資料庫管理 HW04

B12508026 戴偉璿

November 13, 2025

1. To check if PostgreSQL can avoid dirty read, I design two transactions:

   - Transaction A: Update balance to 999 of account_id 1

```sql
            begin;
            update accounts set balance = 999 where account_id = 1;
            commit;
```

   - Transaction B: Read the record.

```sql
            begin; select * from accounts where account_id = 1; commit;


```

   The execution steps are as follows:

   (a) Transaction A begins.

   (b) Transaction A updates balance to 999 of account_id 1, but does not commit yet.

   (c) Transaction B begins.

   (d) Transaction B reads the record of account_id 1.

   (e) Transaction B gets the old balance (not 999), which means dirty read is avoided.

   (f) Transaction B commits.

   (g) Transaction A commits.

   (h) Transaction B begins.

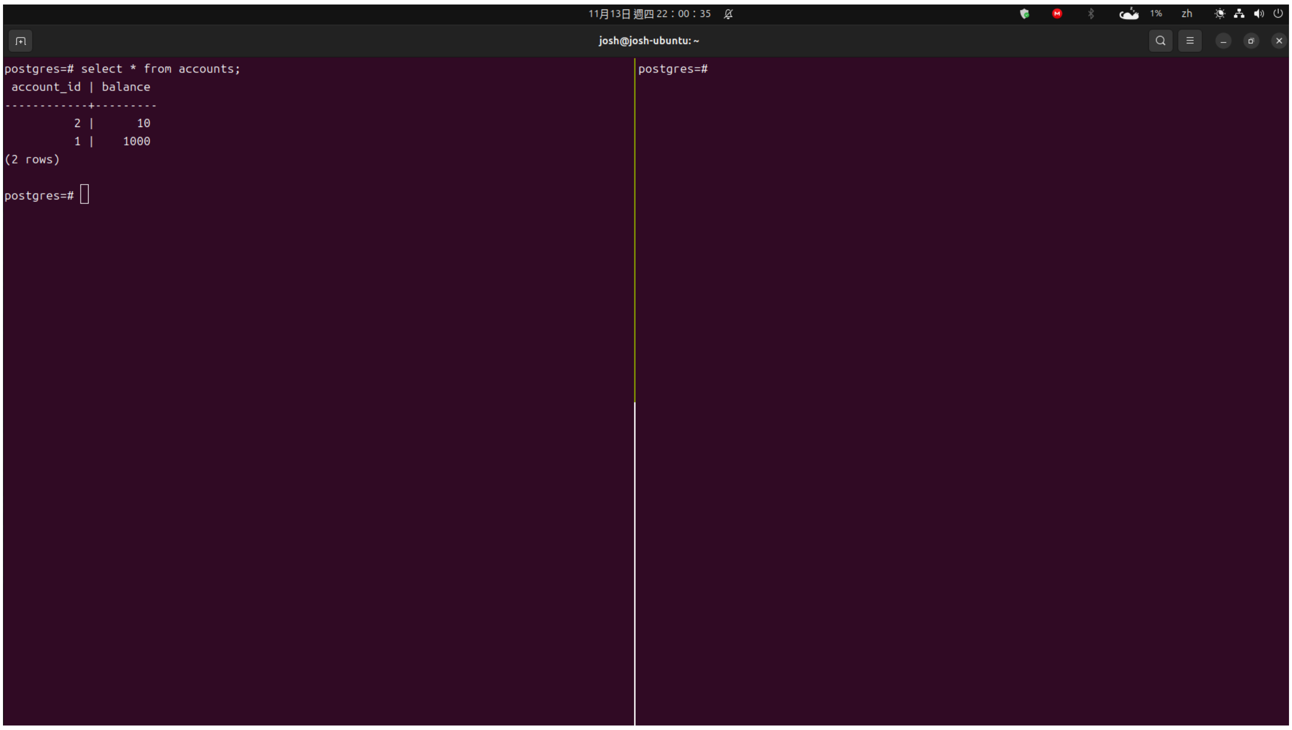
   (i) Transaction B reads the record of account_id 1.

   (j) Transaction B gets the new balance (999) after Transaction A commits.

   (k) Transaction B commits.

