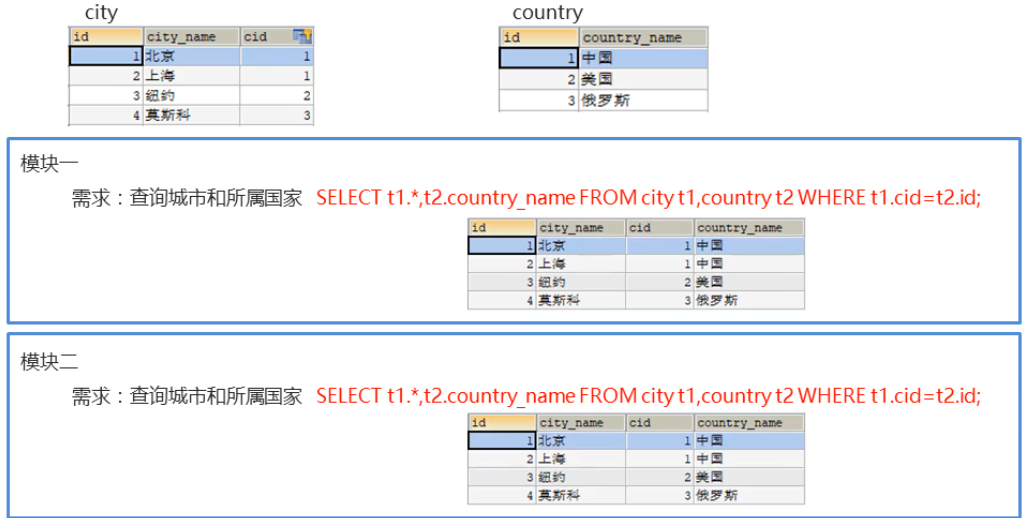
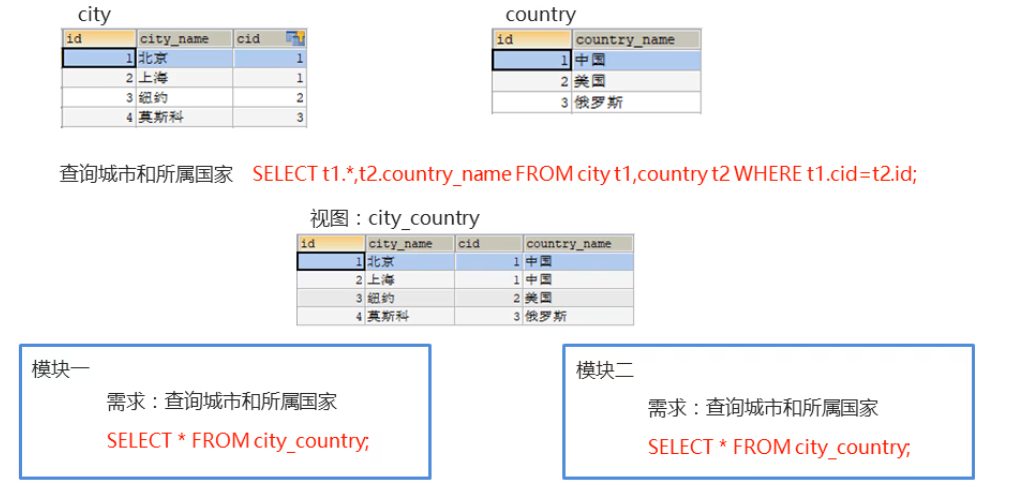
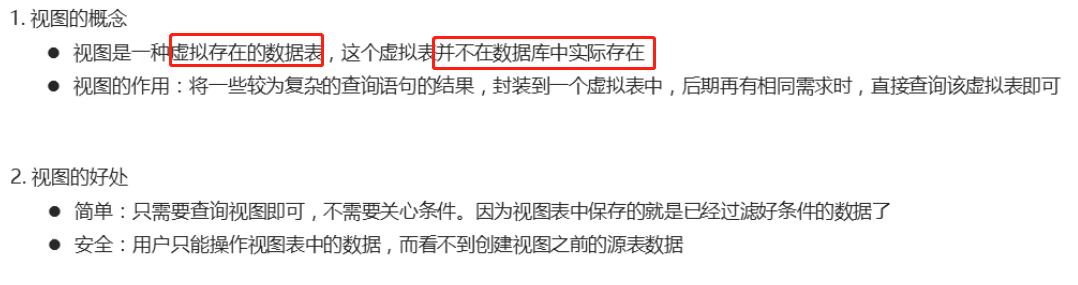
03 mysql高级

# 1（掌握）MySQL视图

## 1.1（掌握）视图简介







## 1.2（掌握）视图的数据准备

### 数据准备

|  |
| --- |
| -- 创建db5数据库  CREATE DATABASE db5;  -- 使用db5数据库  USE db5;  -- 创建country表  CREATE TABLE country(  id INT PRIMARY KEY AUTO\_INCREMENT, -- 国家id  NAME VARCHAR(30) -- 国家名称  );  -- 添加数据  INSERT INTO country VALUES (NULL,'中国'),(NULL,'美国'),(NULL,'俄罗斯');  -- 创建city表  CREATE TABLE city(  id INT PRIMARY KEY AUTO\_INCREMENT, -- 城市id  NAME VARCHAR(30), -- 城市名称  cid INT, -- 外键列。关联country表的主键列id  CONSTRAINT cc\_fk1 FOREIGN KEY (cid) REFERENCES country(id) -- 添加外键约束  );  -- 添加数据  INSERT INTO city VALUES (NULL,'北京',1),(NULL,'上海',1),(NULL,'纽约',2),(NULL,'莫斯科',3); |

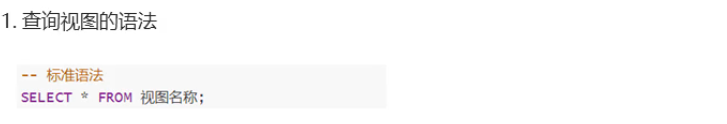
## 1.3（掌握）创建视图



### 练习

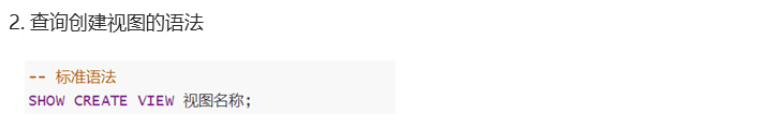
|  |
| --- |
| /\*  创建视图  标准语法  CREATE VIEW 视图名称 [(列名列表)] AS 查询语句;  \*/  -- 创建city\_country视图，保存城市和国家的信息(使用指定列名)  CREATE VIEW city\_country (city\_id,city\_name,country\_name) AS  SELECT  c1.id,  c1.name,  c2.name  FROM  city c1,  country c2  WHERE  c1.cid=c2.id; |

