**Homework 4**

2014313303

홍태하

Problem #1-1

Q1. Fill in the blank if the result exists (Result of “select”)

➀ no rows selected (현재 problem1에는 A,B가 들어있는 상태이다.)

➁ no rows selected (현재 problem1에는 A,B,C가 들어있는 상태이다.)

➂ 수정된 data가 출력됨.

DATA

-----------

A

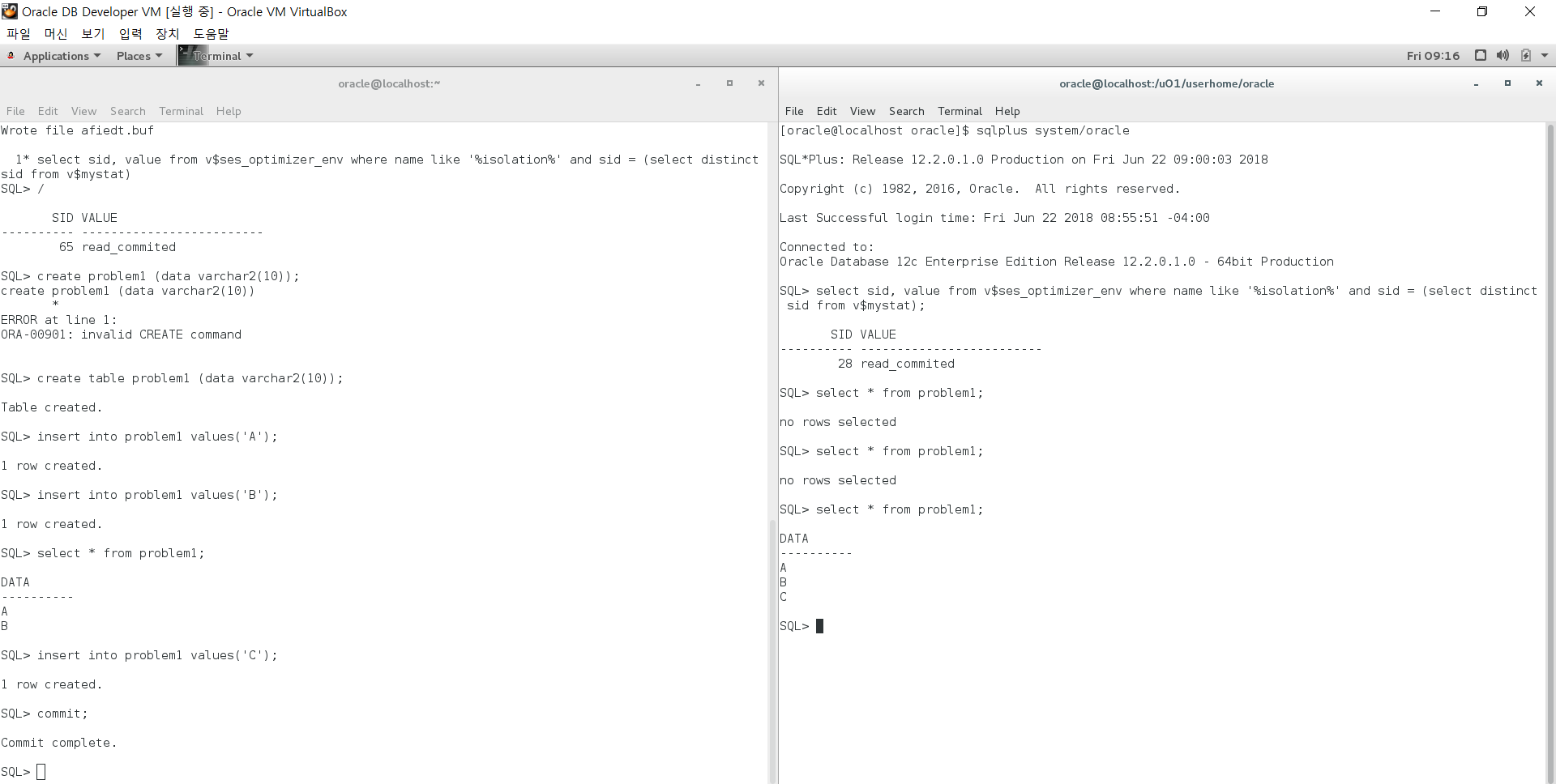
B

C

Q2. What is the problem in this situation?

