

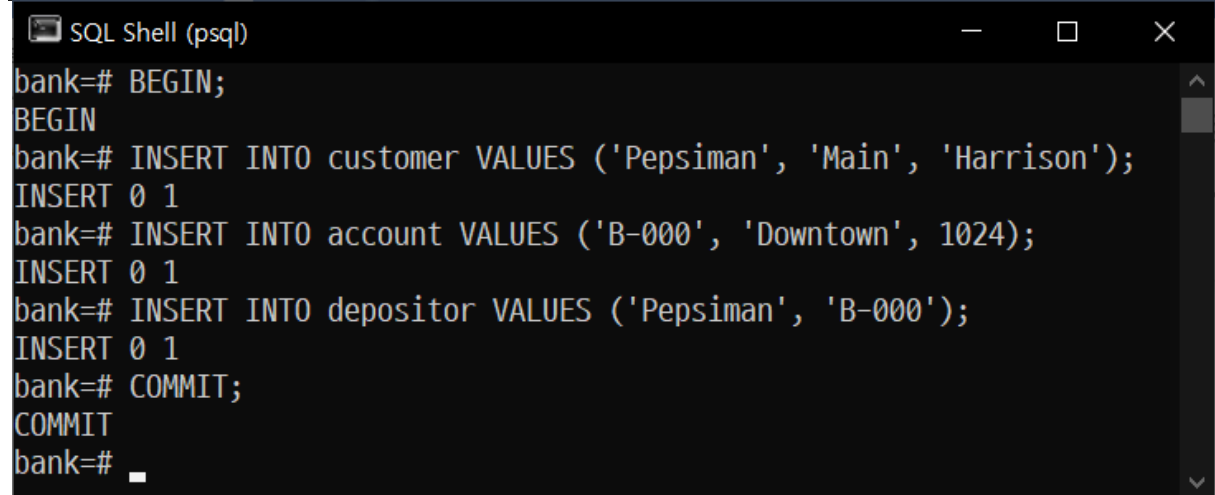
## Database HW7: Practice6

2018320205 신대성

### Exercise 1

1. register a new depositor (with arbitrary values)

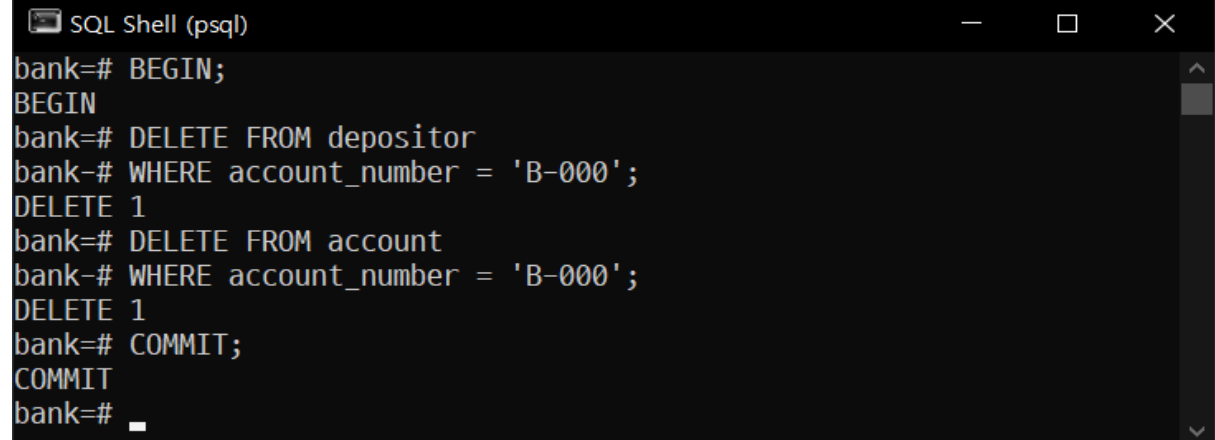
```
BEGIN;
INSERT INTO customer VALUES ('Pepsiman', 'Main', 'Harrison');
INSERT INTO account VALUES ('B-000', 'Downtown', 1024);
INSERT INTO depositor VALUES ('Pepsiman', 'B-000');
COMMIT;
```