## 1.4（掌握）查询视图



### 练习

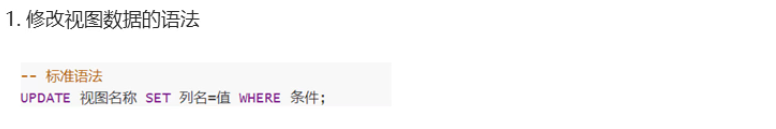
|  |
| --- |
| /\*  查询视图  标准语法  SELECT \* FROM 视图名称;  \*/  -- 查询视图  SELECT \* FROM city\_country; |



### 练习

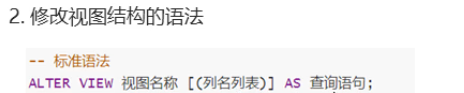
|  |
| --- |
| /\*  查询创建视图的语句  标准语法  SHOW CREATE VIEW 视图名称;  \*/  -- 查询city\_country的创建语句  SHOW CREATE VIEW city\_country; |

## 1.5（掌握）修改视图



### 练习

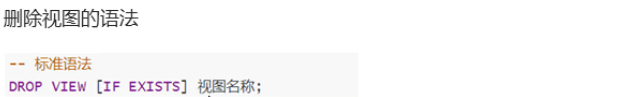
|  |
| --- |
| /\*  修改视图数据  标准语法  UPDATE 视图名称 SET 列名=值 WHERE 条件;  \*/  -- 修改视图数据,将北京修改为深圳。(注意：修改视图数据后，源表中的数据也会随之修改)  UPDATE city\_country SET city\_name='深圳' WHERE city\_name='北京';  -- 查询city\_country视图  SELECT \* FROM city\_country;  -- 查询city表  SELECT \* FROM city; |



### 练习

|  |
| --- |
| /\*  修改视图结构  标准语法  ALTER VIEW 视图名称 (列名列表) AS 查询语句;  \*/  -- 将视图中的country\_name修改为name  ALTER VIEW city\_country (city\_id,city\_name,NAME) AS  SELECT  c1.id,  c1.name,  c2.name  FROM  city c1,  country c2  WHERE  c1.cid=c2.id; |

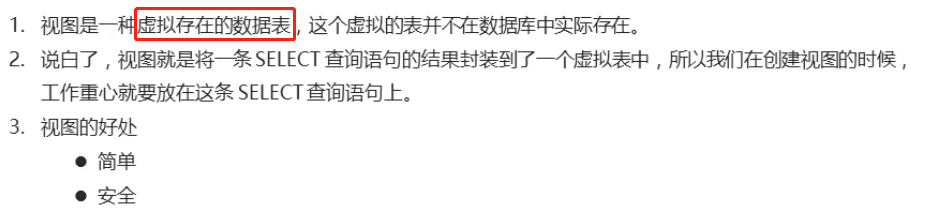
## 1.6（掌握）删除视图



### 练习

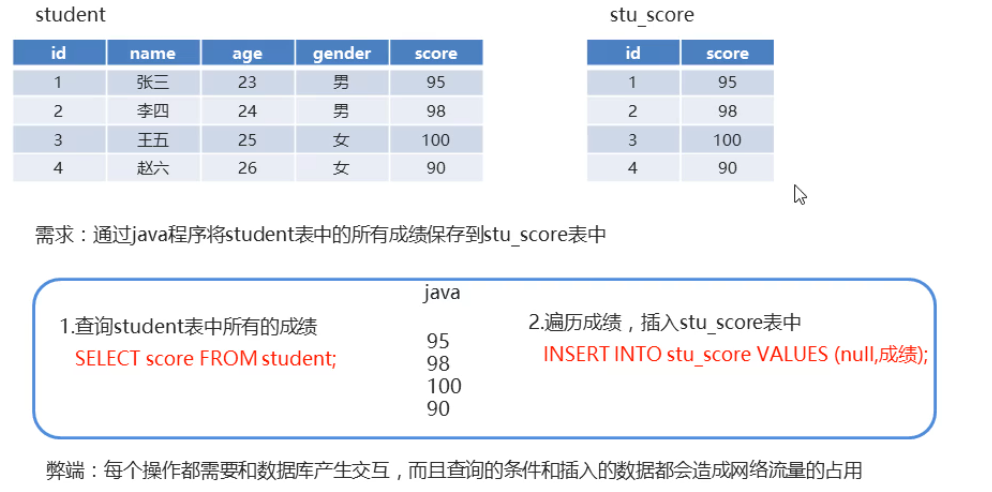
|  |
| --- |
| /\*  删除视图  标准语法  DROP VIEW [IF EXISTS] 视图名称;  \*/  -- 删除city\_country视图  DROP VIEW IF EXISTS city\_country; |