READ COMMITTED 모드에서는 Session1에서 수정한 결과를 commit하기 전까지 Session2에 반영되지 않는다. 따라서 (4)commit하기 전에 출력된 (1), (2)에서는 아무 data도 출력되지 않았고 commit후에 출력된 (5)번 select문에서는 수정된 data가 출력되었다. 이렇게 되면 Session2에서 모르는 사이에 Session1에서 값을 변경할 수 있기 때문에 의도하지 않은 결과가 출력될 수 있다. (값이 consistent하지 않다.)

Q3. Screen shot



(5)

(4)

(3)

(2)

(1)

Problem #1-2

Q1. Fill in the blank if the result exists (Result of “select”)

➀ Problem #1-1에서 데이터 A,B,C가 삽입되었으므로 A,B,C가 출력된다.

DATA

-----------

A

B

C

➁

DATA

-----------

A

B

C

➂

DATA

-----------

A

B

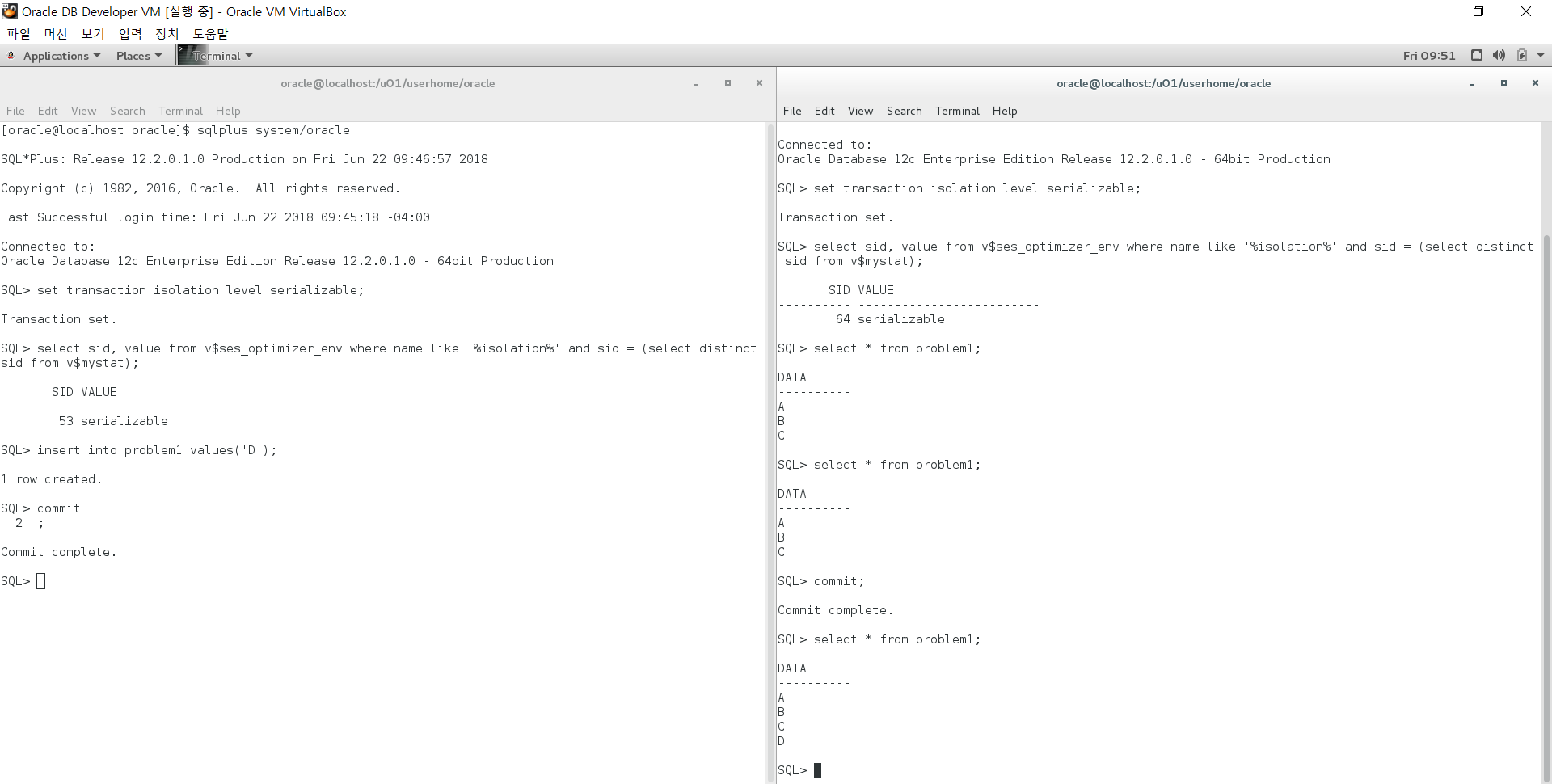
C

D

Q2. What is different with Problem#1-1?

Problem#1-1에서는 READ COMMITTED모드였기 때문에 Session1에서 commit한 후에도 Session1에서 변경한 값이 Session2에 반영되었다. 하지만 SERIALIZABLE 모드를 사용한 Problem#1-2에서는 Session1에서 commit한 후에 Session1에서 변경한 값이 Session2에 반영되지 않았고 Session2에서 commit해준 경우 Session2에서 Session1에서 변경한 값을 출력할 수 있었다.

Q3. Screen shot



(3)

(2)

(6)

(5)

(4)

(1)

Problem #2

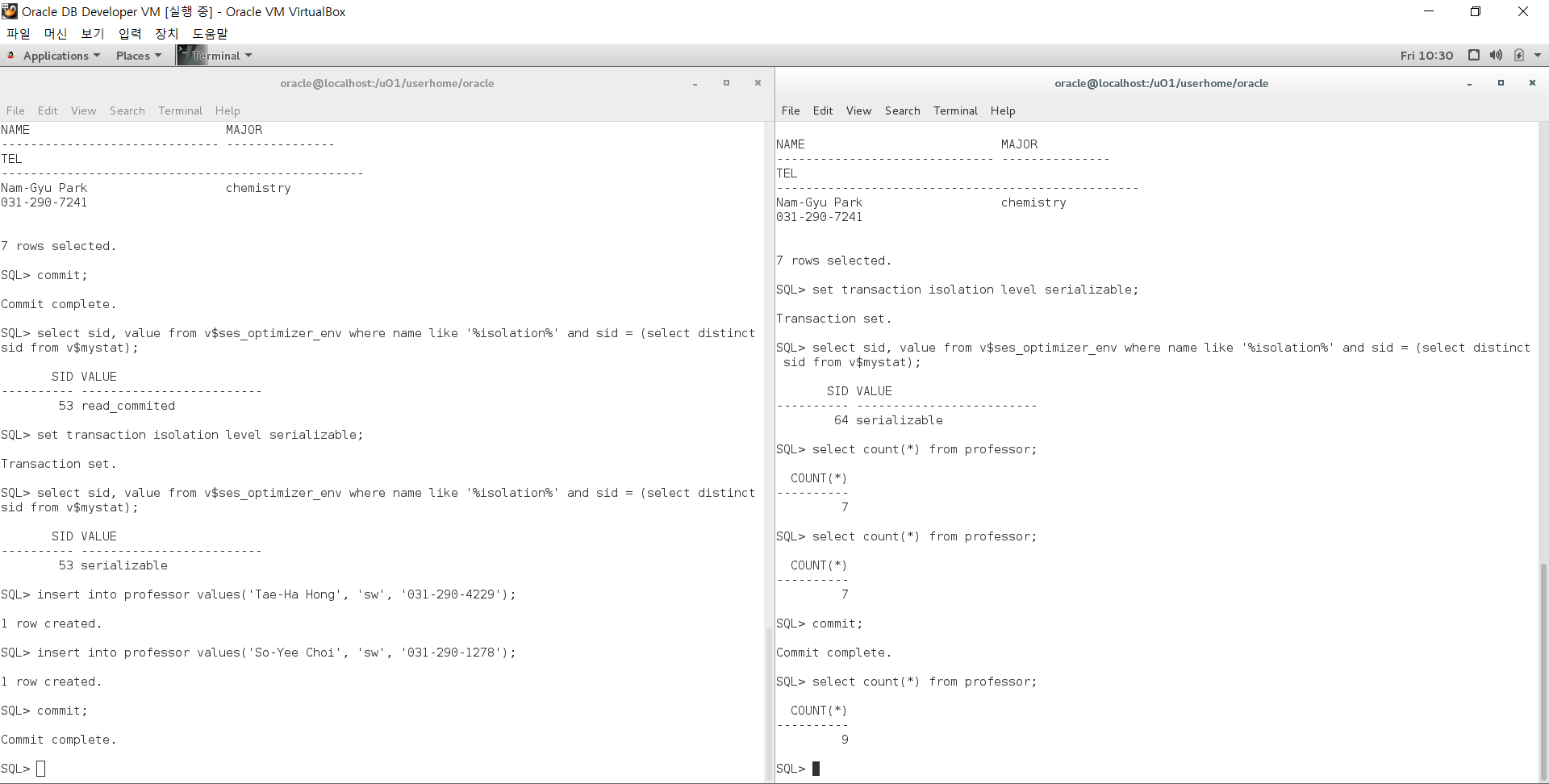
Q1. Define the problem that can arise in this situation.