The screenshot shows a terminal window titled "SQL Shell (psql)". The prompt is "bank=#". The user enters "BEGIN;". The prompt changes to "BEGIN". The user enters "INSERT INTO customer VALUES ('Pepsiman', 'Main', 'Harrison');". The prompt changes to "INSERT 0 1". The user enters "INSERT INTO account VALUES ('B-000', 'Downtown', 1024);". The prompt changes to "INSERT 0 1". The user enters "INSERT INTO depositor VALUES ('Pepsiman', 'B-000');". The prompt changes to "INSERT 0 1". The user enters "COMMIT;". The prompt changes to "COMMIT". The user enters "bank=#". The prompt changes to "bank=#".

2. eliminate the depositor

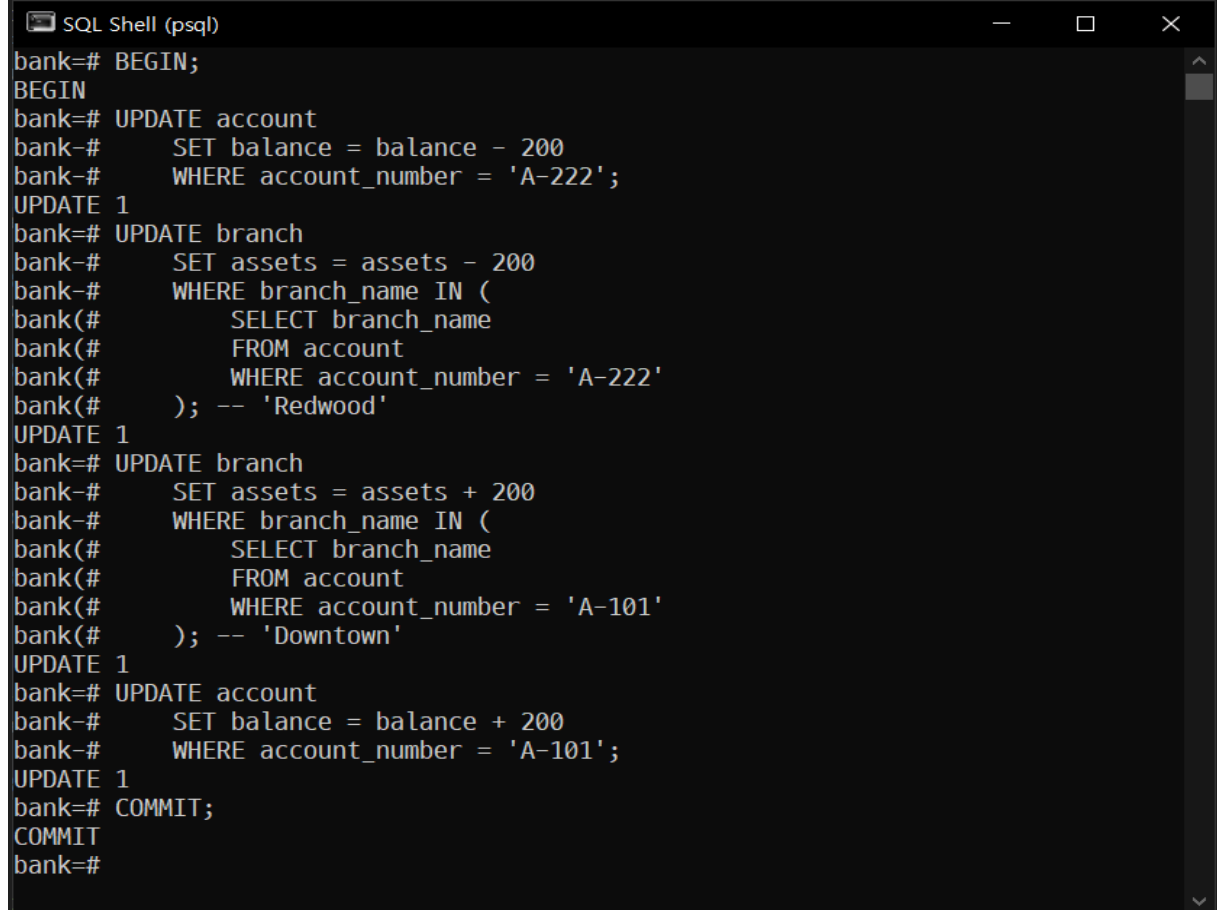
```
BEGIN;
DELETE FROM depositor
WHERE account_number = 'B-000';
DELETE FROM account
WHERE account_number = 'B-000';
COMMIT;
```