## 1.7（掌握）视图的总结

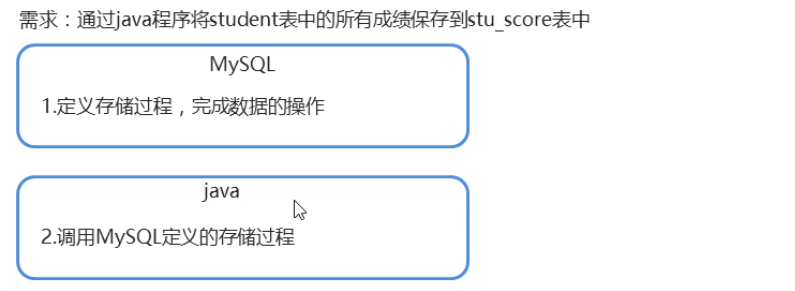


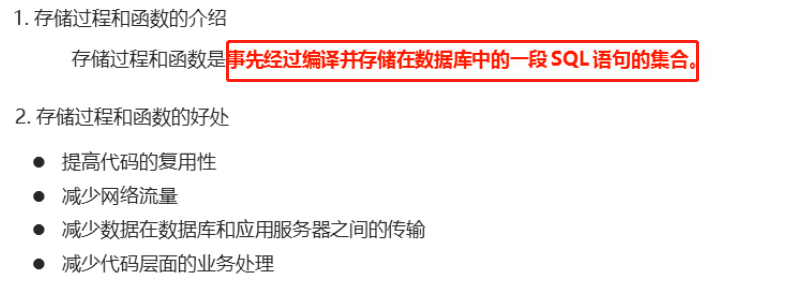
# 2（掌握）MySQL存储过程和函数

## 2.1（掌握）存储过程的简介









## 2.2（掌握）数据准备

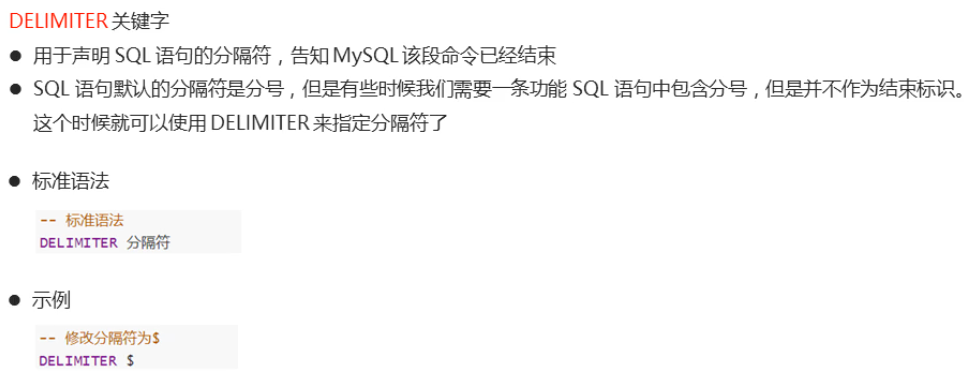
### 数据准备

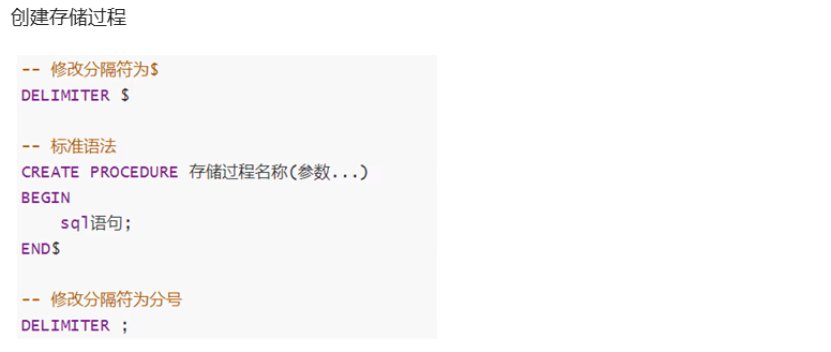
|  |
| --- |
| -- 创建db6数据库  CREATE DATABASE db6;  -- 使用db6数据库  USE db6;  -- 创建学生表  CREATE TABLE student(  id INT PRIMARY KEY AUTO\_INCREMENT, -- 学生id  NAME VARCHAR(20), -- 学生姓名  age INT, -- 学生年龄  gender VARCHAR(5), -- 学生性别  score INT -- 学生成绩  );  -- 添加数据  INSERT INTO student VALUES (NULL,'张三',23,'男',95),(NULL,'李四',24,'男',98),  (NULL,'王五',25,'女',100),(NULL,'赵六',26,'女',90); |

### 练习

|  |
| --- |
| -- 按照性别进行分组，查询每组学生的总成绩。按照总成绩的升序排序  SELECT gender,SUM(score) getSum FROM student GROUP BY gender ORDER BY getSum ASC; |

## 2.3（掌握）创建存储过程





### 练习

|  |
| --- |
| /\*  创建存储过程  -- 修改分隔符为$  DELIMITER $  -- 标准语法  CREATE PROCEDURE 存储过程名称(参数列表)  BEGIN  SQL 语句列表;  END$  -- 修改分隔符为分号  DELIMITER ;  \*/  -- 创建stu\_group()存储过程，封装 分组查询总成绩，并按照总成绩升序排序的功能  DELIMITER $  CREATE PROCEDURE stu\_group()  BEGIN  SELECT gender,SUM(score) getSum FROM student GROUP BY gender ORDER BY getSum ASC;  END$  DELIMITER ; |

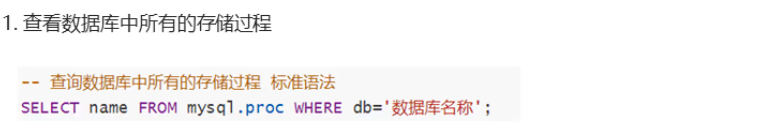
## 2.4（掌握）调用存储过程



### 练习

|  |
| --- |
| /\*  调用存储过程  CALL 存储过程名称(实际参数);  \*/  -- 调用stu\_group()存储过程  CALL stu\_group(); |

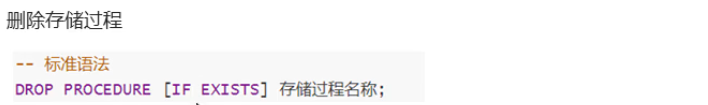
## 2.5（掌握）查看存储过程



### 练习

|  |
| --- |
| /\*  查询数据库中所有的存储过程  SELECT \* FROM mysql.proc WHERE db='数据库名称';  \*/  -- 查看db6数据库中所有的存储过程  SELECT \* FROM mysql.proc WHERE db='db6'; |

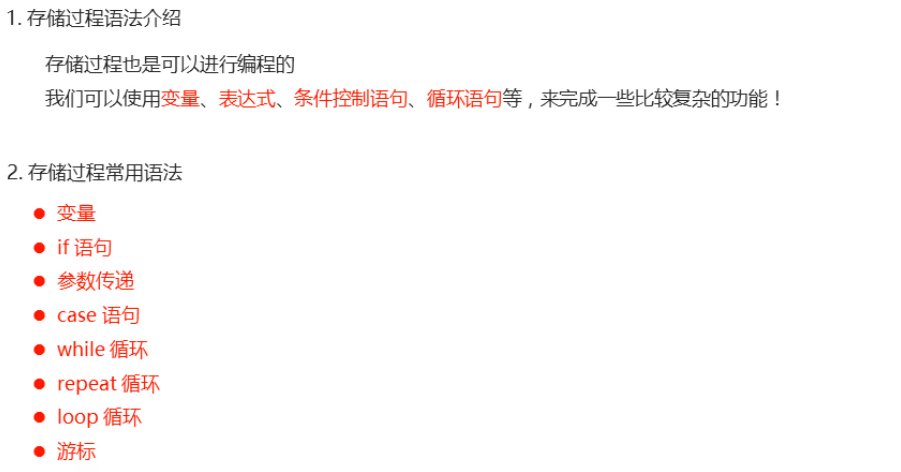
## 2.6（掌握）删除存储过程



### 练习

|  |
| --- |
| /\*  删除存储过程  DROP PROCEDURE [IF EXISTS] 存储过程名称;  \*/  DROP PROCEDURE IF EXISTS stu\_group; |

## 2.7（掌握）变量





### 练习

|  |
| --- |
| /\*  定义变量  DECLARE 变量名 数据类型 [DEFAULT 默认值];  \*/  -- 定义一个int类型变量，并赋默认值为10  DELIMITER $  CREATE PROCEDURE pro\_test1()  BEGIN  -- 定义变量  DECLARE num INT DEFAULT 10;  -- 使用变量  SELECT num;  END$  DELIMITER ;  -- 调用pro\_test1存储过程  CALL pro\_test1(); |



