MySQL 高级

目录 Contents

存储过程和函数介绍

student

id	name	age	gender	score
1	张三	23	男	95
2	李四	24	男	98
3	王五	25	女	100
4	赵六	26	女	90

需求：在 Java 程序中获取 student 表的总成绩的描述信息。

如果分数大于 380 分，则提示学习优秀。

如果分数在 320 到 380 分之间，则提示学习不错。

如果分数小于 320 分，则提示学习一般。

1. 查询 student 表的总成绩。
2. 对总成绩判断，提示对应的信息即可。

MySQL 存储过程和函数

存储过程和函数介绍

- 存储过程和函数是事先经过编译并存储在数据库中的一段 SQL 语句的集合。
- 存储过程和函数的好处

  提高代码的复用性。

  减少数据在数据库和应用服务器之间的传输，提高效率。

  减少代码层面的业务处理。
- 存储过程和函数的区别

  存储函数必须有返回值。

  存储过程可以没有返回值。

MySQL 存储过程和函数

存储过程的创建和调用

- 创建存储过程

```
-- 修改结束分隔符
DELIMITER $

-- 创建存储过程
CREATE PROCEDURE 存储过程名称(参数列表)
BEGIN
                SQL 语句列表;
END$

-- 修改结束分隔符
DELIMITER ;
```

- 调用存储过程

```
CALL 存储过程名称(实际参数);
```

MySQL 存储过程和函数

存储过程的查看和删除

- 查看数据库中所有的存储过程

```
SELECT * FROM mysql.proc WHERE db='数据库名称';
```

- 删除存储过程

```
DROP PROCEDURE [IF EXISTS] 存储过程名称;
```

存储过程语法 - 变量

- 定义变量

```
DECLARE 变量名 数据类型 [DEFAULT 默认值];
```

- 变量赋值方式一

```
SET 变量名 = 变量值;
```

- 变量赋值方式二

```
SELECT 列名 INTO 变量名 FROM 表名 [WHERE 条件];
```

存储过程语法 – if 语句

- if 语句标准语法

```
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;
```

存储过程语法 - 参数传递

- 存储过程的参数和返回值

```
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$
```

IN：代表输入参数，需要由调用者传递实际数据(默认)

OUT：代表输出参数，该参数可以作为返回值

INOUT：代表既可以作为输入参数，也可以作为输出参数

存储过程语法 – while 循环

- while 循环语法

```
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;
```

MySQL 存储过程和函数

存储函数

- 存储函数和存储过程是非常相似的，区别在于存储函数必须有返回值。
- 创建存储函数

```
CREATE FUNCTION 函数名称(参数列表)
RETURNS 返回值类型
BEGIN
        SQL 语句列表;
        RETURN 结果;
END$
```

- 调用存储函数

```
SELECT 函数名称(实际参数);
```

- 删除存储函数

```
DROP FUNCTION 函数名称;
```

目录 Contents

MySQL 触发器

触发器介绍

account 表

id	name	money
1	张三	1000
2	李四	1000

account_log 表

id	operation	operation_time	operation_id	operation_params

MySQL 触发器

触发器介绍

account 表

id	name	money
1	张三	1000
2	李四	1000
3	王五	2000

account_log 表

id	operation	operation_time	operation_id	operation_params

MySQL 触发器

触发器介绍

account 表

id	name	money
1	张三	1000
2	李四	1000
3	王五	2000

account_log 表

id	operation	operation_time	operation_id	operation_params
1	INSERT	2099-09-09 09:09	3	插入后{id=3,name=王五,money=2000}

MySQL 触发器

触发器介绍

account 表

id	name	money
1	张三	1000
2	李四	2000
3	王五	2000

account_log 表

id	operation	operation_time	operation_id	operation_params
1	INSERT	2099-09-09 09:09	3	插入后{id=3,name=王五,money=2000}

MySQL 触发器

触发器介绍

account 表

id	name	money
1	张三	1000
2	李四	2000
3	王五	2000

account_log 表

id	operation	operation_time	operation_id	operation_params
1	INSERT	2099-09-09 09:09	3	插入后{id=3,name=王五,money=2000}
2	UPDATE	2099-10-10 10:10	2	更新前{id=2,name=李四,money=1000} 更新后{id=2,name=李四,money=2000}

MySQL 触发器

触发器介绍

account 表

id	name	money
1	张三	1000
2	李四	2000

account_log 表

id	operation	operation_time	operation_id	operation_params
1	INSERT	2099-09-09 09:09	3	插入后{id=3,name=王五,money=2000}
2	UPDATE	2099-10-10 10:10	2	更新前{id=2,name=李四,money=1000} 更新后{id=2,name=李四,money=2000}

MySQL 触发器

触发器介绍

account 表

id	name	money
1	张三	1000
2	李四	2000

account_log 表

id	operation	operation_time	operation_id	operation_params
1	INSERT	2099-09-09 09:09	3	插入后{id=3,name=王五,money=2000}
2	UPDATE	2099-10-10 10:10	2	更新前{id=2,name=李四,money=1000} 更新后{id=2,name=李四,money=2000}
3	DELETE	2099-11-11 11:11	3	删除前{id=3,name=王五,money=2000}

MySQL 触发器

触发器介绍

- 触发器是与表有关的数据库对象，可以在 insert、update、delete 之前或之后触发并执行触发器中定义的 SQL 语句。
- 这种特性可以协助应用系统在数据库端确保数据的完整性、日志记录、数据校验等操作。
- 使用别名 NEW 和 OLD 来引用触发器中发生变化的内容记录。