The screenshot shows a terminal window titled "SQL Shell (psql)". The prompt is "bank=#". The user enters "BEGIN;". The prompt changes to "BEGIN". The user enters "DELETE FROM depositor". The prompt changes to "DELETE 1". The user enters "WHERE account\_number = 'B-000';". The prompt changes to "DELETE 1". The user enters "DELETE FROM account". The prompt changes to "DELETE 1". The user enters "WHERE account\_number = 'B-000';". The prompt changes to "DELETE 1". The user enters "COMMIT;". The prompt changes to "COMMIT". The user enters "bank=#". The prompt changes to "bank=#".

3. transfer balance 200 from depositor A (at branch 'a') to B (at branch 'b')

```
BEGIN;
UPDATE account
  SET balance = balance - 200
  WHERE account_number = 'A-222';
UPDATE branch
  SET assets = assets - 200
  WHERE branch_name IN (
    SELECT branch_name
    FROM account
    WHERE account_number = 'A-222'
  ); -- 'Redwood'
UPDATE branch
  SET assets = assets + 200
  WHERE branch_name IN (
    SELECT branch_name
    FROM account
    WHERE account_number = 'A-101'
  ); -- 'Downtown'
UPDATE account
  SET balance = balance + 200
  WHERE account_number = 'A-101';
COMMIT;
```

