```
#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;
#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；
start transaction;
select*from employees;
#记录数目为 12

#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 value('111111','张三','大专','1972-6-23',1,8,'中山路35-1-508','81111118','9');

#4）在连接A中显示employees表的员工信息，观察记录的数目；
select*from employees;
#记录数目为13

#5）在连接B中回滚事务；
rollback;

#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*from employees;
#记录数目为 12
#7）在连接A提交事务。
commit;
```

结论：islocation 设置为 read uncommitted 时，链接 A 可以读取链接 B 中未提交的记录

```
#2.观察 @@transaction_isolation 设置为 read-commited时，不可重复读的情况。
#1）在一个连接A中，设置transaction isolation 设置为 read-commited;
set session transaction isolation level read committed;
#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；
start transaction;
select*from employees;
#记录数目为 12

#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 value('111111','张三','大专','1972-6-23',1,8,'中山路35-1-508','81111118','9');
#5）在连接B中回滚事务。

#4）在连接A中显示employees表的员工信息，观察记录的数目；
select*from employees;
#记录数目为12
```

```
#5）在连接B中回滚事务；
rollback;

#6）在连接A中显示employees表的员工信息，观察记录的数目；
select*from employees;
#记录数目为12

#7）在另一个连接B中，在employees表再插入记录，并提交；
insert into employees value('222222','李四','大专','1988-6-23',1,8,'中山路00-1-508','82222228','9');
commit;

#8）在连接A中显示employees表的员工信息，观察记录的数目；
select*from employees;
#记录数目为13
#9）在连接A提交事务。
commit;
```

结论：islocation 设置为 read committed 时，A 无法读取 B 提交前的信息，可以读取 B 提交后的信息

```
#3.观察 transaction_isolation 设置为 repeatable read时，幻读的情况。
#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*from employees;
#记录数目为 13

#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 value('111111','张三','大专','1972-6-23',1,8,'中山路35-1-508','81111118','9');

#4）在连接A中显示employees表的员工信息，观察记录的数目；
select*from employees;
#记录数目为13
#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*from employees;
#记录数目为14
```

结论：islocation 设置为幻读时，A 提交前无法读取 B 中（无论是否提交）的信息，A 提交后才能读取到 B 中的信息

```
#4.观察 transaction isolation 设置为 serializable时的情况。
#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*from employees;
#记录数目为 14
```

```
20      #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；
21 •    insert into employees value('333333','王五','本科','1944-6-23',1,8,'中山路11-1-508','83333338','9');
22 •    commit;
23 •    select*from employees;
```

Output

Action Output

#	Time	Action	Message
11	17:12:54	insert into employees value('222222','李四','大专','1988-6-23',1,8,'中山路00-1-508','82222228','9')	1 row(s) affected
12	17:12:55	commit	0 row(s) affected
13	17:18:27	set @@autocommit=1	0 row(s) affected
14	17:18:29	insert into employees value('111111','张三','大专','1972-6-23',1,8,'中山路35-1-508','81111118','9')	1 row(s) affected
15	17:22:48	insert into employees value('333333','王五','本科','1944-6-23',1,8,'中山路11-1-508','83333338','9')	Running...

```
#4）在连接A中显示employees表的员工信息，观察记录的数目；
select*from employees;
#记录数目为 14
#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*from employees;
#记录数目为 15
```

结论：A 提交前，B 不能进行操作；A 提交后 B 自动进行。

```
    use yggl;
    /*二、使用命令完成如下操作，将所使用的命令复制到实验报告，
并上交。
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。
（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。
2、查询视图
（1）从视图 ds_view 中查询出部门号为 3 的部门名称。
（2）从视图 Employees_view 中查询出姓名为“王林”的员工的实
际收入。*/
use yggl;
create view ds_view as select*from departments;
create view employees_view as select employees.employeeID,employees.name,(income-outcome) as sincome
from employees join salary on employees.employeeId=salary.employeeID;
select departmentname from ds_view where departmentID='3';
select sincome from employees_view where name='王林';
```