### 练习

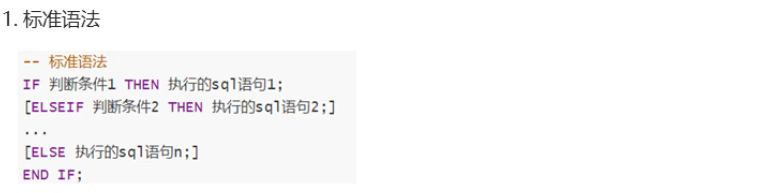
|  |
| --- |
| /\*  变量赋值-方式一  SET 变量名 = 变量值;  \*/  -- 定义一个varchar类型变量并赋值  DELIMITER $  CREATE PROCEDURE pro\_test2()  BEGIN  -- 定义变量  DECLARE NAME VARCHAR(10);  -- 为变量赋值  SET NAME = '存储过程';  -- 使用变量  SELECT NAME;  END$  DELIMITER ;  -- 调用pro\_test2存储过程  CALL pro\_test2(); |

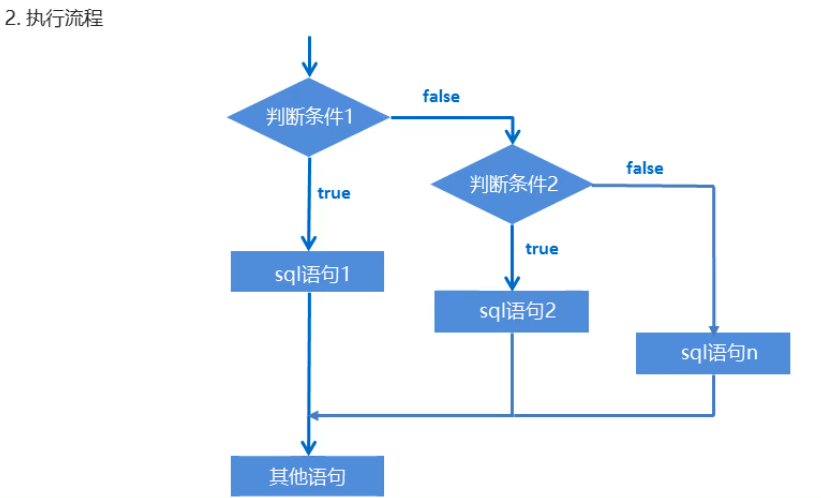


### 练习

|  |
| --- |
| /\*  变量赋值-方式二  SELECT 列名 INTO 变量名 FROM 表名 [WHERE 条件];  \*/  -- 定义两个int变量，用于存储男女同学的总分数  DELIMITER $  CREATE PROCEDURE pro\_test3()  BEGIN  -- 定义两个变量  DECLARE men,women INT;  -- 查询男同学的总分数，为men赋值  SELECT SUM(score) INTO men FROM student WHERE gender='男';  -- 查询女同学的总分数，为women赋值  SELECT SUM(score) INTO women FROM student WHERE gender='女';  -- 使用变量  SELECT men,women;  END$  DELIMITER ;  -- 调用pro\_test3存储过程  CALL pro\_test3(); |

## 2.8（掌握）if语句

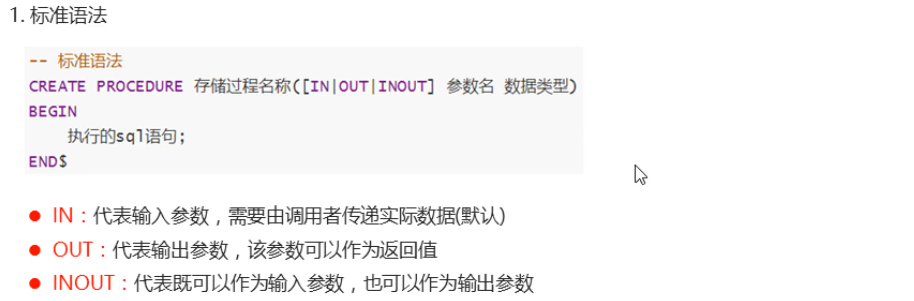




### 练习

|  |
| --- |
| /\*  if语句  IF 判断条件1 THEN 执行的sql语句1;  [ELSEIF 判断条件2 THEN 执行的sql语句2;]  ...  [ELSE 执行的sql语句n;]  END IF;  \*/  /\*  定义一个int变量，用于存储班级总成绩  定义一个varchar变量，用于存储分数描述  根据总成绩判断：  380分及以上 学习优秀  320 ~ 380 学习不错  320以下 学习一般  \*/  DELIMITER $  CREATE PROCEDURE pro\_test4()  BEGIN  -- 定义变量  DECLARE total INT;  DECLARE info VARCHAR(10);  -- 查询总成绩，为total赋值  SELECT SUM(score) INTO total FROM student;  -- 对总成绩判断  IF total > 380 THEN  SET info = '学习优秀';  ELSEIF total >= 320 AND total <= 380 THEN  SET info = '学习不错';  ELSE  SET info = '学习一般';  END IF;  -- 查询总成绩和描述信息  SELECT total,info;  END$  DELIMITER ;  -- 调用pro\_test4存储过程  CALL pro\_test4(); |

## 2.9（掌握）参数传递的概念



## 2.10（掌握）输入参数



### 练习

|  |
| --- |
| /\*  参数传递  CREATE PROCEDURE 存储过程名称([IN|OUT|INOUT] 参数名 数据类型)  BEGIN  SQL 语句列表;  END$  \*/  /\*  输入总成绩变量，代表学生总成绩  输出分数描述变量，代表学生总成绩的描述信息  根据总成绩判断：  380分及以上 学习优秀  320 ~ 380 学习不错  320以下 学习一般  \*/  DELIMITER $  CREATE PROCEDURE pro\_test5(IN total INT)  BEGIN  -- 对总成绩判断  IF total > 380 THEN  SET info = '学习优秀';  ELSEIF total >= 320 AND total <= 380 THEN  SET info = '学习不错';  ELSE  SET info = '学习一般';  END IF;  -- 查询总成绩和描述信息  SELECT total,info;  END$  DELIMITER ;  -- 调用pro\_test5存储过程  CALL pro\_test5(350);  CALL pro\_test5((SELECT SUM(score) FROM student)); |