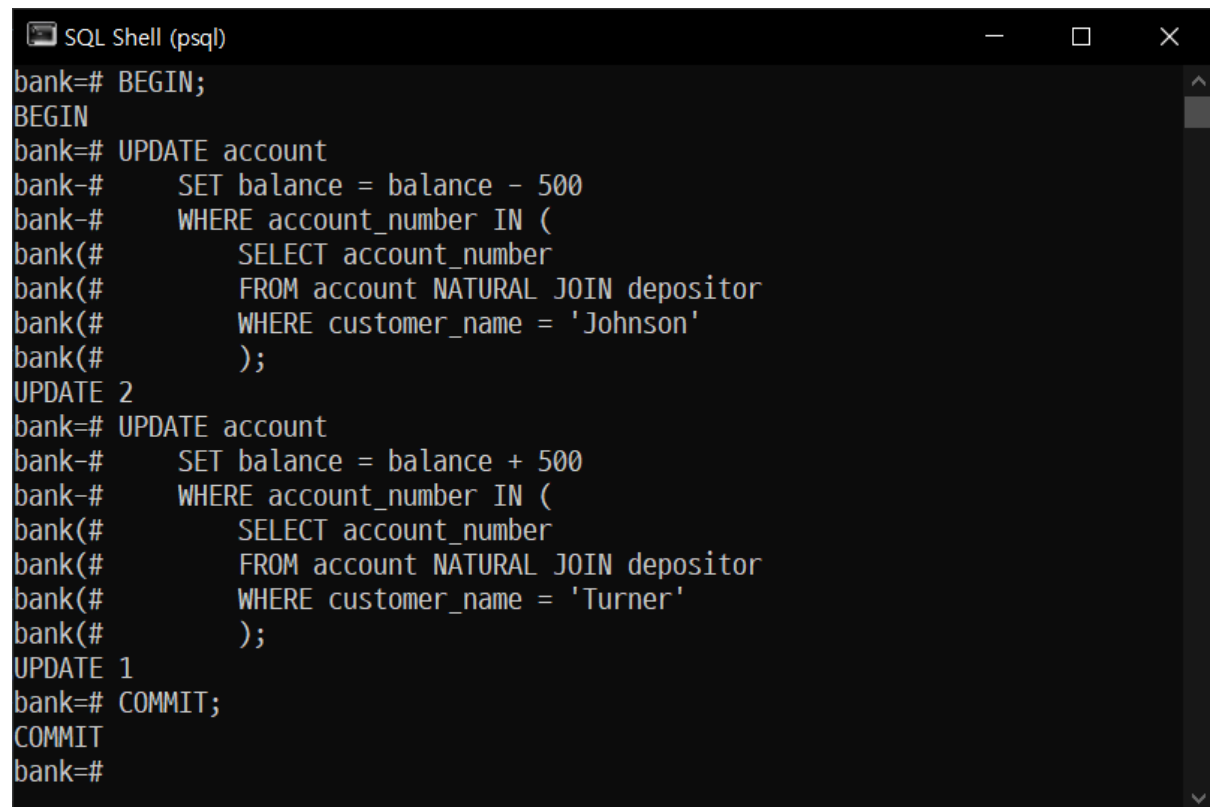


The screenshot shows a terminal window titled "SQL Shell (psql)". The prompt is "bank=#". The user has entered the same SQL script as shown in the previous block. The output shows that the first two UPDATE statements (for account A-222 and branch Redwood) were executed successfully, each returning 1 row. The third UPDATE statement (for branch Downtown) was also executed successfully, returning 1 row. The fourth UPDATE statement (for account A-101) was also executed successfully, returning 1 row. Finally, the COMMIT statement was executed successfully, returning 1 row. The prompt is now "bank=#".

```
SQL Shell (psql)
bank=# BEGIN;
BEGIN
bank=# UPDATE account
      SET balance = balance - 200
      WHERE account_number = 'A-222';
UPDATE 1
bank=# UPDATE branch
      SET assets = assets - 200
      WHERE branch_name IN (
        SELECT branch_name
        FROM account
        WHERE account_number = 'A-222'
      ); -- 'Redwood'
UPDATE 1
bank=# UPDATE branch
      SET assets = assets + 200
      WHERE branch_name IN (
        SELECT branch_name
        FROM account
        WHERE account_number = 'A-101'
      ); -- 'Downtown'
UPDATE 1
bank=# UPDATE account
      SET balance = balance + 200
      WHERE account_number = 'A-101';
UPDATE 1
bank=# COMMIT;
COMMIT
bank=#
```

## Exercise 2

1. Transfer balance 500 from 'Johnson' to 'Turner'



```
SQL Shell (psql)
bank=# BEGIN;
BEGIN
bank=# UPDATE account
bank=#     SET balance = balance - 500
bank=#     WHERE account_number IN (
bank(#         SELECT account_number
bank(#         FROM account NATURAL JOIN depositor
bank(#         WHERE customer_name = 'Johnson'
bank(#     );
UPDATE 2
bank=# UPDATE account
bank=#     SET balance = balance + 500
bank=#     WHERE account_number IN (
bank(#         SELECT account_number
bank(#         FROM account NATURAL JOIN depositor
bank(#         WHERE customer_name = 'Turner'
bank(#     );
UPDATE 1
bank=# COMMIT;
COMMIT
bank=#
```

2. Increase all balance by 500 in the account table, and make a system crash before committing, and crash

```
SQL Shell (psql)
bank=# BEGIN;
BEGIN
bank=# UPDATE account
bank=#     SET balance = balance + 500;
UPDATE 9
bank=# SELECT * FROM account;
 account_number | branch_name | balance
-----+-----+-----
A-215           | Mianus      |    1200
A-102           | Perryridge  |     900
A-217           | Brighton    |    1250
A-333           | Central     |    1350
A-444           | North Town  |    1125
A-222           | Redwood     |    1000
A-201           | Perryridge  |     900
A-101           | Downtown    |     700
A-305           | Round Hill  |    1350
(9개 행)

bank=#
```

이후 psql Shell을 종료해 크래시를 일으켰습니다.

```
SQL Shell (psql)
bank=#
bank=# SELECT * FROM account;
 account_number | branch_name | balance
-----+-----+-----
A-215           | Mianus      |     700
A-102           | Perryridge  |     400
A-217           | Brighton    |     750
A-333           | Central     |     850
A-444           | North Town  |     625
A-222           | Redwood     |     500
A-201           | Perryridge  |     400
A-101           | Downtown    |     200
A-305           | Round Hill  |     850
(9개 행)

bank=#
bank=#
```

