实验报告

一、事务处理

1. isolation 设置为 read uncommited时，脏读的情况

```
#  建立两个连接
-- 连接A
# 1. isolation 设置为 read uncommited时，脏读的情况
## 1) 在一个连接A中，设置transaction isolation 设置为 read-uncommited
SET SESSION TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;
START TRANSACTION;
## 2)连接A中开始事务，显示employees表中所有员工信息，观察记录的数目
SELECT COUNT(*) FROM employees;

-- 连接B
## 3)在另一个连接B中，取消自动提交set @@autocommit=0，在employees表插入一条记录
SET @@autocommit = 0;
INSERT INTO employees VALUES ('999999','测试','博士','1990-01-01',1,5,'测试地
址','12345678','1');

-- 连接A
## 4)在连接A中显示employees表的员工信息，观察记录的数目
SELECT COUNT(*) FROM employees;

-- 连接B
## 5)在连接B中回滚事务
ROLLBACK;

-- 连接A
## 6)在连接A再次显示employees表中所有员工信息，观察记录的数目
SELECT COUNT(*) FROM employees;
## 7)在连接A提交事务
COMMIT;
```

2.观察 @@transaction_isolation 设置为 read-commited时，不可重复读的情况

```
-- 连接A
## 1)在一个连接A中，设置transaction isolation 设置为 read-commited
SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;
START TRANSACTION;
## 2)连接A中开始事务，显示employees表中所有员工信息，观察记录的数目
SELECT COUNT(*) FROM employees;

-- 连接B
## 3)在另一个连接B中，自动提交set @@autocommit=0，在employees表插入一条记录
SET @@autocommit = 0;
```

```
INSERT INTO employees VALUES ('999999','测试','博士','1990-01-01',1,5,'测试地
址','12345678','1');

-- 连接A
## 4)在连接A中显示employees表的员工信息，观察记录的数目
SELECT COUNT(*) FROM employees;

-- 连接B
## 5)在连接B中回滚事务
ROLLBACK;

-- 连接A
## 6)在连接A中显示employees表的员工信息，观察记录的数目
SELECT COUNT(*) FROM employees;

-- 连接B
## 7)在另一个连接B中，在employees表再插入记录，并提交
INSERT INTO employees VALUES ('999999','测试','博士','1990-01-01',1,5,'测试地
址','12345678','1');
COMMIT;

-- 连接A
## 8)在连接A中显示employees表的员工信息，观察记录的数目
SELECT COUNT(*) FROM employees;
## 9)在连接A提交事务
COMMIT;
```

3.观察 transaction_isolation 设置为 repeatable read时，幻读的情况

```
-- 连接A
## 1)在一个连接A中，设置transaction isolation 设置为 repeatable-read
SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;
## 2)开始事务，显示employees表中所有员工信息，观察记录的数目
START TRANSACTION;
SELECT COUNT(*) FROM employees;

-- 连接B
## 3)在另一个连接B中，自动提交set @@autocommit=1，在employees表插入一条记录
SET @@autocommit = 1;
INSERT INTO employees VALUES ('888888','测试2','硕士','1995-05-05',0,2,'新地
址','87654321','2');

-- 连接A
## 4)在连接A中显示employees表的员工信息，观察记录的数目
SELECT COUNT(*) FROM employees;
## 5)在连接A中提交事务
COMMIT;

-- 提交后查看
## 6)在连接A再次显示employees表中所有员工信息，观察记录的数目
SELECT COUNT(*) FROM employees;
```

4.观察 transaction isolation 设置为 serializable时的情况

```
-- 连接A
## 1)在一个连接A中，transaction isolation 设置为 serializable
SET SESSION TRANSACTION ISOLATION LEVEL SERIALIZABLE;
## 2)开始事务，显示employees表中所有员工信息，观察记录的数目
START TRANSACTION;
SELECT COUNT(*) FROM employees;

-- 连接B尝试插入会被阻塞
## 3)在另一个连接B中，在employees表插入一条记录，并提交事务，观察执行情况
-- INSERT INTO employees VALUES ('777777','测试3','本科','2000-01-01',1,1,'地址
3','11223344','3');

-- 连接A
## 4)在连接A中显示employees表的员工信息，观察记录的数目
SELECT COUNT(*) FROM employees;
## 5)在连接A中提交事务；
COMMIT;
## 6)在连接A再次显示employees表中所有员工信息，观察记录的数目
SELECT COUNT(*) FROM employees;
```

二、视图操作

1. 创建视图

```
## (1)在 YGGL 数据库创建视图 ds_view，视图包含 Departments表的全部列
CREATE VIEW ds_view AS
SELECT * FROM Departments;
## (2)在 YGGL 数据库创建视图 Employees_view，视图包含员工号码、姓名和实际收入
CREATE VIEW Employees_view AS
SELECT e.EmployeeID, e.Name, (s.InCome - s.OutCome) AS ActualIncome
FROM Employees e JOIN Salary s ON e.EmployeeID = s.EmployeeID;
```

2. 查询视图

```
# 查询视图
## (1)从视图 ds_view 中查询出部门号为 3 的部门名称
SELECT DepartmentName FROM ds_view WHERE DepartmentID = '3';
## (2)从视图 Employees_view 中查询出姓名为“王林”的员工的实际收入
SELECT ActualIncome FROM Employees_view WHERE Name = '王林';
```