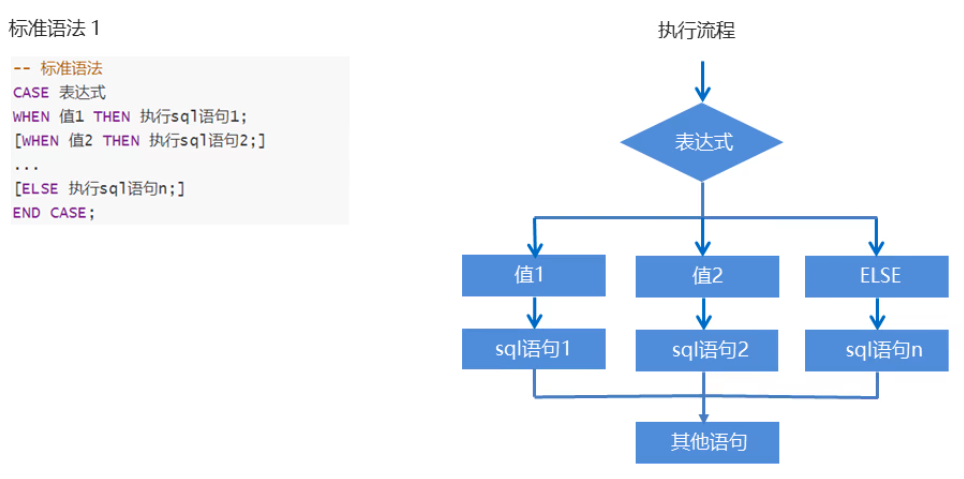
## 2.11（掌握）输出参数

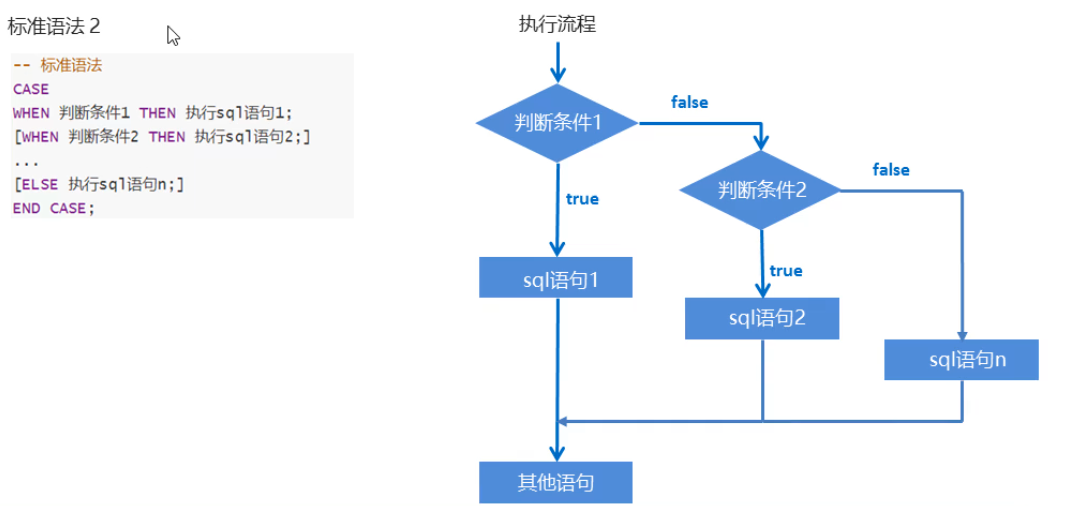


### 练习

|  |
| --- |
| /\*  参数传递  CREATE PROCEDURE 存储过程名称([IN|OUT|INOUT] 参数名 数据类型)  BEGIN  SQL 语句列表;  END$  \*/  /\*  输入总成绩变量，代表学生总成绩  输出分数描述变量，代表学生总成绩的描述信息  根据总成绩判断：  380分及以上 学习优秀  320 ~ 380 学习不错  320以下 学习一般  \*/  DELIMITER $  CREATE PROCEDURE pro\_test6(IN total INT,OUT info VARCHAR(10))  BEGIN  -- 对总成绩判断  IF total > 380 THEN  SET info = '学习优秀';  ELSEIF total >= 320 AND total <= 380 THEN  SET info = '学习不错';  ELSE  SET info = '学习一般';  END IF;  END$  DELIMITER ;  -- 调用pro\_test6存储过程  CALL pro\_test6(350,@info);  CALL pro\_test6((SELECT SUM(score) FROM student),@info);  SELECT @info; |

## 2.12（掌握）case语句

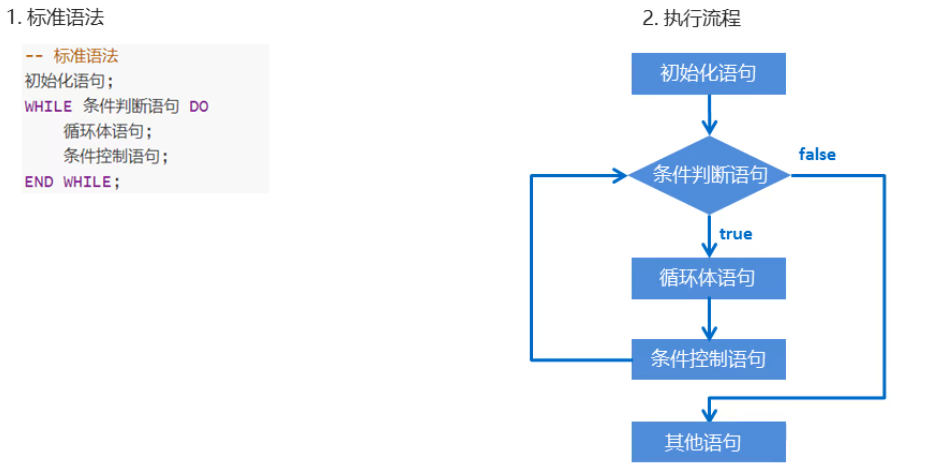




### 练习

|  |
| --- |
| /\*  CASE语句  CASE  WHEN 判断条件1 THEN 执行sql语句1;  [WHEN 判断条件2 THEN 执行sql语句2;]  ...  [ELSE 执行sql语句n;]  END CASE;  \*/  /\*  输入总成绩变量，代表学生总成绩  输出分数描述变量，代表学生总成绩的描述信息  根据总成绩判断：  380分及以上 学习优秀  320 ~ 380 学习不错  320以下 学习一般  \*/  DELIMITER $  CREATE PROCEDURE pro\_test7(IN total INT)  BEGIN  -- 定义总成绩描述信息变量  DECLARE info VARCHAR(10);  -- 判断总成绩  CASE  WHEN total > 380 THEN  SET info = '学习优秀';  WHEN total >= 320 AND total <= 380 THEN  SET info = '学习不错';  ELSE  SET info = '学习一般';  END CASE;  END$  DELIMITER ;  -- 调用pro\_test7存储过程  CALL pro\_test7(310); |