## Exercise 3

### 1. Dirty read

```
SQL Shell (psql)
bank=# -- B
bank=# BEGIN;
BEGIN
bank=# UPDATE account
bank=#     SET balance = balance + 1000
bank=#     WHERE account_number = 'A-215';
UPDATE 1
bank=#
```

```
선택 SQL Shell (psql)
bank=# -- A
bank=# BEGIN;
BEGIN
bank=# SELECT * FROM account WHERE account_number = 'A-215';
 account_number | branch_name | balance
-----+-----+-----
A-215          | Mianus      |    700
(1개 행)

bank=# COMMIT;
COMMIT
bank=#
```

Shell B의 커밋이 끝나지 않은 상태에서 다른 Shell A에서 값을 가져와도 Shell B의 변경사항이 적용되지 않았습니다. 이는 psql에서는 어떤 level도 Dirty read를 허용하지 않기 때문입니다.

## 2. Non-repeatable read

```
SQL Shell (psql)
bank=# -- A
bank=# BEGIN;
BEGIN
bank=# SELECT * FROM account WHERE account_number = 'A-201';
 account_number | branch_name | balance
-----+-----+-----
A-201          | Perryridge |    400
(1개 행)

bank=#
```

```
SQL Shell (psql)
bank=# -- B
bank=# BEGIN;
BEGIN
bank=# UPDATE account
bank=#     SET balance = balance + 1000
bank=#     WHERE account_number = 'A-201';
UPDATE 1
bank=# COMMIT;
COMMIT
bank=#
```

```
SQL Shell (psql)
bank=# -- A
bank=# BEGIN;
BEGIN
bank=# SELECT * FROM account WHERE account_number = 'A-201';
  account_number | branch_name | balance
-----+-----+-----
A-201           | Perryridge |    400
(1개 행)

bank=# SELECT * FROM account WHERE account_number = 'A-201';
  account_number | branch_name | balance
-----+-----+-----
A-201           | Perryridge |   1400
(1개 행)

bank=# COMMIT;
COMMIT
bank=#
```

같은 Transaction 안에서 같은 SQL문을 수행했지만, B의 커밋으로 인해 다른 값이 나오게 됩니다. 이는 기본 Isolation level인 Read Committed에서는 Non-repeatable read를 허용하기 때문입니다.

### 3. Phantom read

```
SQL Shell (psql)
bank=# -- A
bank=# BEGIN;
BEGIN
bank=# SELECT * FROM account;
 account_number | branch_name | balance
-----+-----+-----
A-215           | Mianus      |      700
A-102           | Perryridge  |      400
A-217           | Brighton    |      750
A-333           | Central     |      850
A-444           | North Town  |      625
A-222           | Redwood     |      500
A-101           | Downtown    |      200
A-305           | Round Hill  |      850
A-201           | Perryridge  |     1400
(9개 행)
```

bank=#

```
SQL Shell (psql)
bank=# -- B
bank=# BEGIN;
BEGIN
bank=# INSERT INTO account VALUES ('A-999', 'Anam', 1000);
INSERT 0 1
bank=# COMMIT;
COMMIT
bank=#
```



```
SQL Shell (psql)
bank=# SELECT * FROM account;
 account_number | branch_name | balance
-----+-----+-----
 A-215          | Mianus      |      700
 A-102          | Perryridge |      400
 A-217          | Brighton   |      750
 A-333          | Central    |      850
 A-444          | North Town |      625
 A-222          | Redwood    |      500
 A-101          | Downtown   |      200
 A-305          | Round Hill |      850
 A-201          | Perryridge |     1400
 A-999          | Anam       |     1000
(10개 행)

bank=#
```

A의 Transaction에서 account의 튜플들을 확인하는 중에, B의 커밋으로 인해 두번째 쿼리에서 값이 추가되었는데, 이는 기본 Isolation level인 Read Committed에서는 Phantom read를 허용하기 때문입니다.

