**Postgresql、MySQL原子性测试报告**

**测试结果：**通过对MySQL 与 PostgreSQL 各六项原子性测试，前 5 项表现一致，仅第六项“死锁”测试出现的位置不同， MySQL 默认采用了“REPEATABLE READ”隔离级别，PostgreSQL 默认为READ COMMITTED（读已提交）隔离级别。**初步结论：两数据库原子性无本质差异。**

**测试环境：**VMbox、CentOS7.9　Postgresql14.9、5.5.68-MariaDB

当编写数据库原子性测试用例时，需要使用SQL语句来模拟不同的操作和情景。以下是一些使用PostgreSQL、MySQL语法的具体SQL示例，用于测试不同操作的原子性：

**1. 插入操作原子性测试：**

​

-- 测试成功插入一条记录

CREATE TABLE users (

  id SERIAL PRIMARY KEY,

  username VARCHAR(50) NOT NULL,

  email VARCHAR(100) NOT NULL

);

​

INSERT INTO users (username, email) VALUES ('testuser', 'test@example.com');

​

​

INSERT 0 1

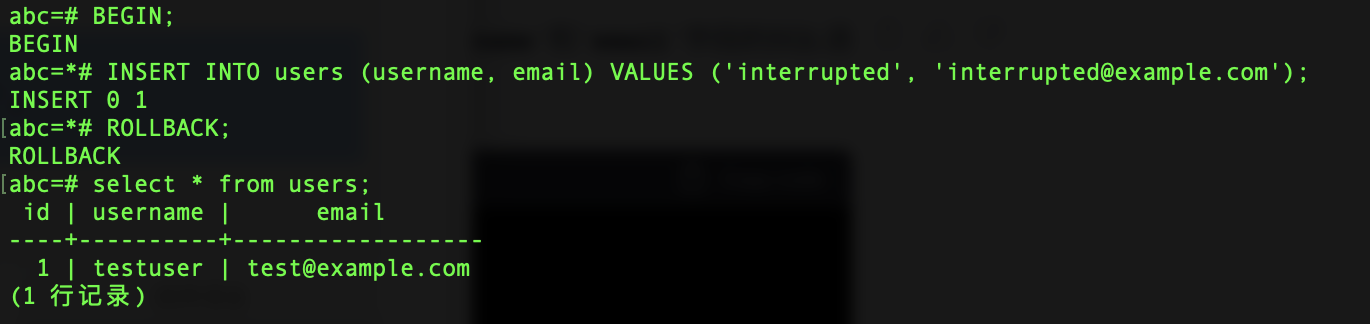
-- 测试插入操作被中断不会插入任何记录

BEGIN;

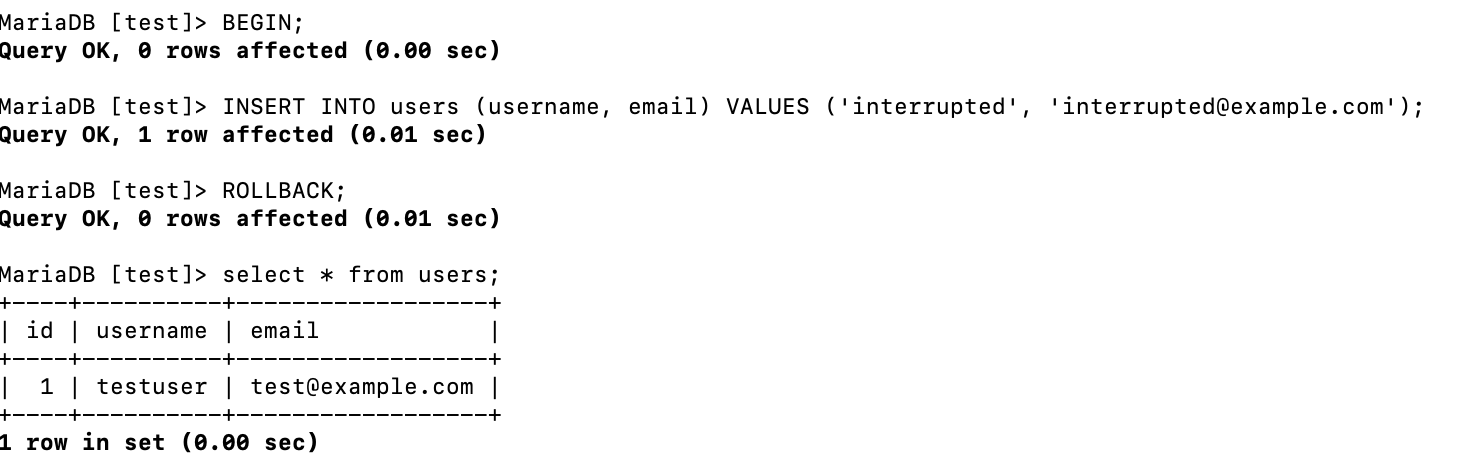
INSERT INTO users (username, email) VALUES ('interrupted', 'interrupted@example.com');

