

# 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

---

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

---

- Transaction B: Read the record.

---

```
1      begin; select * from accounts where account_id = 1; commit;
2
3
```

---

The execution steps are as follows:

- Transaction A begins.
- Transaction A updates balance to 999 of account\_id 1, but does not commit yet.
- Transaction B begins.
- Transaction B reads the record of account\_id 1.
- Transaction B gets the old balance (not 999), which means dirty read is avoided.
- Transaction B commits.
- Transaction A commits.
- Transaction B begins.
- Transaction B reads the record of account\_id 1.
- Transaction B gets the new balance (999) after Transaction A commits.
- 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.

```
11月13日 週四 22:00:35 zh
josh@josh-ubuntu:~ postgres=# select * from accounts;
account_id | balance
-----+-----
2 |      10
1 |    1000
(2 rows)
postgres=#[[REDACTED]
```

Figure 1: Orginal status of the accounts table

```
11月13日 週四 22:02:23 zh
josh@josh-ubuntu:~ postgres=# select * from accounts;
account_id | balance
-----+-----
2 |      10
1 |    1000
(2 rows)

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

COMMIT
postgres=#[[REDACTED]
```

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

The screenshot shows two terminal sessions side-by-side. The left session (Transaction A) starts with a query to view all accounts, showing two rows: account\_id 2 with balance 10 and account\_id 1 with balance 1000. It then begins a transaction to update account\_id 1's balance to 999. The right session (Transaction B) starts with a query to select from accounts where account\_id = 1, which returns one row with balance 1000. It then begins its own transaction to update account\_id 1's balance to 999. Finally, Transaction B commits, and its query shows the updated balance of 999 for account\_id 1.

```
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
postres=*=# 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