## Exercise 4

### 1. Dirty read

```
SQL Shell (psql)
bank=# -- B
bank=# BEGIN;
BEGIN
bank=# UPDATE account
bank=#     SET balance = balance + 1000
bank=#     WHERE account_number = 'A-215';
UPDATE 1
bank=#
```

```
SQL Shell (psql)
bank=# -- A
bank=# BEGIN;
BEGIN
bank=# SET TRANSACTION isolation level Serializable;
SET
bank=# SELECT * FROM account WHERE account_number = 'A-215';
 account_number | branch_name | balance
-----+-----+-----
A-215           | Mianus      |    700
(1개 행)

bank=# COMMIT;
COMMIT
bank=#
```

B의 커밋이 끝나지 않은 상태에서 A에서 값을 가져올 때, 업데이트 이전의 값을 가져오며, 이는 psql에서 Dirty Read는 막는다는 것을 알 수 있습니다.

## 2. Non-repeatable read

```
SQL Shell (psql)
bank=# -- A
bank=# BEGIN;
BEGIN
bank=# SET TRANSACTION isolation level Serializable;
SET
bank=# SELECT * FROM account WHERE account_number = 'A-201';
 account_number | branch_name | balance
-----+-----+-----
A-201          | Perryridge  |    1400
(1개 행)
```

```
SQL Shell (psql)
bank=# -- B
bank=# BEGIN;
BEGIN
bank=# UPDATE account
bank=#     SET balance = balance + 1000
bank=#     WHERE account_number = 'A-201';
UPDATE 1
bank=# COMMIT;
COMMIT
bank=#
```

```
SQL Shell (psql)
bank=# -- A
bank=# BEGIN;
BEGIN
bank=# SET TRANSACTION isolation level Serializable;
SET
bank=# SELECT * FROM account WHERE account_number = 'A-201';
 account_number | branch_name | balance
-----+-----+-----
A-201           | Perryridge  |    1400
(1개 행)

bank=# SELECT * FROM account WHERE account_number = 'A-201';
 account_number | branch_name | balance
-----+-----+-----
A-201           | Perryridge  |    1400
(1개 행)

bank=# COMMIT;
COMMIT
```

B에서 값이 변경되고 커밋되었지만, A에서는 같은 Transaction 안에서 같은 쿼리에 대해서 같은 결과를 보여주고 있습니다. 이는 serializable level에서는 Non-repeatable read를 허용하지 않기 때문입니다.

### 3. Phantom read

```
SQL Shell (psql)
bank=# -- A
bank=# BEGIN;
BEGIN
bank=# SET TRANSACTION isolation level Serializable;
SET
bank=# SELECT * FROM account;
 account_number | branch_name | balance
-----+-----+-----
A-215           | Mianus      |      700
A-102           | Perryridge |      400
A-217           | Brighton   |      750
A-333           | Central    |      850
A-444           | North Town |      625
A-222           | Redwood    |      500
A-101           | Downtown   |      200
A-305           | Round Hill |      850
A-201           | Perryridge |     2400
(9개 행)

bank=#
```

```
SQL Shell (psql)
bank=# -- B
bank=# BEGIN;
BEGIN
bank=# INSERT INTO account VALUES ('A-999', 'Anam', 1000);
INSERT 0 1
bank=# COMMIT;
COMMIT
bank=#
```

```
SQL Shell (psql)

bank=# SELECT * FROM account;
 account_number | branch_name | balance
-----+-----+-----
 A-215          | Mianus      |      700
 A-102          | Perryridge  |      400
 A-217          | Brighton    |      750
 A-333          | Central     |      850
 A-444          | North Town  |      625
 A-222          | Redwood     |      500
 A-101          | Downtown    |      200
 A-305          | Round Hill  |      850
 A-201          | Perryridge  |     2400
(9개 행)

bank=# COMMIT;
COMMIT
bank=#
```

A의 트랜잭션 도중에 B에서 튜플을 추가했지만, A에서는 추가되지 않았습니다. 이는 serializable level에서 Phantom read를 허용하지 않기 때문입니다.