

This is our first table, on a first db

QueryQuery History

1

2

3

4

5

6

7

8

9

10

CREATE TABLE recommendations (  
user\_id INT,  
item\_id INT,  
recommendation\_date DATE  
);  
INSERT INTO recommendations  
SELECT generate\_series(1, 100000), (random()\*10000)::int, CURRENT\_DATE - 1;  
SELECT \* FROM recommendations;

Data OutputMessagesNotifications

SQL

user\_idinteger

item\_idinteger

recommendation\_date

1

2

3

4

1

2

3

4

6088

16

885

5380

2025-03-22

2025-03-22

2025-03-22

2025-03-22

Our second table on second db

ObjectToolsEditViewWindowHelp

Welcome

postgres/postgres...

source\_db/postgre...

target\_db/postgres@PostgreSQL 16\*

target\_db/postgres@PostgreSQL 16

No limit

QueryQuery History

1

2

3

4

5

6

7

CREATE TABLE recommendations (  
user\_id INT,  
item\_id INT,  
recommendation\_date DATE  
) PARTITION BY RANGE (recommendation\_date);  
CREATE EXTENSION dblink;

Create a partition for yesterday as we usually run our code at 0-1 a.m.

elcome postgres/postgres... x source\_db/postgre... x target\_db/postgres@PostgreSQL 16\* x

```
target_db/postgres@PostgreSQL 16
Query History

CREATE OR REPLACE FUNCTION create_daily_partition()
RETURNS void AS $$
DECLARE
    y DATE := CURRENT_DATE - 1;
    t DATE := CURRENT_DATE;
    part_name TEXT := 'recommendations_' || TO_CHAR(y, 'YYYY-MM-DD');
BEGIN
    EXECUTE format(
        'CREATE TABLE IF NOT EXISTS %I PARTITION OF recommendations FOR VALUES FROM (%L) TO (%L);',
        part_name, y::text, t::text
    );
END $$ LANGUAGE plpgsql;

SELECT create_daily_partition();
```

Now we move data with dblink

```
target_db/postgres@PostgreSQL 16
Query History

DO $$
DECLARE
    batch_size INT := 10000;
    offset_rows INT := 0;
    rows_fetched INT;
BEGIN
    -- Подключаемся к source_db через dblink
    PERFORM dblink_connect('myconn', 'dbname=source_db user=postgres password=1');

    -- Цикл переноса данных батчами
    LOOP
        EXECUTE format($sql$
            INSERT INTO recommendations (user_id, item_id, recommendation_date)
            SELECT user_id, item_id, recommendation_date
            FROM dblink('myconn', %L)
            AS t(user_id INT, item_id INT, recommendation_date DATE);
        $sql$,
        format(
            'SELECT user_id, item_id, recommendation_date
            FROM recommendations
            WHERE recommendation_date = CURRENT_DATE - INTERVAL ''1 day''
            ORDER BY user_id
            OFFSET %s LIMIT %s',
            offset_rows, batch_size
        ));
        GET DIAGNOSTICS rows_fetched = ROW_COUNT;
        EXIT WHEN rows_fetched < batch_size;
        offset_rows := offset_rows + batch_size;
    END LOOP;

    -- Отключаемся от dblink
    PERFORM dblink_disconnect('myconn');
```

Data Output Messages Notifications

do

Query returned successfully in 1 secs 91 msec.

Now check it

target\_db/postgres@PostgreSQL 16

Query Query History

```
1
2 SELECT * FROM recommendations
3
```

Data Output Messages Notifications

	user_id integer	item_id integer	recommendation_date date
1	1	6088	2025-03-22
2	2	16	2025-03-22
3	3	885	2025-03-22
4	4	5380	2025-03-22
5	5	4926	2025-03-22
6	6	6863	2025-03-22
7	7	1362	2025-03-22
8	8	1823	2025-03-22
9	9	7295	2025-03-22
10	10	4670	2025-03-22
11	11	4737	2025-03-22
12	12	5481	2025-03-22
13	13	4481	2025-03-22
14	14	1170	2025-03-22
15	15	4615	2025-03-22
16	16	4261	2025-03-22
17	17	1530	2025-03-22
18	18	4647	2025-03-22
19	19	4880	2025-03-22
20	20	749	2025-03-22
21	21	4228	2025-03-22
22	22	4967	2025-03-22
23	23	4195	2025-03-22
24	24	6773	2025-03-22
25	25	9515	2025-03-22
26	26	7184	2025-03-22
27	27	337	2025-03-22
28	28	466	2025-03-22

target\_db/postgres@PostgreSQL 16

Query Query History

```
1
2 SELECT COUNT(*) FROM recommendations
3 WHERE recommendation_date = CURRENT_DATE - 1;
```

Data Output Messages Notifications

	count bigint
1	100000

Now for 1m users and 100 items for each

```
source_db/postgres@PostgreSQL 16

Query  Query History

1
2  INSERT INTO recommendations
3  SELECT
4      generate_series(1, 1000000) AS user_id,
5      (random() * 10000)::int AS item_id,
6      CURRENT_DATE - 1 AS recommendation_date
7  FROM generate_series(1, 100);
8

Data Output  Messages  Notifications

INSERT 0 1000000000

Query returned successfully in 3 min 59 secs.
```

target\_db/postgres@PostgreSQL 16

---

Query History

```

14      -- Переносим батчи
15 LOOP
16     BEGIN
17         EXECUTE format($sql$
18             INSERT INTO recommendations (user_id, item_id, recommendation_date)
19             SELECT user_id, item_id, recommendation_date
20             FROM dblink('myconn', %L)
21             AS t(user_id INT, item_id INT, recommendation_date DATE);
22     $sql$,
23     format(
24         'SELECT user_id, item_id, recommendation_date
25          FROM recommendations
26          WHERE recommendation_date = CURRENT_DATE - INTERVAL ''1 day''
27          ORDER BY user_id
28          OFFSET %s LIMIT %s',
29         offset_rows, batch_size
30     ));
31
32     GET DIAGNOSTICS rows_fetched = ROW_COUNT;
33
34     RAISE NOTICE '% Batch % completed. Rows inserted: %', current_batch, rows_fetched;
35
36 EXCEPTION WHEN OTHERS THEN
37     RAISE WARNING '✗ Batch % FAILED at offset %: %', current_batch, offset_rows, SQLERRM;
38     -- OPTIONAL: exit or continue to next batch
39     EXIT;
40 END;
41
42 EXIT WHEN rows_fetched < batch_size;
43 offset_rows := offset_rows + batch_size;
44 current_batch := current_batch + 1;
45 END LOOP;
46
47 PERFORM dblink_disconnect('myconn');
48 END $$;
49

```

Data Output Messages Notifications

ЗАМЕЧАНИЕ:	✔ Batch 12 completed. Rows inserted: 1000000
ЗАМЕЧАНИЕ:	✔ Batch 13 completed. Rows inserted: 1000000
ЗАМЕЧАНИЕ:	✔ Batch 14 completed. Rows inserted: 1000000
ЗАМЕЧАНИЕ:	✔ Batch 15 completed. Rows inserted: 1000000
ЗАМЕЧАНИЕ:	✔ Batch 16 completed. Rows inserted: 1000000
ЗАМЕЧАНИЕ:	✔ Batch 17 completed. Rows inserted: 1000000
Total rows:	Waiting for the query to complete... 00:17:32.123

I decided to batch for 1 million rows, every batch is about a minute so my 100millions rows will be less than 2 hours.

After each batch, I log the batch number and how many rows were inserted