- 触发器分类

触发器类型	OLD	NEW
INSERT 型触发器	无 (因为插入前无数据)	NEW表示将要或者已经新增的数据
UPDATE 型触发器	OLD表示修改之前的数据	NEW表示将要或已经修改后的数据
DELETE 型触发器	OLD 表示将要或者已经删除的数据	无 (因为删除后状态无数据)

MySQL 触发器

触发器的操作

- 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 ;
```

MySQL 触发器

触发器的操作

- 查看触发器

```
SHOW TRIGGERS;
```

- 删除触发器

```
DROP TRIGGER 触发器名称;
```

目录 Contents

MySQL 事务

事务介绍

account 表

id	name	money
1	张三	1000
2	李四	1000

需求：张三给李四转账500元

```
UPDATE account SET money=money-500 WHERE name='张三';
UPDATE account SET money=money+500 WHERE name='李四';
```

MySQL 事务

事务介绍

account 表

id	name	money
1	张三	500
2	李四	1500

需求：张三给李四转账500元

```
UPDATE account SET money=money-500 WHERE name='张三';
UPDATE account SET money=money+500 WHERE name='李四';
```

MySQL 事务

事务介绍

account 表

id	name	money
1	张三	1000
2	李四	1000

需求：张三给李四转账500元

```
UPDATE account SET money=money-500 WHERE name='张三';
UPDATE account SET money=money+500 WHERE naaaaaaaaame='李四';
```

MySQL 事务

事务介绍

account 表

id	name	money
1	张三	500
2	李四	1000

需求：张三给李四转账500元

```
UPDATE account SET money=money-500 WHERE name='张三';
UPDATE account SET money=money+500 WHERE naaaaaaaaame='李四';
```

MySQL 事务

事务介绍

account 表

id	name	money
1	张三	1000
2	李四	1000

需求：张三给李四转账500元

```
UPDATE account SET money=money-500 WHERE name='张三';
UPDATE account SET money=money+500 WHERE naaaaaaaaame='李四';
```

MySQL 事务

事务介绍

- 事务：一条或多条 SQL 语句组成一个执行单元，其特点是这个单元要么同时成功要么同时失败。
- 单元中的每条 SQL 语句都相互依赖，形成一个整体。
- 如果某条 SQL 语句执行失败或者出现错误，那么整个单元就会撤回到事务最初的状态。
- 如果单元中所有的 SQL 语句都执行成功，则事务就顺利执行。

MySQL 事务

事务的操作

- 开启事务

```
START TRANSACTION;
```

- 回滚事务

```
ROLLBACK;
```

- 提交事务

```
COMMIT;
```

MySQL 事务

事务的提交方式

- 事务提交方式的分类

  自动提交(MySQL默认)。

  手动提交。

- 查看事务提交方式

  ```
  SELECT @@AUTOCOMMIT;
  ```

- 修改事务提交方式

  ```
  SET @@AUTOCOMMIT=数字;
  ```

事务的四大特征(ACID)

- 原子性(Atomicity)

  原子性是指事务包含的所有操作要么全部成功，要么全部失败回滚。

  因此事务的操作如果成功就必须要完全应用到数据库，如果操作失败则不能对数据库有任何影响。

- 一致性(Consistency)

  一致性是指事务必须使数据库从一个一致性状态变换到另一个一致性状态。

  也就是说一个事务执行之前和执行之后都必须处于一致性状态。

- 隔离性(isolcation)

  隔离性是当多个用户并发访问数据库时，比如操作同一张表时，数据库为每一个用户开启的事务。

  不能被其他事务的操作所干扰，多个并发事务之间要相互隔离。

- 持久性(durability)

  持久性是指一个事务一旦被提交了，那么对数据库中的数据的改变就是永久性的。

  即便是在数据库系统遇到故障的情况下也不会丢失提交事务的操作。

事务的隔离级别

- 事务的隔离级别

  多个客户端操作时，各个客户端的事务之间应该是隔离的，相互独立的，不受影响的。

  而如果多个事务操作同一批数据时，就会产生不同的问题，我们需要设置不同的隔离级别来解决这些问题。

- 隔离级别分类

隔离级别	名称	会引发的问题
read uncommitted	读未提交	脏读、不可重复读、幻读
read committed	读已提交	不可重复读、幻读
repeatable read	可重复读	幻读
serializable	串行化	无

事务的隔离级别

- 引发的问题

问题	现象
脏读	在一个事务处理过程中读取到了另一个未提交事务中的数据，导致两次查询结果不一致
不可重复读	在一个事务处理过程中读取到了另一个事务中修改并已提交的数据，导致两次查询结果不一致
幻读	查询某数据不存在，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。或查询数据不存在执行删除操作，却发现删除成功

事务的隔离级别

- 查询数据库隔离级别

```
SELECT @@TX_ISOLATION;
```

- 修改数据库隔离级别

```
SET GLOBAL TRANSACTION ISOLATION LEVEL 级别字符串;
```

事务的隔离级别

- 隔离级别问题演示
  1. 脏读问题演示。
  2. 不可重复读问题演示。
  3. 幻读问题演示。

事务的隔离级别

- 隔离级别总结

序号	隔离级别	名称	脏读	不可重复读	幻读	数据库默认隔离级别
1	read uncommitted	读未提交	是	是	是	
2	read committed	读已提交	否	是	是	Oracle
3	repeatable read	可重复读	否	否	是	MySQL
4	serializable	串行化	否	否	否	

注意：隔离级别从小到大安全性越来越高，但是效率越来越低，所以不建议修改数据库默认的隔离级别。

传智播客旗下高端IT教育品牌