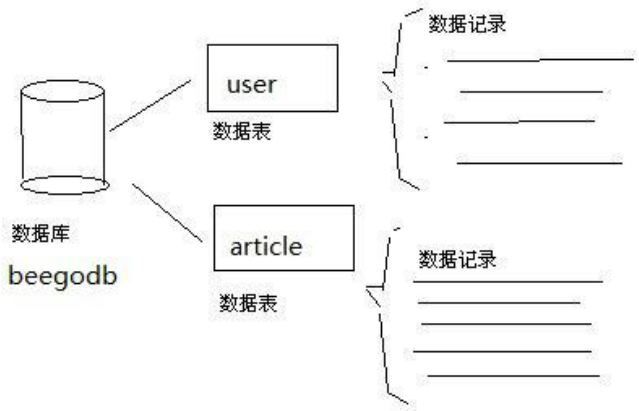
Mysql 常用命令

主讲教师：（大地）
合作网站：www.itying.com （IT 营）
我的专栏：https://www.itying.com/category-79-b0.html

一、数据库的组成



二、数据库的连接

```
C:\Users\Administrator>mysql -u root -p
Enter password: **************
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 11
Server version: 8.0.20 MySQL Community Server - GPL

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

三、Mysql 的基础命令

注意: mysql 命令是忽略大小写的

1) 显示当前存在的数据库

```
show  databases;
```

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sakila             |
| sys                |
| world              |
+--------------------+
6 rows in set (0.00 sec)
```

2)选择你所需要操作的数据库

```
use  beegodb;
```

```
mysql> use beegodb
Database changed
mysql> _
```

3)查看当前数据库有哪些表

```
show tables;
```

```
mysql> show tables;
+------------------+
| Tables_in_beegodb |
+------------------+
| article          |
| user             |
+------------------+
2 rows in set (0.00 sec)
```

4)查看一张表的所有内容

```
select * from user;
```

```
mysql> select * from user;
+----+----------+----------+
| id | username | password |
+----+----------+----------+
|  1 | zhangsan | 1324124  |
+----+----------+----------+
1 row in set (0.00 sec)
```

5) 数据库设置中文编码

```
set names utf8;
```

6) 创建一个数据库

```
create database book;
```

```
mysql> create database book;
Query OK, 1 row affected (0.03 sec)
```

7) 在数据库里创建一张表

```
create table users(
   username varchar(20),
   sex int(1),
   status int(1)
);
```

```
mysql> use book
Database changed
mysql> create table users(
    -> username varchar(20),
    -> sex int(1),
    -> status int(1)
    -> );
Query OK, 0 rows affected, 2 warnings (0.05 sec)
```

8)显示表的结构

```
describe users;
```

```
mysql> describe users;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| username | varchar(20) | YES  |     | NULL    |       |
| sex      | int         | YES  |     | NULL    |       |
| status   | int         | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

9)给表插入一条数据

```
insert into users(username,sex,status) values ("itying",1,1);
```

```
mysql> insert into users(username,sex,status) values ("golang",1,1);
Query OK, 1 row affected (0.02 sec)
```

```
mysql> select * from users;
+----------+------+--------+
| username | sex  | status |
+----------+------+--------+
| itying   |    1 |      1 |
| golang   |    1 |      1 |
| php      |    0 |      1 |
| beego    |    0 |      1 |
+----------+------+--------+
4 rows in set (0.00 sec)
```

10)根据条件筛选指定的数据

```
SELECT * FROM users WHERE username = 'itying';
```

```
mysql> select * from users where username="itying";
+----------+------+--------+
| username | sex  | status |
+----------+------+--------+
| itying   |    1 |      1 |
+----------+------+--------+
1 row in set (0.00 sec)
```

11)修改指定的数据

```
UPDATE users SET status = 10 WHERE username='itying';
```

```
mysql> UPDATE users SET status = 10 WHERE username='itying';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

```
mysql> SELECT * FROM USERS;
+----------+------+--------+
| username | sex  | status |
+----------+------+--------+
| itying   |    1 |     10 |
| golang   |    1 |      1 |
| php      |    0 |      1 |
| beego    |    0 |      1 |
+----------+------+--------+
4 rows in set (0.00 sec)
```

