3个索引 (nickname) (nickname,account)(nickname,account,created\_time)

在使用查询的时候遵循mysql组合索引的"**最左前缀**"

1. **不按索引最左列开始查询**（多列索引） 例如index(‘c1’, ‘c2’, ‘c3’) where‘c2 =‘aaa’ 不使用索引,where `c2` = `aaa` and `c3`=`sss` 也不能使用索引
2. 查询中**某个列有范围查询**，则其右边的所有列都无法使用查询（多列查询）

Where c1= ‘xxx’and c2 like = ‘aa%’ and c3=’sss’

该查询只会使用索引中的前两列,因为like是范围查询

1. 不能跳过某个字段来进行查询,这样使用不到索引,比如我的sql 是：

explain select \* from `award` where nickname > 'rSUQFzpkDz3R' and account = 'DYxJoqZq2rd7' and created\_time = 1449567822;

那么这时候他使用不到其组合索引.因为我的索引是 (nickname, account, created\_time),如果第一个字段出现范围符号的查找,那么将不会用到索引,如果我是第二个或者第三个字段使用范围符号的查找,那么他会利用索引,利用的索引是(nickname)

## 索引删除

删除索引的mysql格式：

DORP INDEX IndexName ON `TableName`

## 使用索引的优点

1. 可以通过建立唯一索引或者主键索引,保证数据库表中每一行数据的唯一性.

2. 建立索引可以大大提高检索的数据,以及减少表的检索行数

3. 在表连接的连接条件可以加速表与表连接的相连

4. 在分组和排序子句进行数据检索,可以减少查询时间中分组和排序时所消耗的时间(数据库的记录会重新排序)

5. 建立索引,在查询中使用索引可以提高性能

## 使用索引的缺点

1. 在创建索引和维护索引会耗费时间,随着数据量的增加而增加

2. 索引文件会占用物理空间,除了数据表需要占用物理空间之外,每一个索引还会占用一定的物理空间

3. 当对表的数据进行 INSERT,UPDATE,DELETE 的时候,索引也要动态的维护,这样就会降低数据的维护速度,(建立索引会占用磁盘空间的索引文件。一般情况这个问题不太严重，但如果你在一个大表上创建了多种组合索引，索引文件的会膨胀很快)。

## 使用索引需要注意的地方

在建立索引的时候应该考虑索引应该建立在数据库表中的某些列上面 哪一些索引需要建立,哪一些索引是多余的.

一般来说,

1. 在经常需要搜索的列上建立索引,可以加快搜索速度
2. 主键列上可以确保列的唯一性
3. 在表与表的连接条件上加上索引,可以加快连接查询的速度
4. 在经常需要排序(order by),分组(group by)和去重（distinct）的列上加索引可以加快排序查询的时间, (单独order by 用不了索引，索引考虑加where 或加limit)
5. 在一些where 之后的<、<=、>、>=、BETWEEN IN以及某些情况下的like建立字段的索引(B-TREE)
6. like语句的如果你对nickname字段建立了一个索引.当查询的时候的语句是 nickname like '%ABC%' 那么这个索引将不会起到作用.而nickname like 'ABC%' 那么将可以用到索引
7. 索引不会包含NULL列,如果列中包含NULL值则不会被包含在索引中,复合索引中如果有一列含有NULL值那么这个组合索引都将失效,一般需要给默认值0或者 ' '字符串
8. 使用短索引,如果你的一个字段是Char(32)或者int(32),在创建索引的时候指定前缀长度 比如前10个字符 (前提是多数值是唯一的..)那么短索引可以提高查询速度,并且可以减少磁盘的空间,也可以减少I/0操作.
9. 不要在列上进行运算,这样会使得mysql索引失效,也会进行全表扫描
10. 选择越小的数据类型越好,因为通常越小的数据类型通常在磁盘,内存,cpu,缓存中占用的空间很少,处理起来更快