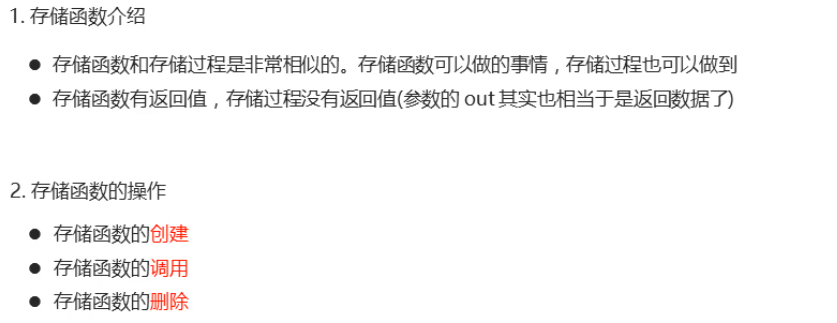
## 2.13（掌握）while循环

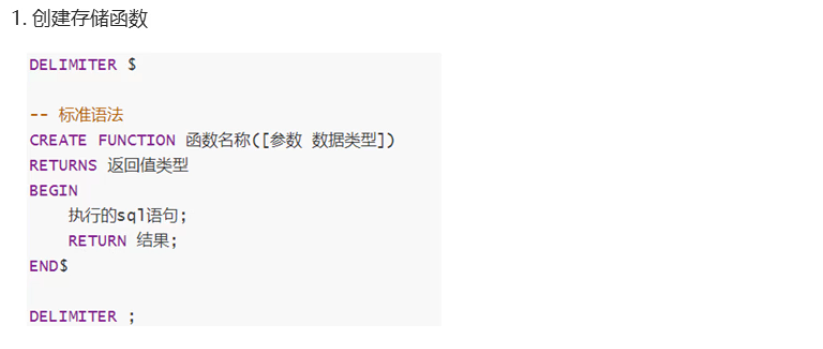


### 练习

|  |
| --- |
| /\*  while循环  初始化语句;  WHILE 条件判断语句 DO  循环体语句;  条件控制语句;  END WHILE;  \*/  -- 计算1~100之间的偶数和  DELIMITER $  CREATE PROCEDURE pro\_test8()  BEGIN  -- 定义求和变量  DECLARE result INT DEFAULT 0;  -- 定义初始化变量  DECLARE num INT DEFAULT 1;  -- while循环  WHILE num <= 100 DO  IF num % 2 = 0 THEN  SET result = result + num;  END IF;    SET num = num + 1;  END WHILE;    -- 查询求和结果  SELECT result;  END$  DELIMITER ;  -- 调用pro\_test8存储过程  CALL pro\_test8(); |

## 2.14（掌握）存储函数的操作





### 练习

|  |
| --- |
| /\*  创建存储函数  CREATE FUNCTION 函数名称([参数 数据类型])  RETURNS 返回值类型  BEGIN  执行的sql语句;  RETURN 结果;  END$  \*/  -- 定义存储函数，获取学生表中成绩大于95分的学生数量  DELIMITER $  CREATE FUNCTION fun\_test1()  RETURNS INT  BEGIN  -- 定义变量  DECLARE s\_count INT;  -- 查询成绩大于95分的数量，为s\_count赋值  SELECT COUNT(\*) INTO s\_count FROM student WHERE score > 95;  -- 返回统计结果  RETURN s\_count;  END$  DELIMITER ; |



### 练习

|  |
| --- |
| /\*  调用函数  SELECT 函数名称(实际参数);  \*/  -- 调用函数  SELECT fun\_test1(); |



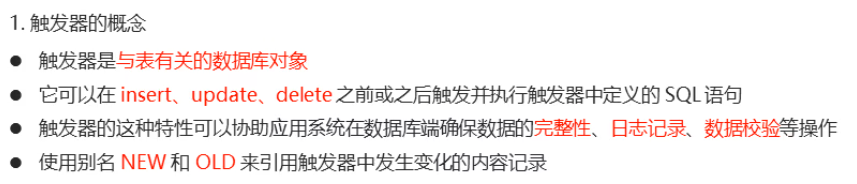
### 练习

|  |
| --- |
| /\*  删除函数  DROP FUNCTION 函数名称;  \*/  -- 删除函数  DROP FUNCTION fun\_test1; |

# 3（掌握）MySQL触发器

## 3.1（掌握）触发器简介





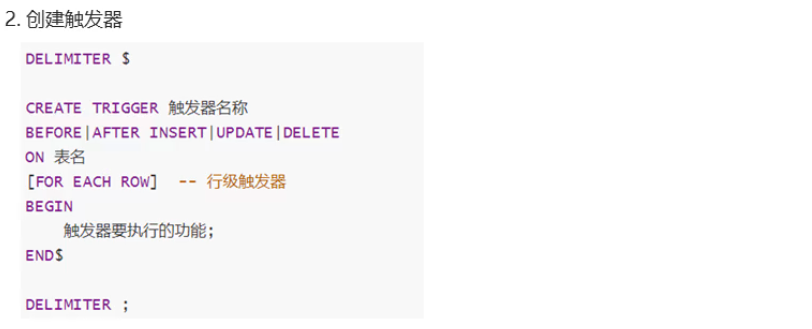


## 3.2（掌握）数据准备

### 数据准备

|  |
| --- |
| -- 创建db7数据库  CREATE DATABASE db7;  -- 使用db7数据库  USE db7;  -- 创建账户表account  CREATE TABLE account(  id INT PRIMARY KEY AUTO\_INCREMENT, -- 账户id  NAME VARCHAR(20), -- 姓名  money DOUBLE -- 余额  );  -- 添加数据  INSERT INTO account VALUES (NULL,'张三',1000),(NULL,'李四',1000);  -- 创建日志表account\_log  CREATE TABLE account\_log(  id INT PRIMARY KEY AUTO\_INCREMENT, -- 日志id  operation VARCHAR(20), -- 操作类型 (insert update delete)  operation\_time DATETIME, -- 操作时间  operation\_id INT, -- 操作表的id  operation\_params VARCHAR(200) -- 操作参数  ); |

## 3.3（掌握）INSERT型触发器



### 练习

|  |
| --- |
| /\*  创建触发器  DELIMITER $  CREATE TRIGGER 触发器名称  BEFORE|AFTER INSERT|UPDATE|DELETE  ON 表名  FOR EACH ROW  BEGIN  触发器要执行的功能;  END$  DELIMITER ;  \*/  -- 创建INSERT型触发器。用于对account表新增数据进行日志的记录  DELIMITER $  CREATE TRIGGER account\_insert  AFTER INSERT  ON account  FOR EACH ROW  BEGIN  INSERT INTO account\_log VALUES (NULL,'INSERT',NOW(),new.id,CONCAT('插入后{id=',new.id,',name=',new.name,',money=',new.money,'}'));  END$  DELIMITER ;  -- 向account表添加一条记录  INSERT INTO account VALUES (NULL,'王五',2000);  -- 查询account表  SELECT \* FROM account;  -- 查询account\_log表  SELECT \* FROM account\_log; |

## 3.4（掌握）UPDATE型触发器

### 练习

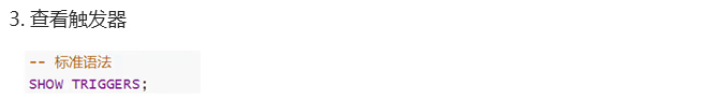
|  |
| --- |
| /\*  创建触发器  DELIMITER $  CREATE TRIGGER 触发器名称  BEFORE|AFTER INSERT|UPDATE|DELETE  ON 表名  FOR EACH ROW  BEGIN  触发器要执行的功能;  END$  DELIMITER ;  \*/  -- 创建UPDATE型触发器。用于对account表修改数据进行日志的记录  DELIMITER $  CREATE TRIGGER account\_update  AFTER UPDATE  ON account  FOR EACH ROW  BEGIN  INSERT INTO account\_log VALUES (NULL,'UPDATE',NOW(),new.id,CONCAT('更新前{id=',old.id,',name=',old.name,',money=',old.money,'}','更新后{id=',new.id,',name=',new.name,',money=',new.money,'}'));  END$  DELIMITER ;  -- 修改account表中李四的金额为2000  UPDATE account SET money=2000 WHERE id=2;  -- 查询account表  SELECT \* FROM account;  -- 查询account\_log表  SELECT \* FROM account\_log; |