ROLLBACK;

​



Postgresql：成功：没有新纪录出现；



MariaDB：成功：没有新纪录出现；

**2. 更新操作原子性测试：**

​

-- 测试成功更新一条记录

UPDATE users SET email = 'test@example.com' WHERE username ='testuser';

​

-- 测试更新操作被中断不会部分更新记录

BEGIN;

UPDATE users SET email = 'new@example.com' WHERE username ='testuser';

ROLLBACK;



Postgresq：成功，更新操作被中断记录未更新



MariaDB：成功，更新操作被中断记录未更新

**3. 删除操作原子性测试：**

INSERT INTO users (username, email) VALUES ('testuser1', 'test@example.com');

​

-- 测试成功删除一条记录

DELETE FROM users WHERE username ='testuser1';

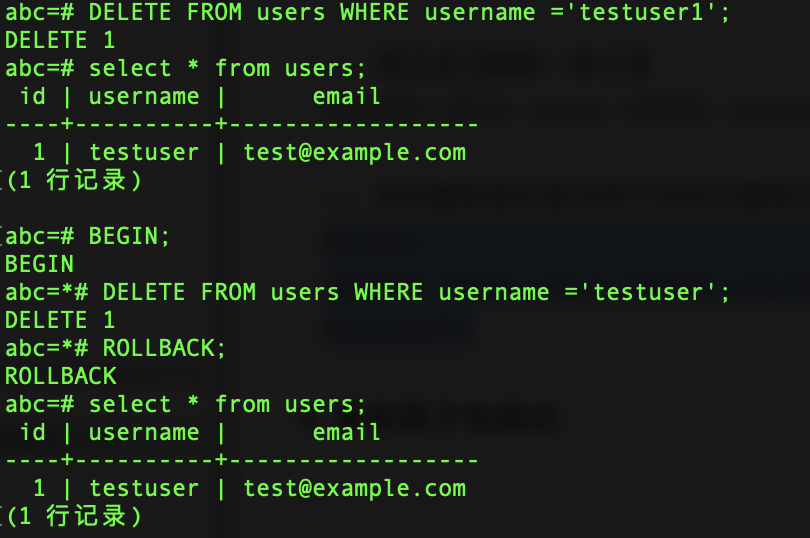
​

-- 测试删除操作被中断不会部分删除记录

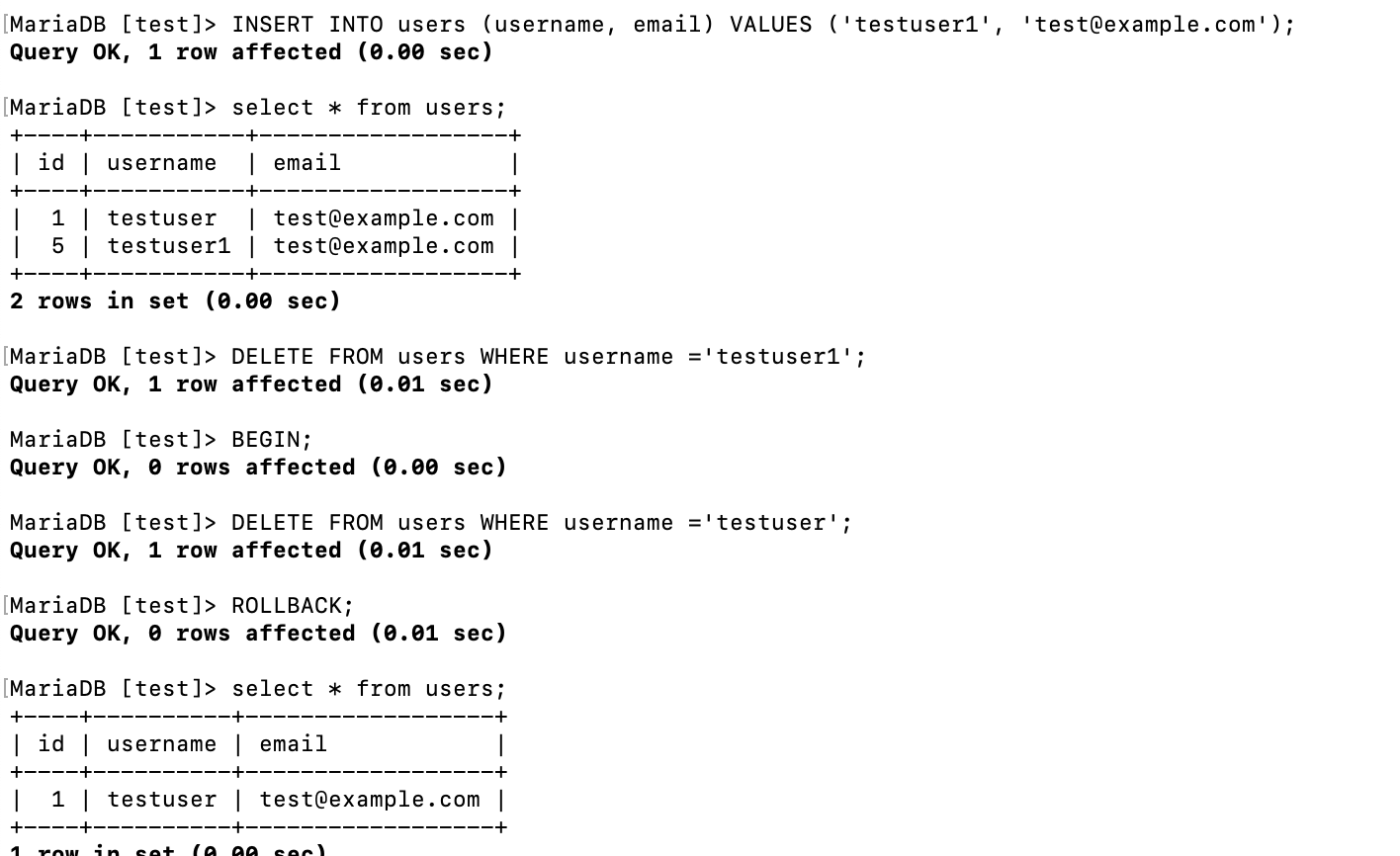
BEGIN;

DELETE FROM users WHERE username ='testuser';

ROLLBACK;



Postgresql成功：删除操作被中断不会部分删除记录



MariaDB成功：删除操作被中断不会部分删除记录

**4. 事务原子性测试：**

​

-- 测试事务中的操作全部成功

BEGIN;

INSERT INTO users (username, email) VALUES ('testuser2', 'test2@example.com');

UPDATE users SET email= 'test2@example.com' WHERE username = 'testuser2';

COMMIT;