12)删除指定的数据

```
DELETE FROM users WHERE username='php';
```

```
mysql> SELECT * FROM USERS;
+----------+------+--------+
| username | sex  | status |
+----------+------+--------+
| itying   |    1 |     10 |
| golang   |    1 |      1 |
| php      |    0 |      1 |
| beego    |    0 |      1 |
+----------+------+--------+
4 rows in set (0.00 sec)

mysql> DELETE FROM users WHERE username='php'
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM USERS;
+----------+------+--------+
| username | sex  | status |
+----------+------+--------+
| itying   |    1 |     10 |
| golang   |    1 |      1 |
| beego    |    0 |      1 |
+----------+------+--------+
3 rows in set (0.00 sec)
```

13)按指定的数据排序

```
SELECT * FROM users ORDER BY status DESC;  //按照 status 倒叙排序
```

```
mysql> SELECT * FROM users ORDER BY status DESC;  //正序
+----------+------+--------+
| username | sex  | status |
+----------+------+--------+
| itying   |    1 |     10 |
| golang   |    1 |      1 |
| beego    |    0 |      1 |
+----------+------+--------+
3 rows in set (0.00 sec)
```

14)统计数量

```
select count(1) from nav
```

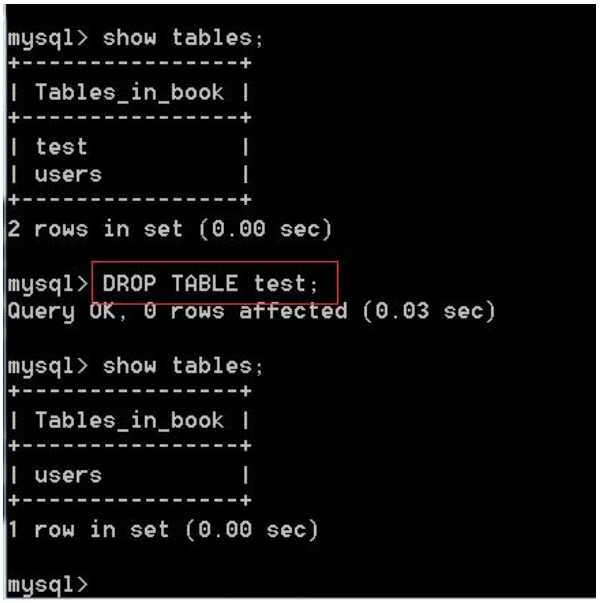
15)Limit

```
select id,name from nav limit 2;


select id,name from nav limit 2,2;
```

16)删除指定的表

```
DROP  TABLE  test;
```



15)删除指定的数据库

```
DROP  DATABASE  book;
```

三、MySQL 字段的常用数据类型

整数型：TINYINT，SMALLINT，INT，BIGINT

浮点型：FLOAT，DOUBLE，DECIMAL(M,D)

字符型：CHAR，VARCHAR

备注型：TINYTEXT，TEXT，LONGTEXT

日期型：DATETIME，DATE，TIMESTAMP（了解）

1、整数型

类型	字节	最小值	最大值
		(带符号的/无符号的)	(带符号的/无符号的)
TINYINT	1	-128	127
		0	255
SMALLINT	2	-32768	32767
		0	65535
MEDIUMINT	3	-8388608	8388607
		0	16777215
INT	4	-2147483648	2147483647
		0	4294967295
BIGINT	8	-9223372036854775808	9223372036854775807
		0	18446744073709551615

TINYINT 最大长度 4

SMALLINT 最大长度 6

MEDIUMINT 最大长度 8

int 最大长度是 11 位：如果在建表时不指定字段 int 类型的长度时，系统则默认生成长度为 11 的字段。11 也是 int 类型的最大长度，其中第一位表示符号+或者-，后面十位表示数字

BIGINT 最大长度 20：

注意：int(M) 在 integer 数据类型中，M 表示最大显示宽度。在 int(M) 中，M 的值跟 int(M) 所占多少存储空间并无任何关系。和数字位数也无关系 int(3)、int(4)、int(8) 在磁盘上都是占用 4 btyes 的存储空间。

int(11)，tinyint(1)，bigint(20)，后面的数字，不代表占用空间容量。而代表最小显示位数。这个东西基本没有意义，除非你对字段指定 zerofill。