## 3.5（掌握）DELETE型触发器

### 练习

|  |
| --- |
| /\*  创建触发器  DELIMITER $  CREATE TRIGGER 触发器名称  BEFORE|AFTER INSERT|UPDATE|DELETE  ON 表名  FOR EACH ROW  BEGIN  触发器要执行的功能;  END$  DELIMITER ;  \*/  -- 创建DELETE型触发器。用于对account表删除数据进行日志的记录  DELIMITER $  CREATE TRIGGER account\_delete  AFTER DELETE  ON account  FOR EACH ROW  BEGIN  INSERT INTO account\_log VALUES (NULL,'DELETE',NOW(),old.id,CONCAT('删除前{id=',old.id,',name=',old.name,',money=',old.money,'}'));  END$  DELIMITER ;  -- 删除account表中王五  DELETE FROM account WHERE id=3;  -- 查询account表  SELECT \* FROM account;  -- 查询account\_log表  SELECT \* FROM account\_log; |

## 3.6（掌握）查看和删除触发器



### 练习

|  |
| --- |
| /\*  查看触发器  SHOW TRIGGERS;  \*/  -- 查看触发器  SHOW TRIGGERS; |



### 练习

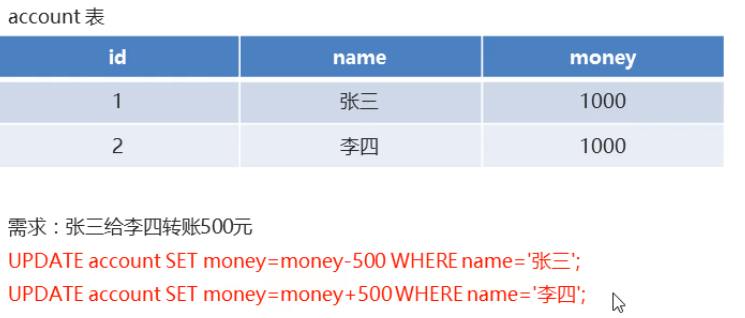
|  |
| --- |
| /\*  删除触发器  DROP TRIGGER 触发器名称;  \*/  -- 删除account\_delete触发器  DROP TRIGGER account\_delete; |

## 3.7（掌握）触发器总结



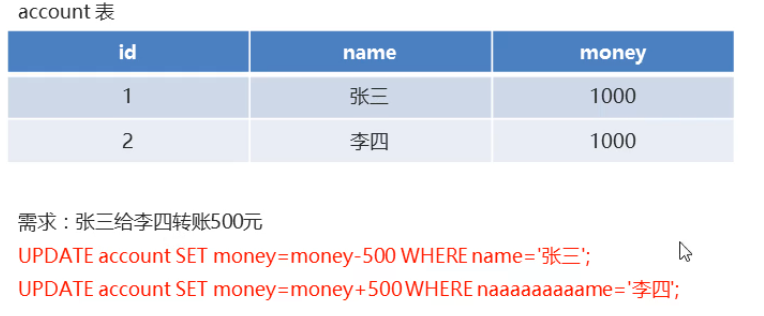
# 4（掌握）MySQL事务

## 4.1（掌握）事务的简介



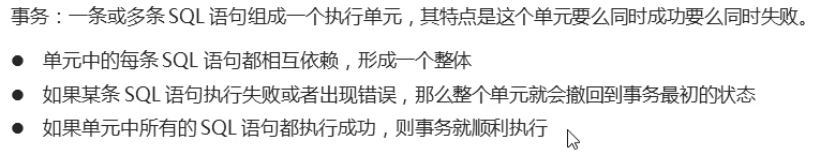
**结果**





**结果**



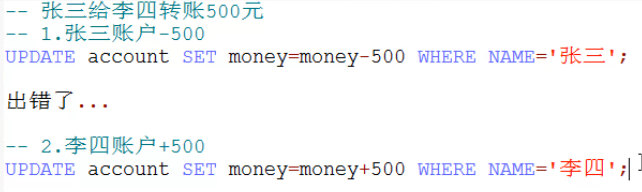


## 4.2（掌握）数据准备

### 数据准备

|  |
| --- |
| -- 创建db8数据库  CREATE DATABASE db8;  -- 使用db8数据库  USE db8;  -- 创建账户表  CREATE TABLE account(  id INT PRIMARY KEY AUTO\_INCREMENT, -- 账户id  NAME VARCHAR(20), -- 账户名称  money DOUBLE -- 账户余额  );  -- 添加数据  INSERT INTO account VALUES (NULL,'张三',1000),(NULL,'李四',1000); |

## 4.3（掌握）未管理事务的场景



### 练习

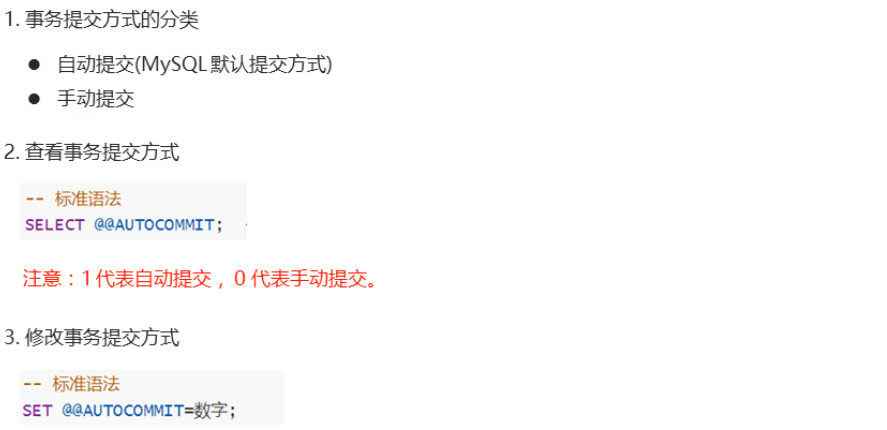
|  |
| --- |
| -- 张三给李四转账500元  -- 1.张三账户-500  UPDATE account SET money=money-500 WHERE NAME='张三';  出错了...  -- 2.李四账户+500  UPDATE account SET money=money+500 WHERE NAME='李四'; |

## 4.4（掌握）管理事务的场景

### 练习

|  |
| --- |
| /\*  开启事务：START TRANSACTION;  回滚事务：ROLLBACK;  提交事务：COMMIT;  \*/  -- 张三给李四转账500元  -- 开启事务  START TRANSACTION;  -- 1.张三账户-500  UPDATE account SET money=money-500 WHERE NAME='张三';  出错了...  -- 2.李四账户+500  UPDATE account SET money=money+500 WHERE NAME='李四';  -- 回滚事务  ROLLBACK;  -- 提交事务  COMMIT; |