​

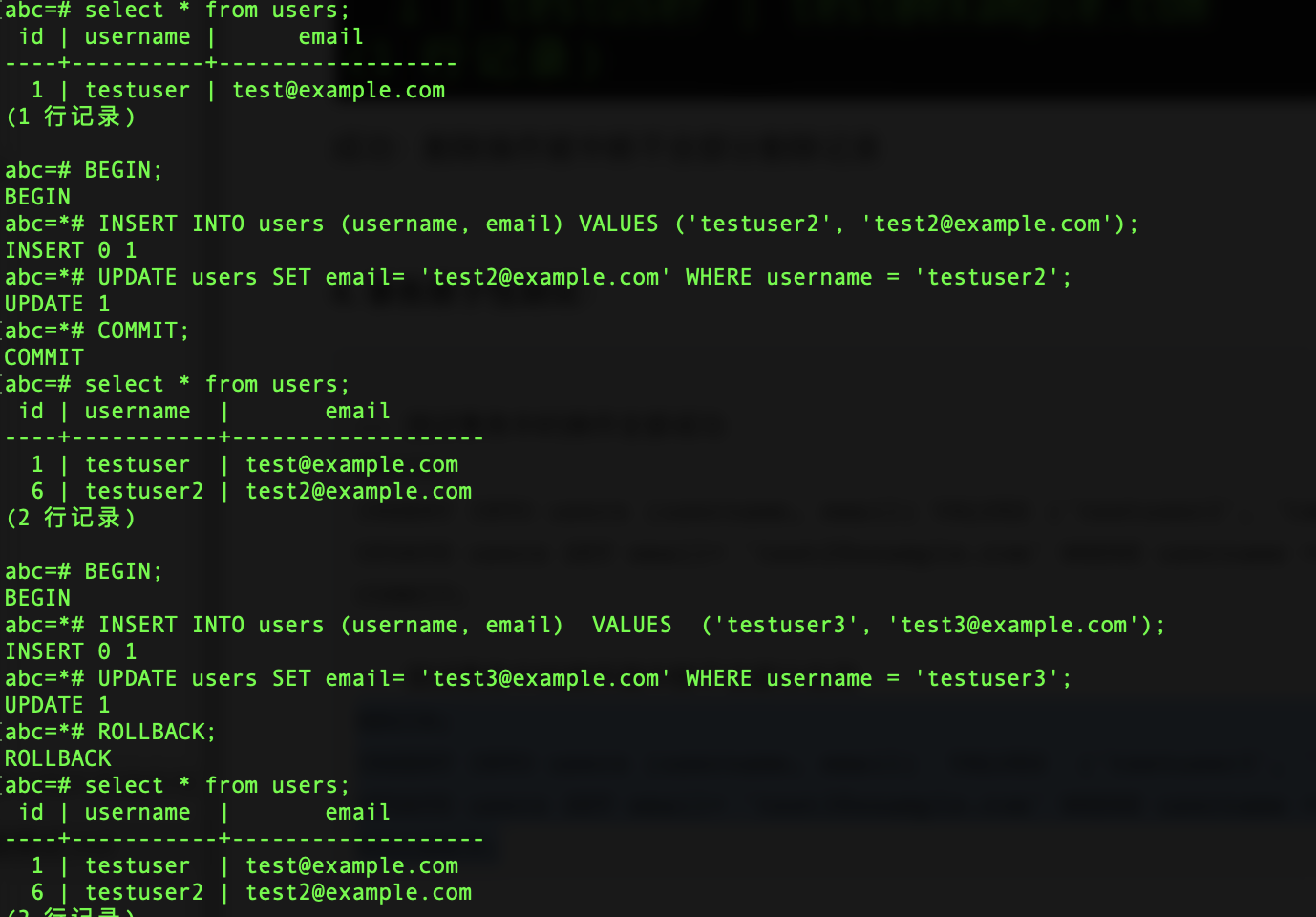
-- 测试事务中的操作被中断不会部分生效