所以我们在设计 mysql 数据库时，建表时，mysql 会自动分配长度：int(11)、tinyint(4)、smallint(6)、mediumint(9)、bigint(20)。

所以，就用这些默认的显示长度就可以了。不用再去自己填长度，比如搞个 int(10)、tinyint(1) 之类的，基本没用。而且导致表的字段类型多样化。

2、浮点型

类型	字节	最小值	最大值
FLOAT	4	+-1.175494351E-38	+-3.402823466E+38

DOUBLE	8	+-2.2250738585072014E-308	+-1.7976931348623157E+308
DECIMAL	可变	它的取值范围可变。	

FLOAT 和 DOUBLE 在不指定精度时，默认会按照实际的精度（由计算机硬件和操作系统决定），DECIMAL 如果不指定精度，默认为（10，0）。

浮点数相对于定点数的优点是在长度一定的情况下，浮点数能够表示更大的范围；缺点是会引起精度问题。

3、字符串型

值	CHAR(4)	存储需求	VARCHAR(4)	存储需求
''	' '	4 个字节	''	1 个字节
'ab'	'ab '	4 个字节	'ab '	3 个字节
'abcd'	'abcd'	4 个字节	'abcd'	5 个字节
'abcdefgh'	'abcd'	4 个字节	'abcd'	5 个字节

varchar 使用额外的 1-2 字节内来存储值长度，列长度<=255 使用 1 字节保存，其它情况使用 2 字节保存。例如 varchar(10)会占用 11 字节存储空容间，varchar(500)会占用 502 字节存储空间。

varchar 最大长度可以是多少？

根据字符集，字符类型若为 gbk，每个汉子占用 2 个字节，最大长度不能超过 65535/2 =32766；字符类型若为 utf8，每个汉子最多占用 3 个字节，最大长度不能超过 65535/3 =21845，若超过这个限制，则会自动将 varchar 类型转为 mediumtext 或 longtext,

4、备注型

类	描述	
TINYTEXT	字符串，最大长度 255 个字符	
TEXT	字符串，最大长度 65535 个字符	
MEDIUMTEXT	字符串，最大长度 16777215 个字符	
LONGTEXT	字符串，最大长度 4294967295 个字符	

5、日期型（实际项目很少用）

列类型	零值
DATETIME	'0000-00-00 00:00:00'

DATE	'0000-00-00'
TIMESTAMP	00000000000000
TIME	'00:00:00'
YEAR	0000

实际项目中我们存储日期用的都是 int 类型的时间戳。

四、Mysql 查询语句详解

1、创建一个班级数据库 school，里面包含一张班级表 class，包含编号(id)、姓名(name)、邮件(email)、成绩(score)

创建一个 school 的数据库

```
create database school;
```

创建一个 class 表里面有 id、name、email、score 字段

```
CREATE TABLE class (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(255),
  `email` varchar(255),
  `score` tinyint(4),
  PRIMARY KEY (`id`)
)
```

PRIMARY KEY (`id`) 表示 id 为主键，让 id 值唯一，不得重复。

2、给这个班级表 class 新增 5-10 条学员记录

```
INSERT INTO `class` VALUES (1, '张三', 'itying@qq.com', 89);
INSERT INTO `class` VALUES (2, '李四', '441@qq.com', 87);
INSERT INTO `class` VALUES (3, '王五', 'nodejs@qq.com', 98);
INSERT INTO `class` VALUES (4, '赵四', 'java@qq.com', 59);
INSERT INTO `class` VALUES (5, '小马', '888@qq.com', 66);
INSERT INTO `class` VALUES (6, '小李', '999@hotmail.com', 88);
INSERT INTO `class` VALUES (7, '小张', 'golang@163.com', 55);
```

3、查看班级所有字段的记录以及查看班级 id,name 的记录

```
select * from class;
```

```
select id,name from class;
```