   Following are the screenshots of each step. Left panel shows Transaction A, right panel shows Transaction B. Figure 1 shows the original status of the accounts table, we can see the balance of account_id 1 is 1000. Figure 2 shows Transaction A updates balance to

999 of account_id 1 but does not commit yet, so that Transaction B still reads the old balance (1000). Figure 3 shows Transaction A commits, and then Transaction B reads the new balance (999) of account_id 1. You can determine the execution order by the system time shown in the top of each figure.

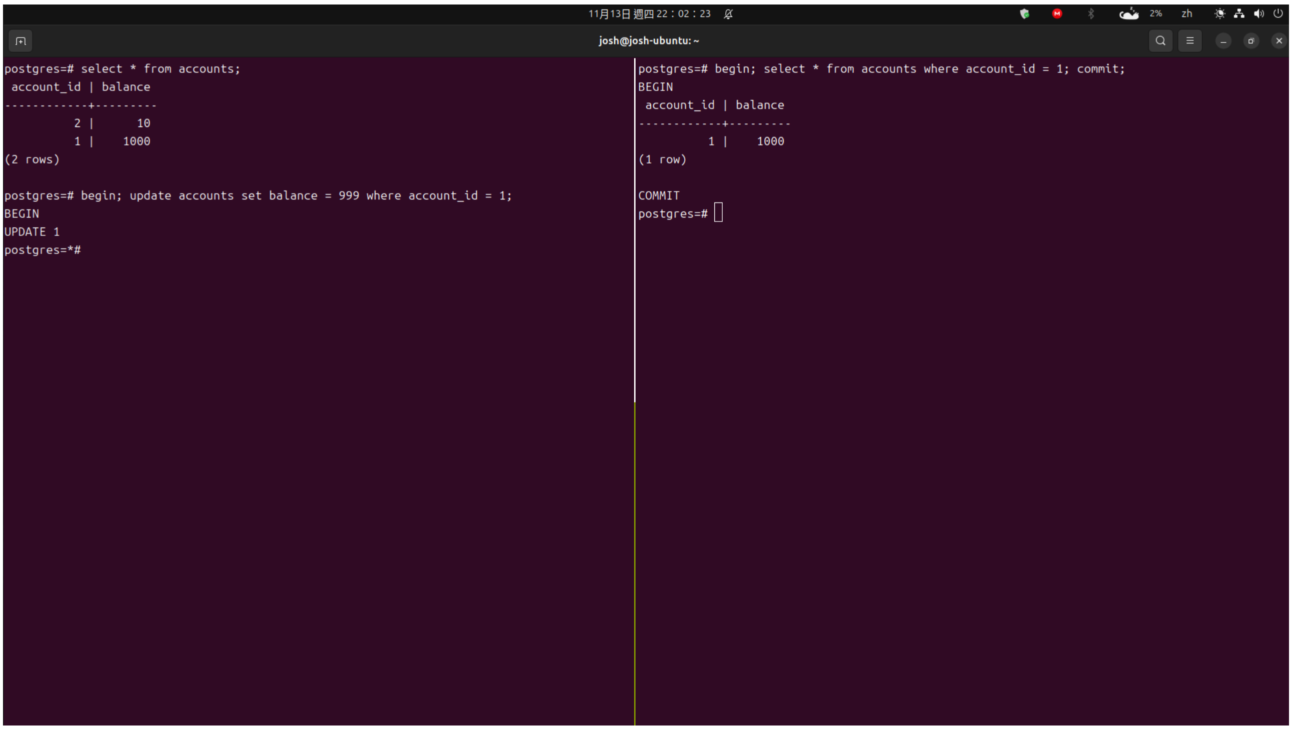


```
postgres=# select * from accounts;
 account_id | balance
------------+---------
          2 |      10
          1 |    1000
(2 rows)

postgres=#
```

```
postgres=#
```