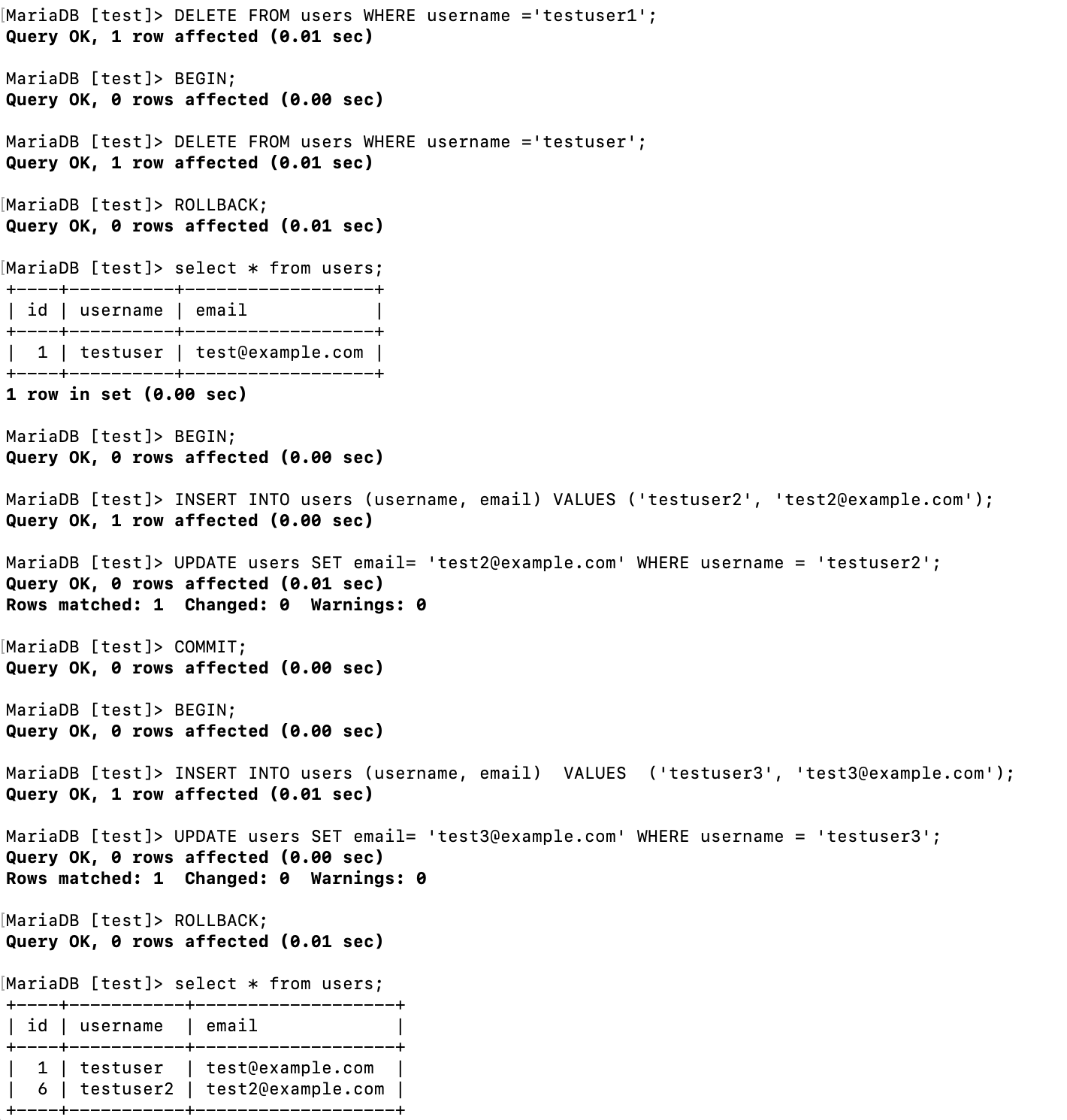
BEGIN;