## 什么情况下不创建索引

1. 查询中很少使用到的列不应该创建索引,如果不合理建立索引会增大空间需求,反而降低mysql的性能.
2. 数据内容很少的列也不应该建立索引,比如一个性别字段0或者1,在查询中,结果集的数据占了表中数据行的比例比较大,mysql需要扫描的行数很多,增加索引,并不能提高效率
3. 定义为text和image和bit数据类型的列不应该增加索引,
4. 当表的修改(UPDATE,INSERT,DELETE)操作远远大于检索(SELECT)操作时不应该创建索引,这两个操作是互斥的关系

# MySQL VIEW+JOIN

以great\_web数据库为例，note表有用户留言信息和留言用户的id（uid），user表保存用户的信息（用户id），查询出每一个用户的所有留言：

create view noteView as (select title,uid from note);

create view userView as (select id,name from user);

select userView.name as userName,noteView.title as noteTitle from userView,noteView where userView.id = noteView.uid;

## 视图（VIEW）

<http://www.cnblogs.com/chenpi/p/5133648.html>

视图就是一条SELECT语句执行结果后的集合，视图是基于若干张表的引用，是一张虚表，不存储具体的数据（基本表发生改变，视图也会跟着改变），可以跟基本表一样，对视图进行CRUD操作（增删改操作有条件限制）。

减少复杂的SQL语句，增强可读性，更加安全。

创建好视图后，可以直接基于创建好的视图进行CRUD操作。

不能在一张由多张表关联表连接而成的视图上进行同时修改两张表的操作。

…

## 连接（JOIN）

<http://www.cnblogs.com/dinglinyong/p/6656315.html> （文氏图）

1. 内连接：选取共有部分 select u.name,n.title from note n inner join user u on **n.uid=u.id**;
2. 左&中（外）连接：共有部分和左边部分 select u.name,n.title from note n left join user u on **n.state=u.state**;
3. 右&中（外）连接：共有部分和右边select u.name,n.title from note n right join user u on **n.state=u.state**;
4. 左连接：去除公共部分的左边部分：select u.name,n.title from note n left join user u on **n.state=u.state** where **u.state** **is null**;
5. 右链接：去除公共部分的右边部分：select u.name,n.title from note n right join user u on **n.state=u.state** where **n.state** **is null**;
6. 全连接：全部select u.name,n.title from note n **right** join user u on **n.state=u.state** **union** select u.name,n.title from note n **left** join user u on **n.state=u.state**;
7. 两张表独有部分的集合（左连接+外连接）：select \* from note n right join user u on **n.uid=u.id** where **n.uid** is null union select \* from note n left join user u on **n.uid=u.id** where **u.id** is null;

# MySQL in between…and

## In

允许我们在where字句中规定多个值。

SELECT \* FROM teacher WHERE teacher\_id IN(1,2,3)

## between …And

选取介于两个值之间的数据范围，这些值可以是数值、文本或日期。

SELECT \* FROM teacher WHERE teacher\_id BETWEEN 3 AND 5

# SQL中=、in、like区别

三者都可以用来进行数据匹配 。但三者并不相同。

* =： 等号是用来查找与单个值匹配的所有数据；

等号确切知道所要查找的内容，且为单一值时，可以使用等号运算符来进行数据比较。等号运算符中可以使用字符串、日期或数字。

* IN： IN 是用来查找与多个值匹配的所有数据；

当确切知道所要查找的内容，且为多个值时，可以使用 IN 子句来进行数据比较。IN 子句中也可以使用数字、字符串或日期。

* LIKE： 而LIKE用来查找与一个模式匹配的所有数据。

当无法确切知道所要查找的值，而是知道所要查找的数据符合的模式时，可以使用LIKE 子句进行匹配。一般来说，LIKE 子句仅在字符串类型时使用。

通配符：

%：替代0个或多个字符

\_：仅替代一个字符

[abcd]：字符列中的任何单一字符

^(或!)：取相反

# 数据库表间的三种对应（映射）关系

一对一，一对多/多对一，多对多

数据库三大范式：<http://www.cnblogs.com/linjiqin/archive/2012/04/01/2428695.html>

1 确保每列保持原子性

2 确保表中的每列都和主键相关 （复合主键）<http://blog.csdn.net/wangyuchun_799/article/details/49329649>