WHERE 表达式的常用运算符

MYSQL 运算符	含义
=	等于
<	小于
>	大于
<=	小于或等于
>=	大于或等于
!=	不等于
IS NOT NULL	具有一个值
IS NULL	没有值
BETWEEN	在范围内
NOT BETWEEN	不在范围内
IN	指定的范围
OR	两个条件语句之一为真
AND	两个条件语句都为真
NOT	条件语句不为真
LIKE	搜索列中的指定模式

查找 emial 不为空的数据

```
select * from class where email is not null;
```

查找 emial 为空的数据

```
select * from class where email is null;
```

查找成绩大于等于 70 小于等于 90 的数据

```
select * from class where score>=70 AND score<=99;

select * from class where score between 70 and 99;

select * from class where score not between 70 and 99;
```

查找 score 为 89 和 98 的数据

```
select * from class where score in (89,98);
```

查找 score 不是 89 和 98 的数据

```
select * from class where score not in (89,98);
```

Or And

```
select * from class where score=87 or score=88;

select * from class where score>=70 AND score<=99;
```

模糊查询

```
select * from class where email like "%qq%";

select * from class where email like "ja%";                    // ja 开头的

select * from class where email like "%163.com";          //163 结尾的

select * from class where email not like "%163.com";        //不是 163 结尾的
```

五、Mysql 分组函数

函数	用法	描述
AVG()	AVG(column)	返回列的平均值
COUNT()	COUNT(column)	统计行数
MAX()	MAX(column)	求列中的最大值
MIN()	MIN(column)	求列中的最小值
SUM()	SUM(column)	求列中的和

统计班级的平均分

```
select avg(score) from class;
```

统计班级一共多少人

```
select count(1) from class;
```

找出这个班成绩最高的学生

```
select max(score) from class;

select * from class where score in (select max(score) from class);
```

找出这个班成绩最低的学生

```
select min(score) from class;

select * from class where score in (select min(score) from class);
```

统计这个班的总分

```
select sum(score) from class;

统计一年的销售额
```

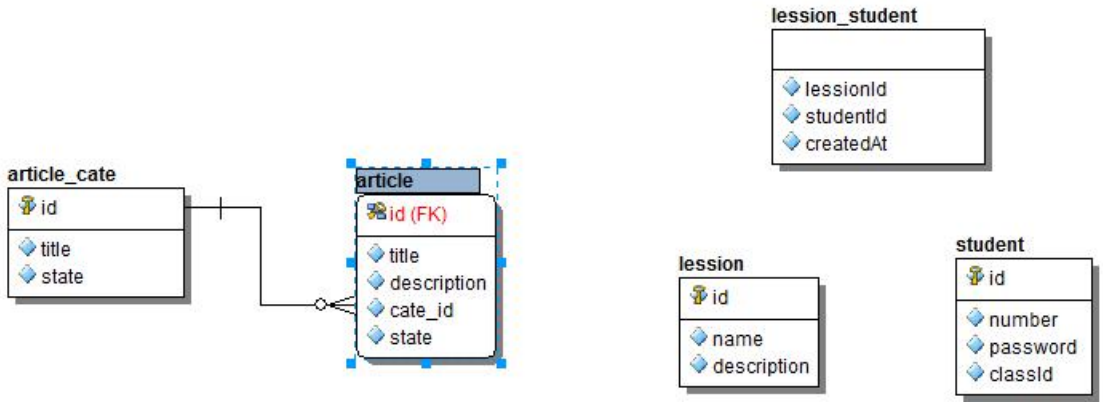
六、Mysql 别名

```
select name as a from class;

select min(score) as minscore from class;
```

七、Mysql 表与表之间的三种关系

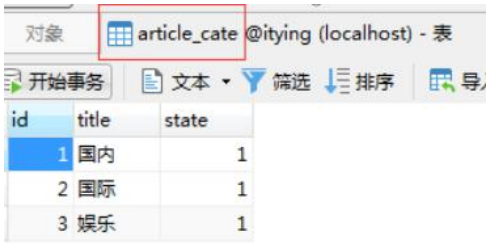
表与表之间一般存在三种关系，即一对一，一对多，多对多关系。



1、一对一

查找一个文章并显示每个文章的分类

id	title	cate_id	state
1	8月份CPI同比上涨2.8% 猪肉价格	1	1
2	中国联通与中国电信共建共享5G网	1	1
3	林郑月娥斥责暴徒破坏港铁:不能因	2	1
4	这些老师的口头禅，想起那些年	2	1
5	美国空军一号差点遭雷劈，特朗普	3	1



id	title	state
1	国内	1
2	国际	1
3	娱乐	1

```
select article.id as id,article.title as title,article_cate.title as cate from article,article_cate Where article.cate_id=article_cate.id
```