Officer측에서 commit하기 전에 KingoBot manager가 값을 변경하고 commit해버리면 Officer측에도 변경된 사항이 적용되어서 갑자기 값이 바뀌게 된다.

Q2. What is the solution ?

SERIALIZABLE 모드를 사용하면 Officer측에서 commit하기 전까지 KingoBot manager와 상관없이 consistent한 값을 조회할 수 있다.

Q3. Show the problem and solution in Oracle VM (Screen shot).



(6)

(5)

(3)

(4)

(2)

(1)

먼저 양쪽 모두 SERIALIZABLE 모드를 적용한다. (1)에서 7명의 교수님이 count되었다. KingoBot manager가 (2)에서 2명이 교수를 추가하고 (3) commit하였다. 하지만 Officer측에서는 그와 상관없이 여전히 7명의 교수님이 count된다(4). Officer측에서 (5)commit한 후에야 (6)9명의 교수님이 count된다.

SERIALIZABLE 모드를 이용하여 KingoBot manager와 Officer가 concurrent하게 작동하면서도 consistent한 값을 얻을 수 있다.

Problem #3

1) Describe the table structure & sample data.

create table basketball(seatname varchar2(30), price number, sold number);

insert into basketball values(‘first’, 50000, 200);

insert into basketball values(‘second’, 30000, 350);