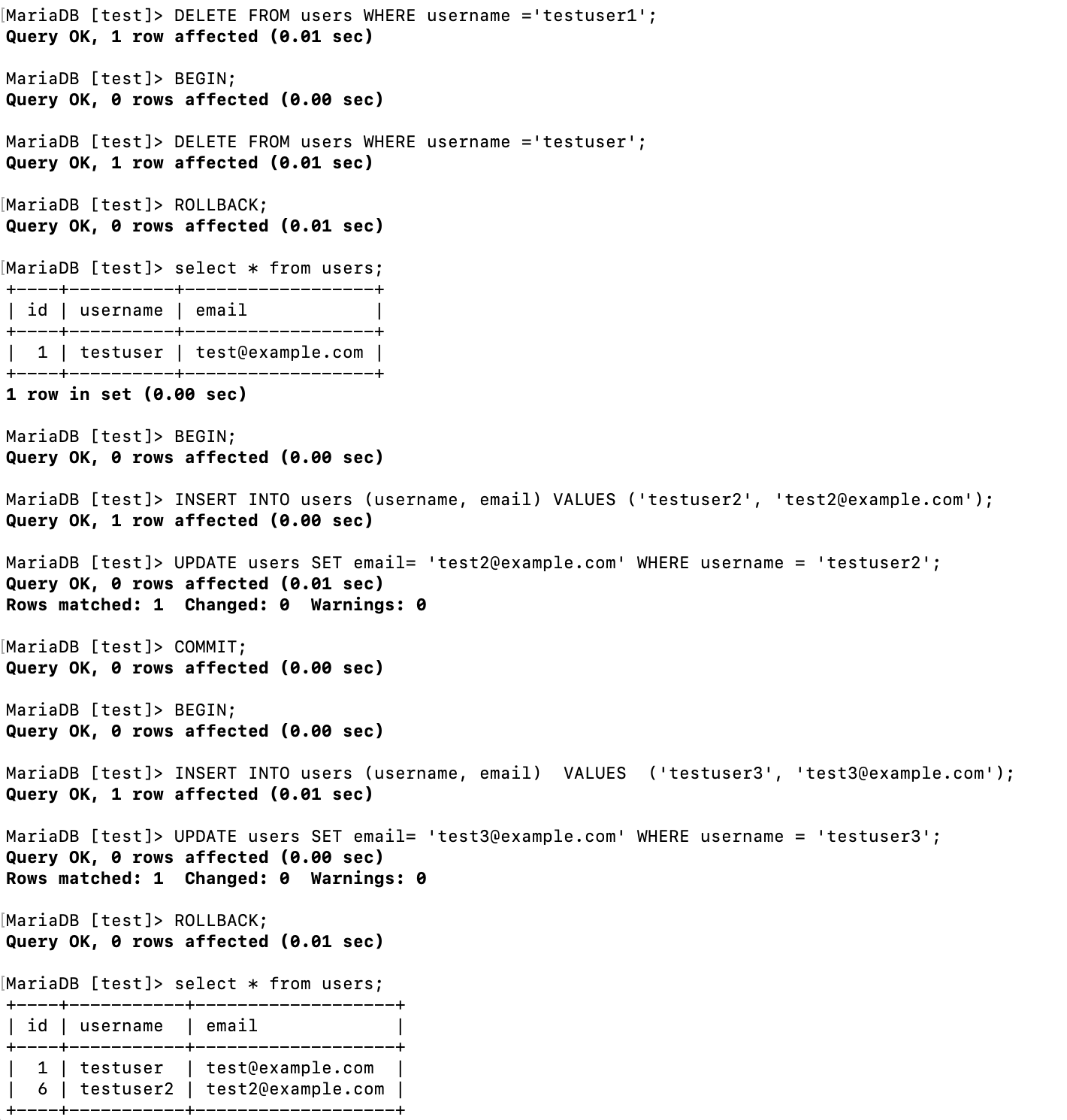
INSERT INTO users (username, email) VALUES ('testuser3', 'test3@example.com');

UPDATE users SET email= 'test3@example.com' WHERE username = 'testuser3';

ROLLBACK;







**5. 并发原子性测试：**

-- 创建多个会话执行插入操作，只有一个能成功

-- 会话1：

BEGIN;

INSERT INTO users (username, email) VALUES ('testuser5', 'test5@example.com');

-- 会话2（同时执行）：

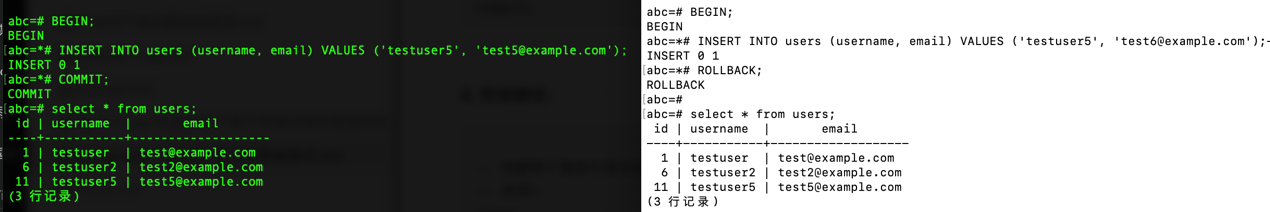
BEGIN;

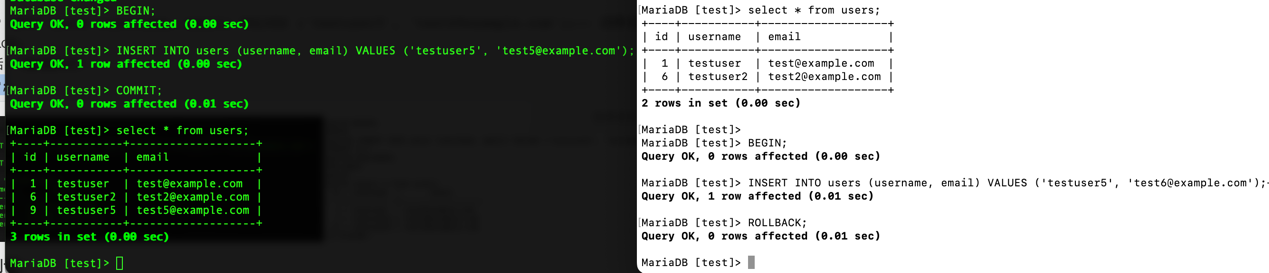
INSERT INTO users (username, email) VALUES ('testuser5', 'test6@example.com');-- 这条会失败