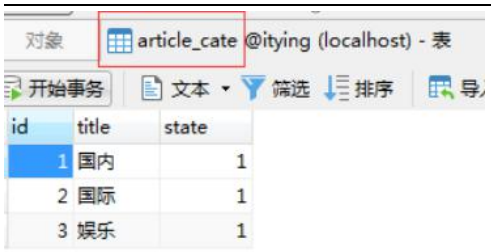
id	title	cate
1	8月份CPI同比上涨2.8% 猪肉价格上涨46.7%	国内
2	中国联通与中国电信共建共享5G网络 用户归属不变	国内
3	林郑月娥斥责暴徒破坏港铁:不能因为没生命就肆意破坏	国际
4	这些老师的口头禅，想起那些年 “被支配的恐惧” 了吗	国际
5	美国空军一号差点遭雷劈，特朗普惊呼：令人惊奇	娱乐

```
select article.id as id,article.title as title,article_cate.title as cate from article INNER JOIN article_cate ON article_cate.id=article.cate_id
```

id	title	cate
1	8月份CPI同比上涨2.8% 猪肉价格上涨46.7%	国内
2	中国联通与中国电信共建共享5G网络 用户归属不变	国内
3	林郑月娥斥责暴徒破坏港铁:不能因为没生命就肆意破坏	国际
4	这些老师的口头禅，想起那些年 “被支配的恐惧” 了吗	国际
5	美国空军一号差点遭雷劈，特朗普惊呼：令人惊奇	娱乐

2、一对多

查看国内新闻下面的所有文章

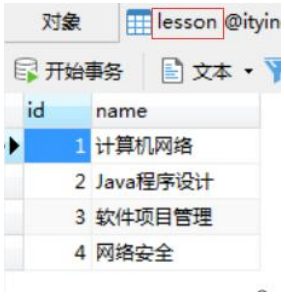


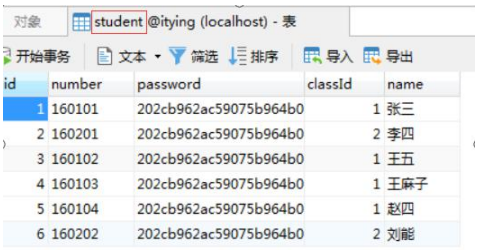

```
select * from article where cate_id=2
```

id	title	cate_id	state
3	林郑月娥斥责暴徒破坏港铁:不能因为没生命就肆意破坏	2	1
4	这些老师的口头禅，想起那些年 “被支配的恐惧”了吗	2	1

3、多对多

一个学生可以选修多门课程，一门课程也可以被多个学生选修。





1、查询张三选修了那些课程

张三的 Id 为 1

普通查询

```
SELECT * FROM lesson where id in (select lesson_id from lesson_student WHERE student_id=1)
```

笛卡尔积关联查询

```
SELECT * FROM lesson,lesson_student where lesson.id=lesson_student.lesson_id AND lesson_student.student_id=1
```

INNER JOIN(内连接)

```
SELECT * FROM lesson INNER JOIN lesson_student ON lesson.id=lesson_student.lesson_id AND lesson_student.student_id=1
```

2、查询 Java 程序设计被那些学生选修了

Java 程序设计的 id 为 2

```
SELECT * FROM student where id in (select student_id from lesson_student WHERE lesson_id=2)
```

```
SELECT * FROM student,lesson_student where student.id=lesson_student.student_id AND lesson_student.lesson_id=2
```

```
SELECT * FROM student INNER JOIN lesson_student ON student.id=lesson_student.student_id AND lesson_student.lesson_id=2
```

八、Mysql 笛卡尔积连接、内连接、左外连接、右外连接

查询数据的时候能不用连接语句尽量不要使用，**笛卡尔积连接**查询速度最慢，项目中用的比较多的是**内连接**。

笛卡尔积连接：

```
SELECT * FROM student,lesson_student where student.id=lesson_student.student_id AND lesson_student.lesson_id=2
```