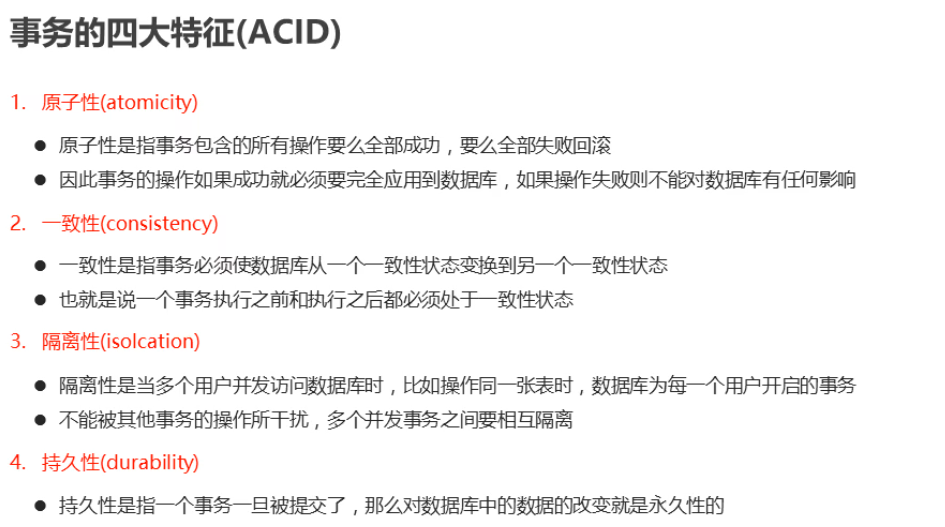
## 4.5（了解）事务的提交方式



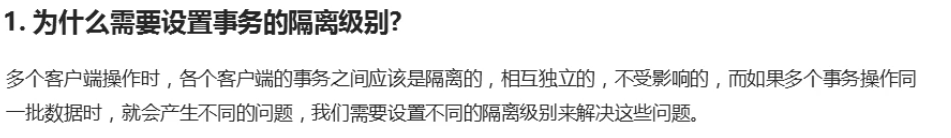
### 练习

|  |
| --- |
| /\*  查询事务提交方式：SELECT @@AUTOCOMMIT; -- 1代表自动提交 0代表手动提交  修改事务提交方式：SET @@AUTOCOMMIT=数字;  \*/  -- 查询事务的提交方式  SELECT @@autocommit;  UPDATE account SET money=2000 WHERE id=1;  COMMIT;  -- 修改事务的提交方式  SET @@autocommit = 1; |

## 4.6（掌握）事务的四大特征

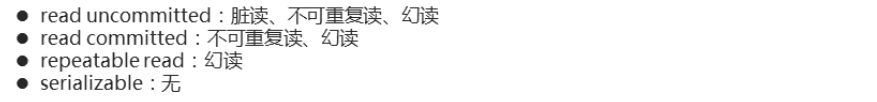


## 4.7（掌握）事务的隔离级别









## 4.8（了解）查询和修改事务的隔离级别



### 练习

|  |
| --- |
| /\*  查询隔离级别：SELECT @@TX\_ISOLATION;  \*/  -- 查询事务隔离级别  SELECT @@tx\_isolation; |



### 练习

|  |
| --- |
| /\*  修改隔离级别：SET GLOBAL TRANSACTION ISOLATION LEVEL 级别字符串;  \*/  -- 修改事务隔离级别(修改后需要重新连接)  SET GLOBAL TRANSACTION ISOLATION LEVEL READ COMMITTED; |

## 4.9（了解）脏读问题的演示和解决

### 练习-窗口1

|  |
| --- |
| /\*  脏读的问题演示和解决  脏读：一个事务中读取到了其他事务未提交的数据  \*/  -- 设置事务隔离级别为read uncommitted  SET GLOBAL TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;  SET GLOBAL TRANSACTION ISOLATION LEVEL READ COMMITTED;  -- 开启事务  START TRANSACTION;  -- 转账  UPDATE account SET money = money-500 WHERE NAME='张三';  UPDATE account SET money = money+500 WHERE NAME='李四';  -- 查询account表  SELECT \* FROM account;  -- 回滚  ROLLBACK;  -- 提交事务  COMMIT; |

### 练习-窗口2

|  |
| --- |
| -- 查询事务隔离级别  SELECT @@tx\_isolation;  -- 开启事务  START TRANSACTION;  -- 查询account表  SELECT \* FROM account;  -- 提交事务  COMMIT; |

## 4.10（了解）不可重复读的演示和解决

### 练习-窗口1

|  |
| --- |
| /\*  不可重复读的问题演示和解决  不可重复读：一个事务中读取到了其他事务已提交的数据  \*/  -- 设置事务隔离级别为read committed  SET GLOBAL TRANSACTION ISOLATION LEVEL READ COMMITTED;  SET GLOBAL TRANSACTION ISOLATION LEVEL REPEATABLE READ;  -- 开启事务  START TRANSACTION;  -- 转账  UPDATE account SET money = money-500 WHERE NAME='张三';  UPDATE account SET money = money+500 WHERE NAME='李四';  -- 查询account表  SELECT \* FROM account;  -- 提交事务  COMMIT; |

### 练习-窗口2

|  |
| --- |
| -- 查询隔离级别  SELECT @@tx\_isolation;  -- 开启事务  START TRANSACTION;  -- 查询account表  SELECT \* FROM account;  -- 提交事务  COMMIT; |

## 4.11（了解）幻读问题的演示和解决

### 练习-窗口1

|  |
| --- |
| /\*  幻读的问题演示和解决  查询某记录是否存在，不存在  准备插入此记录，但执行插入时发现此记录已存在，无法插入  或某记录不存在执行删除，却发现删除成功  \*/  -- 设置隔离级别为repeatable read  SET GLOBAL TRANSACTION ISOLATION LEVEL REPEATABLE READ;  SET GLOBAL TRANSACTION ISOLATION LEVEL SERIALIZABLE;  -- 开启事务  START TRANSACTION;  -- 添加记录  INSERT INTO account VALUES (3,'王五',2000);  INSERT INTO account VALUES (4,'赵六',3000);  -- 查询account表  SELECT \* FROM account;  -- 提交事务  COMMIT; |

### 练习-窗口2

|  |
| --- |
| -- 查询隔离级别  SELECT @@tx\_isolation;  -- 开启事务  START TRANSACTION;  -- 查询account表  SELECT \* FROM account;  -- 添加  INSERT INTO account VALUES (3,'王五',2000);  -- 提交事务  COMMIT; |

## 4.12（了解）隔离级别的总结





## 4.13（掌握）事务的总结