ROLLBACK;

-- 会话1（继续执行）：

COMMIT;

****

****

**6. 死锁测试：**

-- 创建两个事务引发死锁

-- A 会话1：

BEGIN;

UPDATE users SET email = 'test5a@example.com' WHERE username = 'testuser5';

-- B C会话2（同时执行）：

BEGIN;

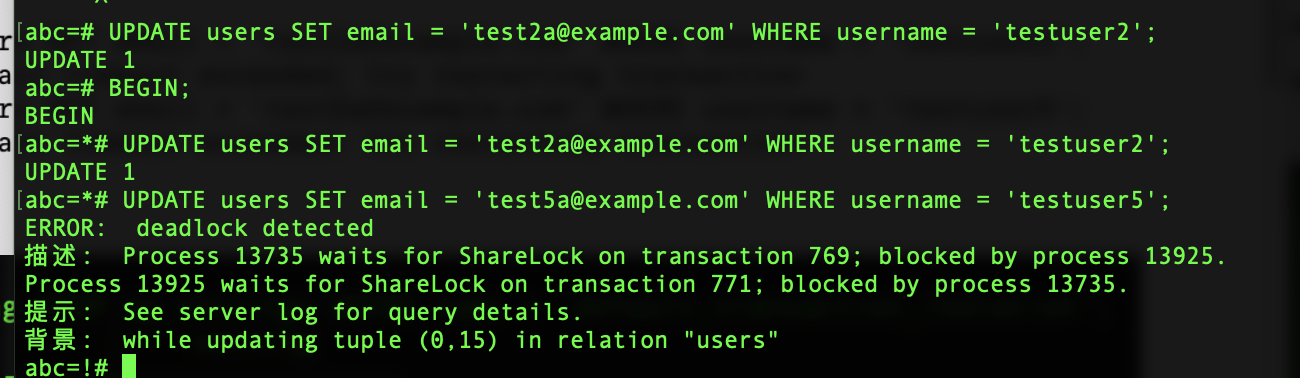
UPDATE users SET email = 'test2a@example.com' WHERE username = 'testuser2';　－mysql 死锁

-- C 会话1（继续执行）：

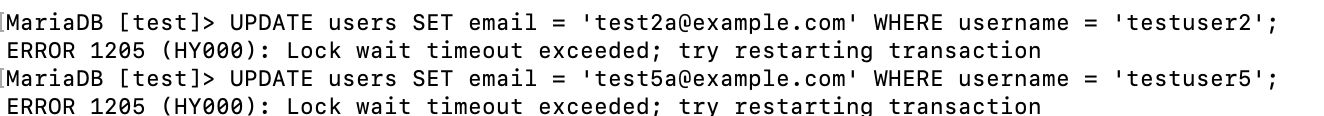
UPDATE users SET email = 'test2a@example.com' WHERE username = 'testuser2'; -- pg死锁

-- D 会话2（继续执行）：

UPDATE users SET email = 'test5a@example.com' WHERE username = 'testuser5'; -- pg、mysql死锁

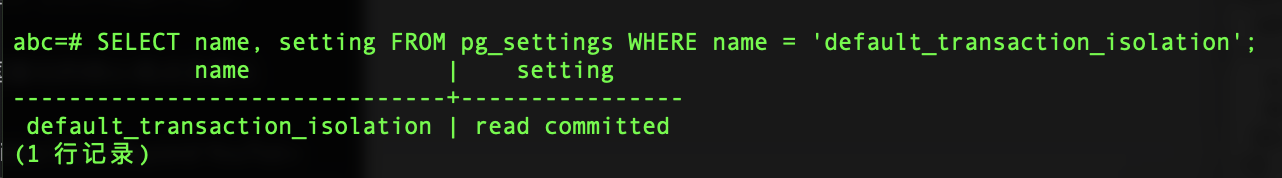


Postgresql成功：C、D 死锁



MariaDB成功：B、D 死锁

**输出PG隔离级别**



**输出mariaDB隔离级别**