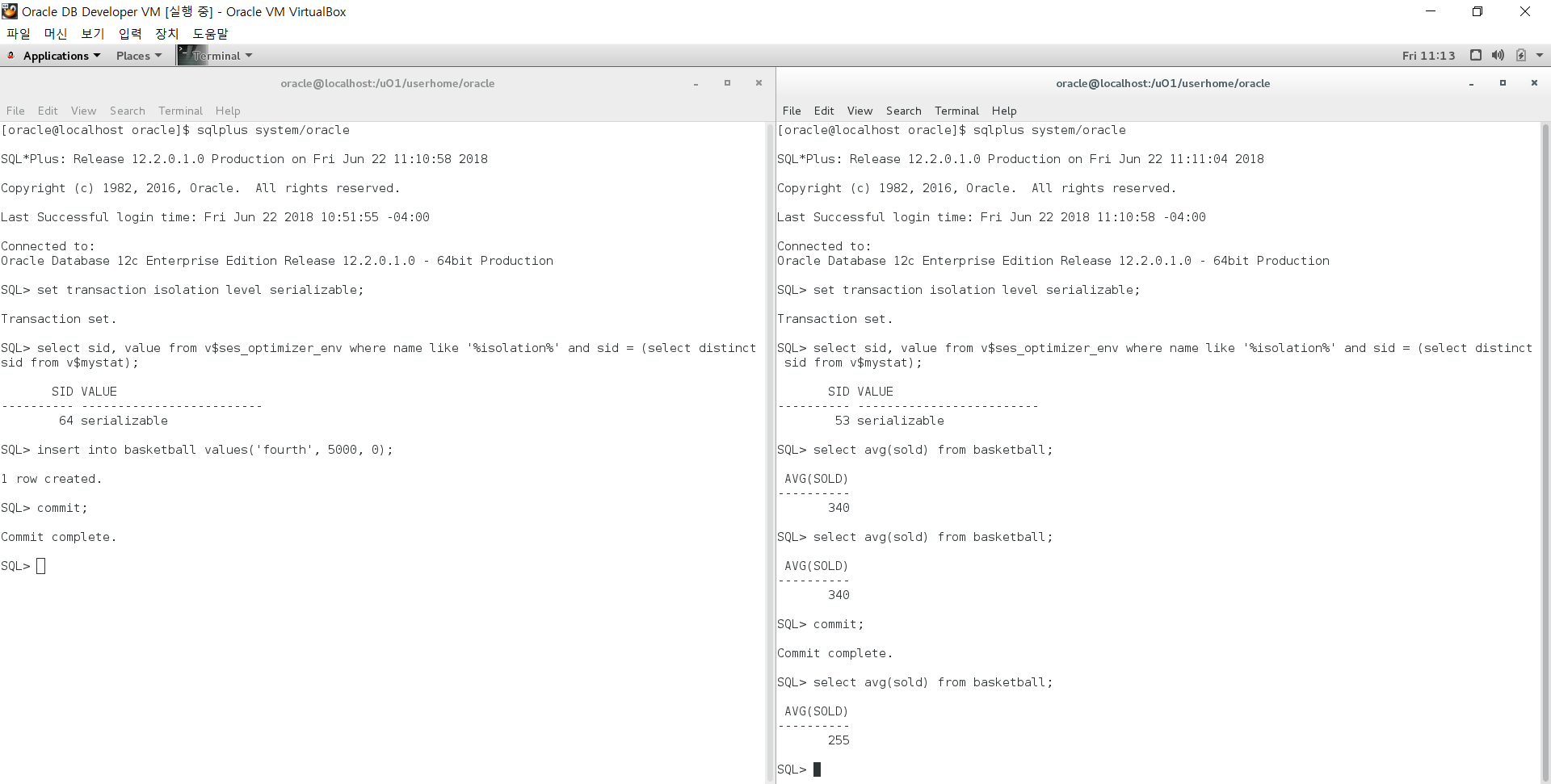
insert into basketball values(‘third’, 10000, 470);

농구 경기 자리 별 가격과 판매된 표의 수를 테이블로 만들었다. 자리는 first, second, third 3가지가 있다.

2) Describe the scenario.

농구 경기 총 관람객 수를 농구 경기 자리의 수로 나누어서 자리별로 평균 인원수가 어떻게 되는지 수시로 체크하고자 한다. 그런데 농구 경기 자리를 하나 더 늘리려고 한다. 만약 농구 경기 자리를 늘리기 위해 테이블에 새로운 tuple을 추가한다면 갑자기 자리 별 평균 인원수가 달라지게 될 것이다. 이것을 방지하기 위해서는 SERIALIZABLE 모드를 사용해야 한다.

3) Screen shot.



(6)

(5)

(4)

(3)

(2)

(1)

(1)에서 자리 별 평균 인원수를 출력하였다. (2)에서 새로운 자리를 테이블에 추가하고 (3)commit하였지만 (4)에서는 값이 변하지 않고 (1)과 같은 값이 출력되었다. (5)에서 commit을 하고 (6)에서 출력을 하자 비로소 변경된 값이 출력되었다. 이렇게 SERIALIZABLE 모드를 활용함으로써 commit전까지 consistent한 값을 얻을 수 있다.