3 确保每列和主键直接相关，而不是间接相关

<http://www.cnblogs.com/abeam/p/7406285.html>

* 一对一：夫妻关系，数据表中表现为外键关联，主键关联。A表中的一条记录对应B表中的一条记录。
* 一对多：存在最普遍的映射关系，如球队与球员。A表中的一条记录对应B表的多条记录，且A的主键作为B的外键。
* 多对多：一般需要张中间表，将两张表进行关联。如学生选课：实体表为学生表和课程表，关系表为选修表。

# mysql常用函数汇总

<http://www.jb51.net/article/40179.htm>

# 数据库三大范式

为了建立冗余较小、结构合理的数据库，设计数据库时必须遵循一定的规则。在关系型数据库中这种规则就称为范式。范式是符合某一种设计要求的总结。要想设计一个结构合理的关系型数据库，必须满足一定的范式。

## 一 确保每列保持原子性

第一范式是**最基本的范式**。如果数据库表中的**所有字段值都是不可分解的原子值**，就说明该数据库表满足了第一范式。

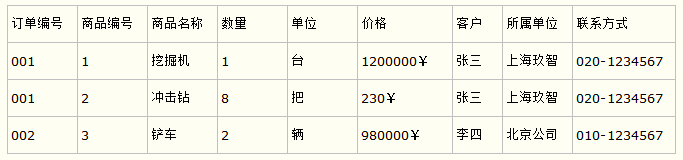
第一范式的合理遵循需要根据系统的实际需求来定。比如某些数据库系统中需要用到“地址”这个属性，本来直接将“地址”属性设计成一个数据库表的字段就行。但是如果系统经常会访问“地址”属性中的“城市”部分，那么就非要将“地址”这个属性重新拆分为省份、城市、详细地址等多个部分进行存储，这样在对地址中某一部分操作的时候将非常方便。这样设计才算满足了数据库的第一范式，如下表所示。



## 二 保表中的每列都和主键相关

第二范式在第一范式的基础之上更进一层。第二范式需要确保数据库表中的每一列都和主键相关，而不能只与主键的某一部分相关（主要针对联合主键而言）。也就是说**在一个数据库表中，一个表中只能保存一种数据，不可以把多种数据保存在同一张数据库表中**。

比如要设计一个订单信息表，因为订单中可能会有多种商品，所以要将订单编号和商品编号作为数据库表的联合主键，如下表所示。



这样就产生一个问题：这个表中是以订单编号和商品编号作为联合主键。这样在该表中商品名称、单位、商品价格等信息不与该表的主键相关，而仅仅是与商品编号相关。所以在这里违反了第二范式的设计原则。

而如果把这个订单信息表进行拆分，把商品信息分离到另一个表中，把订单项目表也分离到另一个表中，就非常完美了。如下所示。



这样设计，在很大程度上减小了数据库的冗余。如果要获取订单的商品信息，使用商品编号到商品信息表中查询即可。

## 三 保每列都和主键列直接相关,而不是间接相关

第三范式需要确保**数据表中的每一列数据都和主键直接相关，而不能间接相关。**

比如在设计一个订单数据表的时候，可以将客户编号作为一个外键和订单表建立相应的关系。而不可以在订单表中添加关于客户其它信息（比如姓名、所属公司等）的字段。如下面这两个表所示的设计就是一个满足第三范式的数据库表。

这样在查询订单信息的时候，就可以使用客户编号来引用客户信息表中的记录，也不必在订单信息表中多次输入客户信息的内容，减小了数据冗余。

