

POSTGRESQL TRANSACTION

Setting up a sample table

```
dvdr rental/postgres@PostgreSQL 13 v
Query Editor  Query History  Explain  Notifications

1  --DROP TABLE IF EXISTS accounts;
2
3  CREATE TABLE accounts (
4      id INT GENERATED BY DEFAULT AS IDENTITY,
5      name VARCHAR(100) NOT NULL,
6      balance DEC(15,2) NOT NULL,
7      PRIMARY KEY(id)
8  );
9
10 --INSERT INTO TABLE
11
12 INSERT INTO accounts(name,balance)
13 VALUES('Bob',10000);
```

```
dvdr rental/postgres@PostgreSQL 13 v
Query Editor  Query History  Explain  Notifications

1  --TO START TRANSACTION
2  BEGIN TRANSACTION;
```

Messages

```
BEGIN
Query returned successfully in 108 msec.
```

```
dvdr rental/postgres@PostgreSQL 13 v
Query Editor  Query History  Explain  Notifications

1  --Example start a new transaction and insert a new account into the accounts table:
2  BEGIN;
3
4  INSERT INTO accounts(name,balance)
5  VALUES('Alice',10000);
6
```

Messages

```
WARNING: there is already a transaction in progress
INSERT 0 1

Query returned successfully in 322 msec.
```

dvdrental/postgres@PostgreSQL 13

Query Editor
Query History
Explain
Notifications

```

1  --you can see the change by querying the accounts table
2  SELECT
3      id,
4      name,
5      balance
6  FROM
7      accounts;
8

```

Data Output

	id [PK] integer	name character varying (100)	balance numeric (15,2)
1	1	Bob	10000.00
2	2	Alice	10000.00

Commit A Transaction

dvdrental/postgres@PostgreSQL 13

Query Editor
Query History
Explain
Notifications


```

1  --Commit A Transaction
2  COMMIT;
3  --you can view the change by querying the accounts table
4  SELECT
5      id,
6      name,
7      balance
8  FROM
9      accounts;
10

```

Data Output

	id [PK] integer	name character varying (100)	balance numeric (15,2)
1	1	Bob	10000.00
2	2	Alice	10000.00

dvdrental/postgres@PostgreSQL 13 ▾

Query EditorQuery HistoryExplainNotifications


```
1 --After executing the COMMIT statement, PostgreSQL also guarantees that the change will be durable if a crash happens.
2 -- start a transaction
3 BEGIN;
4
5 -- insert a new row into the accounts table
6 INSERT INTO accounts(name,balance)
7 VALUES('Alice',10000);
8
9 -- commit the change (or roll it back later)
10 COMMIT;
```

Messages

COMMIT

Query returned successfully in 543 msec.

PostgreSQL COMMIT: Bank account transfer

dvdrental/postgres@PostgreSQL 13 ▾

Query EditorQuery HistoryExplainNotifications

```
1 -- start a new transaction
2 BEGIN;
3 --and subtracting 1000USD from Bob's account with id 1
4 UPDATE accounts
5 SET balance = balance - 1000
6 WHERE id = 1;
7 --check the account balance of both accounts
8 SELECT
9     id,
10     name,
11     balance
12 FROM
13     accounts;
```

Data Output

	id [PK] integer	name character varying (100)	balance numeric (15,2)
1	2	Alice	10000.00
3	1	Bob	9000.00

dvdrental/postgres@PostgreSQL 13

Query Editor
Query History
Explain
Notifications

```

1  --Next, add the same amount (1000USD ) to Alice's account
2  UPDATE accounts
3  SET balance = balance + 1000
4  WHERE id = 2;
5  --This change also is not visible to the second session until we commit it so we need to commit it
6  COMMIT;
7  --Now, you can view the change from any session by
8  SELECT
9      id,
10     name,
11     balance
12 FROM
13     accounts;

```

Data Output

	id [PK] integer	name character varying (100)	balance numeric (15,2)
2	1	Bob	9000.00
3	2	Alice	11000.00

Put it all together

dvdrental/postgres@PostgreSQL 13

Query Editor
Query History
Explain
Notifications

```

1  -- start a transaction
2  BEGIN;
3
4  -- deduct 1000 from account 1
5  UPDATE accounts
6  SET balance = balance - 1000
7  WHERE id = 1;
8
9  -- add 1000 to account 2
10 UPDATE accounts
11 SET balance = balance + 1000
12 WHERE id = 2;
13
14 -- select the data from accounts
15 SELECT id, name, balance
16 FROM accounts;
17
18 -- commit the transaction
19 COMMIT;
20

```

Messages

```

COMMIT

Query returned successfully in 165 msec.

```

Rolling back a transaction

```
1  --rollback
2  ROLLBACK WORK;
3  --First, add Jack's account to the accounts table:
4  INSERT INTO accounts(name, balance)
5  VALUES('Jack',0);
6  --Next, subtract an amount from Bob's account:
7  BEGIN;
8
9  UPDATE accounts
10 SET balance = balance - 1500
11 WHERE id = 1;
12 --Then, adding the same amount to Alice's account:
13 UPDATE accounts
14 SET balance = balance + 1500
15 WHERE id = 3
```

```
1  --However, Alice's account has id 2. So this was a mistake.
2  --To undo the change, you execute the ROLLBACK statement
3  ROLLBACK;
4  --Finally, check the balances of all accounts:
5  SELECT
6      id,
7      name,
8      balance
9  FROM
10     accounts;
```

id	name	balance
1	Bob	9000
2	Alice	11000
3	Jack	0

```
1  -- begin the transaction
2  BEGIN;
3
4  -- deduct the amount from the account 1
5  UPDATE accounts
6  SET balance = balance - 1500
7  WHERE id = 1;
8
9  -- add the amount from the account 3 (instead of 2)
10 UPDATE accounts
11 SET balance = balance + 1500
12 WHERE id = 3;
13
14 -- roll back the transaction
15 ROLLBACK;
```