内连接：

```
SELECT * FROM student INNER JOIN lesson_student ON student.id=lesson_student.student_id AND lesson_student.lesson_id=2
```

左外连接：

```
SELECT * FROM student LEFT JOIN lesson_student ON student.id=lesson_student.student_id AND lesson_student.lesson_id=2
```

右外连接：

```
SELECT * FROM student RIGHT JOIN lesson_student ON student.id=lesson_student.student_id AND lesson_student.lesson_id=2
```

九、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。

如果没有索引，执行查询时候必须从第一条记录开始，扫描整个表的记录，直到符合要求的记录。如果有了索引，mysql 无需扫描任何记录即可顺序找到目标记录的位置。简单说来， 索引就是提高查找数据速度，数据量越多，效果越明显。

Mysql 中常见的索引类型有普通索引、唯一索引、全文索引、空间索引 Spatial

用 50 万条数据演示：

```
INSERT INTO users (`username`) SELECT username from users
```

1、创建普通索引

基本语法：

```
CREATE INDEX indexName ON mytable(username);
```

```
create index index_name on class(name);
```

2、查看索引

```
show index from table_name
```

```
show index from class
```

```
show index from class\G
```

3、删除索引

```
drop index index_name on class;
```

4、创建唯一索引（主键是一种唯一索引）

```
create unique index index_name on class(name);
```

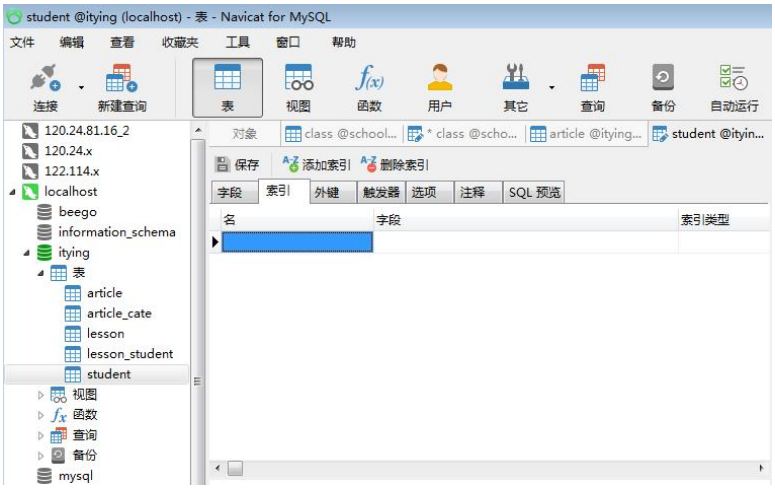
5、另外的一种创建和删除方式

```
alter table class add index index_name(name);

alter table class add unique index_name(name);

alter table class drop index index_name;
```

6、使用可视化工具创建索引



十、Mysql 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。

```
SET NAMES utf8mb4;

SET FOREIGN_KEY_CHECKS = 0;

-- ----------------------------

-- Table structure for user

-- ----------------------------

DROP TABLE IF EXISTS `user`;

CREATE TABLE `user`  (

  `id` int(0) NOT NULL AUTO_INCREMENT,

  `username` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci NULL DEFAULT NULL,

  `balance` decimal(10, 2) NULL DEFAULT NULL,

  PRIMARY KEY (`id`) USING BTREE

) ENGINE = InnoDB CHARACTER SET = utf8mb4 COLLATE = utf8mb4_0900_ai_ci ROW_FORMAT = Dynamic;
```

```
-- ----------------------------

-- Records of user

-- ----------------------------

INSERT INTO `user` VALUES (1, '张三', 100.00);

INSERT INTO `user` VALUES (2, '李四', 100.00);


SET FOREIGN_KEY_CHECKS = 1;
```

例子：张三账户转账转出 100 元到李四的账户

1、张三账户减去 100 元

2、李四账户增加 100 元

```
UPDATE user set balance = balance-100 WHERE id=1

UPDATE user set balance = balance+100 WHERE id=2
```