Figure 1: Orginal status of the accounts table



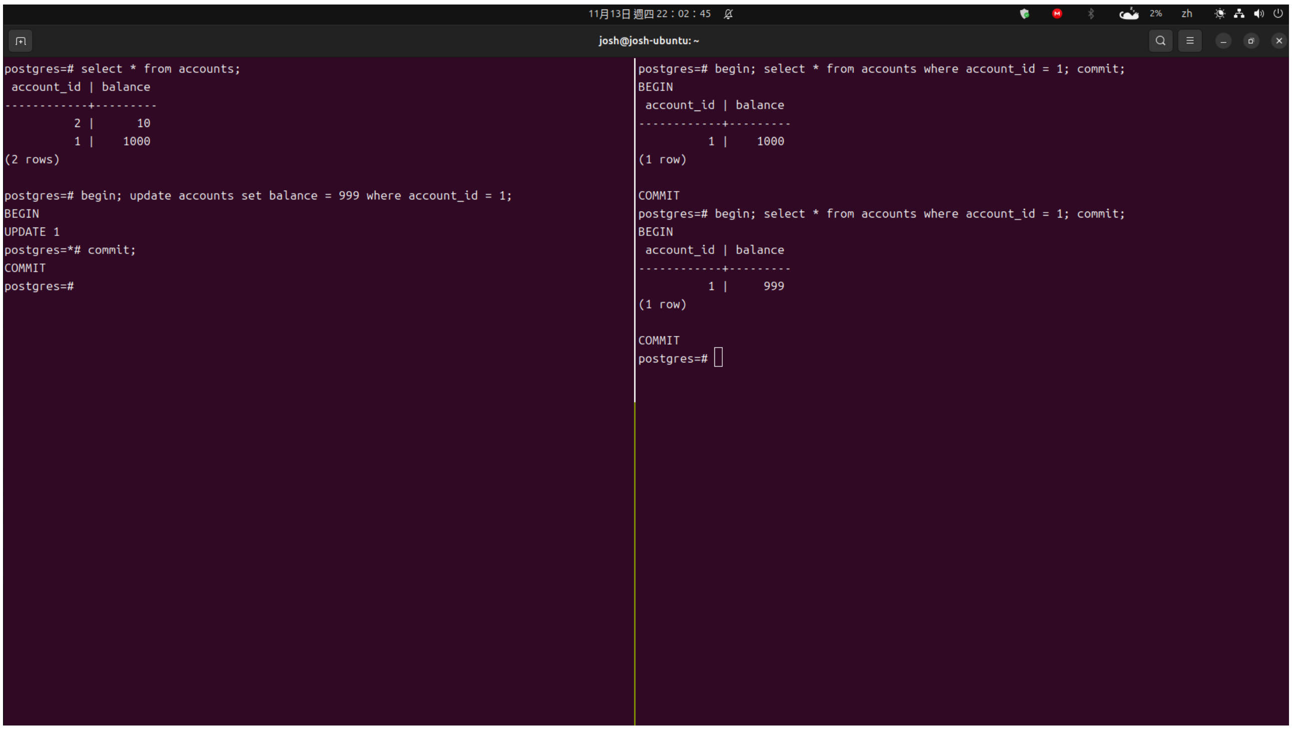
```
postgres=# select * from accounts;
 account_id | balance
------------+---------
          2 |      10
          1 |    1000
(2 rows)

postgres=# begin; update accounts set balance = 999 where account_id = 1;
BEGIN
UPDATE 1
postgres=*#
```

```
postgres=# begin; select * from accounts where account_id = 1; commit;
BEGIN
 account_id | balance
------------+---------
          1 |    1000
(1 row)

COMMIT
postgres=#
```

Figure 2: Transaction A updates balance to 999 of account_id 1, but does not commit yet



```
postgres=# select * from accounts;
 account_id | balance
------------+---------
          2 |      10
          1 |    1000
(2 rows)

postgres=# begin; update accounts set balance = 999 where account_id = 1;
BEGIN
UPDATE 1
postgres=*# commit;
COMMIT
postgres=#
```

```
postgres=# begin; select * from accounts where account_id = 1; commit;
BEGIN
 account_id | balance
------------+---------
          1 |    1000
(1 row)

COMMIT
postgres=# begin; select * from accounts where account_id = 1; commit;
BEGIN
 account_id | balance
------------+---------
          1 |     999
(1 row)

COMMIT
postgres=#
```

Figure 3: Transaction A commits, Transaction B reads the new balance (999) of account_id 1

With the experiments above, we can see that PostgreSQL can avoid dirty read.

2. (a) Only the write