如果我们更新完张三的账户后，准备更新李四账户的时候出现了错误（比如程序错误，或者数据库没法连接、或者异常断电等错误）。这样的话就导致了数据不一致。为了保证数据的一致性，这个时候我们就可以使用事务。

Mysql 中用 BEGIN, ROLLBACK, COMMIT 来实现事务

BEGIN 开始一个事务

ROLLBACK 事务回滚

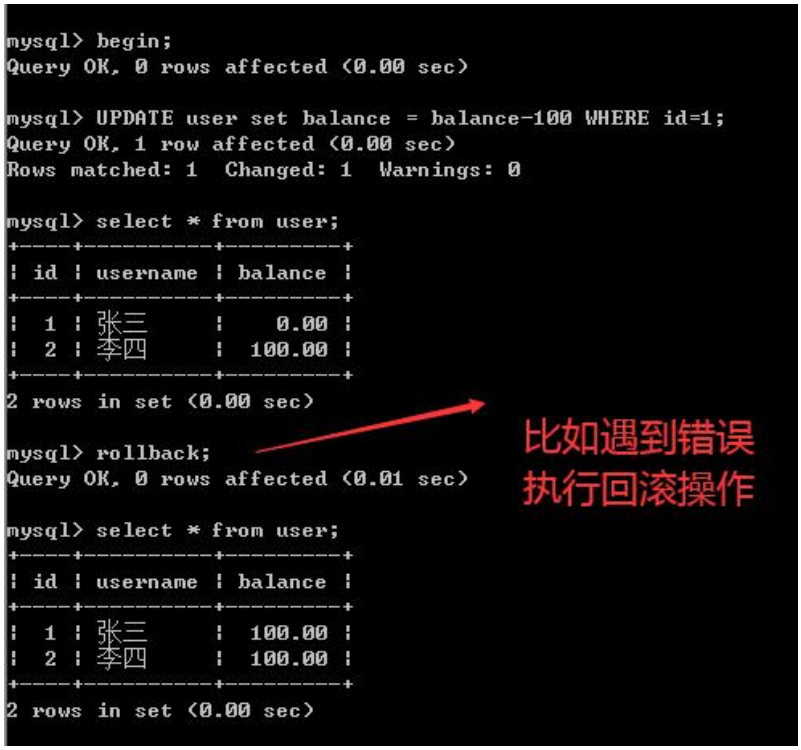
COMMIT 事务确认

```
mysql> select * from user;
+----+----------+---------+
| id | username | balance |
+----+----------+---------+
|  1 | 张三     |  100.00 |
|  2 | 李四     |  100.00 |
+----+----------+---------+
2 rows in set (0.00 sec)
```

```
begin;

update user set balance = balance-100 where id=1
```

```
rollback;
```

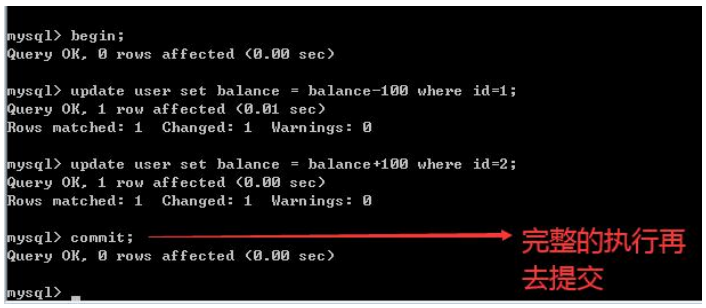



```
begin;

update user set balance = balance-100 where id=1;

update user set balance = balance+100 where id=2;

commit;
```



十一、Mysql 锁

Mysql 中的锁有**表级锁**和**行级锁，**这里主要给大家讲讲最常用的表级锁

1、添加读锁

可以并发读，但是不能并发写，读锁期间，没释放锁之前不能进行写操作。

使用场景：读取结果集的最新版本，同时防止其他事务产生更新该结果集
主要用在需要数据依存关系时确认某行记录是否存在，并确保没有人对这个记录进行 UPDATE 或者 DELETE 操作

```
lock table user read;

unlock tables;
```

2、添加写锁

只有锁表的用户可以进行读写操作，其他用户不行 （并发下对商品库存的操作）

```
lock table user write;

unlock tables;
```