

1.

test_db/postgres@PostgreSQL 17* X

test_db/postgres@PostgreSQL 17

No limit

Query Query History

```
1 SELECT d.oid, d.datname, d.datistemplate, d.dataallowconn, t
2 FROM pg_database d
3 JOIN pg_tablespace t ON t.oid = d.dattablespace;
```

Data Output Messages Notifications

	oid oid	datname name	datistemplate boolean	dataallowconn boolean	spcname name
1	5	postgres	false	true	pg_default
2	1	template1	true	true	pg_default
3	4	template0	true	false	pg_default
4	17463	gym_membership	false	true	pg_default
5	16776	reweb_old	false	true	pg_default
6	18585	reweb	false	true	pg_default
7	20302	AuctionHouseDB	false	true	pg_default
8	20709	sales_history	false	true	pg_default
9	28810	RealEstateDB	false	true	pg_default
10	16852	dvdrental	false	true	pg_default
11	34470	VehiclesSales	false	true	pg_default
12	39126	VehilesSales2	false	true	pg_default
13	42500	ESP	false	true	pg_default
14	43559	test_db	false	true	pg_default

2.

test_db/postgres@PostgreSQL 17* X

test_db/postgres@PostgreSQL 17

No limit

Query Query History

```
1 select *
2 from pg_tablespace
```

Data Output Messages Notifications

	oid [PK] oid	spcname name	spcowner oid	spcacl aclitem[]	spcoptions text[]
1	1663	pg_default	10	[null]	[null]
2	1664	pg_global	10	[null]	[null]
3	43564	mytablespace	10	[null]	[null]

AuctionHouseDB/postgres@PostgreSQL 17* X

AuctionHouseDB/postgres@PostgreSQL 17

No limit

Query Query History

```
1 select d.oid, d.datname, d.datistemplate, d.dataallowconn, t
2 from pg_database d
3 join pg_tablespace t on t.oid = d.dattablespace
```

Data Output Messages Notifications

	oid oid	datname name	datistemplate boolean	dataallowconn boolean	spcname name
1	5	postgres	false	true	pg_default
2	1	template1	true	true	pg_default
3	4	template0	true	false	pg_default
4	17463	gym_membership	false	true	pg_default
5	16776	reweb_old	false	true	pg_default
6	18585	reweb	false	true	pg_default
7	20302	AuctionHouseDB	false	true	pg_default
8	20709	sales_history	false	true	pg_default
9	28810	RealEstateDB	false	true	pg_default
10	16852	dvdrental	false	true	pg_default
11	34470	VehiclesSales	false	true	pg_default
12	39126	VehilesSales2	false	true	pg_default
13	42500	ESP	false	true	pg_default
14	43559	test_db	false	false	mytablespace

3.

test_db/postgres@PostgreSQL 17* x

test_db/postgres@PostgreSQL 17

Query Query History

```

1 CREATE SCHEMA IF NOT EXISTS labs;
2
3 CREATE TABLE IF NOT EXISTS labs.person (
4   id integer NOT NULL,
5   name varchar(15)
6 );
7
8 SELECT schemaname, tablename FROM pg_tables
9 WHERE tablename = 'person';

```

Data Output Messages Notifications

	schemaname name	tablename name
1	labs	person

test_db/postgres@PostgreSQL 17* x

test_db/postgres@PostgreSQL 17

Query Query History

```

1 CREATE SCHEMA IF NOT EXISTS labs;
2
3 CREATE TABLE IF NOT EXISTS labs.person (
4   id integer NOT NULL,
5   name varchar(15)
6 );
7
8 INSERT INTO labs.person VALUES(1, 'Bob');
9 INSERT INTO labs.person VALUES(2, 'Alice');
10 INSERT INTO labs.person VALUES(3, 'Robert');
11
12 SELECT * from labs.person
13

```

Data Output Messages Notifications

	id integer	name character varying (15)
1	1	Bob
2	2	Alice
3	3	Robert

test_db/postgres@PostgreSQL 17* x

test_db/postgres@PostgreSQL 17

Query Query History

```

1 SHOW search_path;
2
3 SET search_path TO labs;
4
5 INSERT INTO person VALUES(1, 'Bob');
6 INSERT INTO person VALUES(2, 'Alice');
7 INSERT INTO person VALUES(3, 'Robert');
8
9 SELECT * from labs.person;

```

Data Output Messages Notifications

	id integer	name character varying (15)
1	1	Bob
2	2	Alice
3	3	Robert
4	1	Bob
5	2	Alice
6	3	Robert

Key Takeaways: Set `search_path` allows insert records without specifying the schema name

4.

test_db/postgres@PostgreSQL 17* × labs.person/test_d... ×

test_db/postgres@PostgreSQL 17

Query

Query History

```

1 CREATE EXTENSION IF NOT EXISTS pageinspect;
2
3 SET search_path TO labs;
4
5 select p.id, p.name, p.ctid, p.xmin, p.xmax from pe
6

```

Data Output

Messages

Notifications

SQL

	id	name	ctid	xmin	xmax
	integer	character varying (15)	tid	xid	xid
1	1	Bob	(0,1)	4549	0
2	2	Alice	(0,2)	4549	0
3	3	Robert	(0,3)	4549	0
4	1	Bob	(0,4)	4550	0
5	2	Alice	(0,5)	4550	0
6	3	Robert	(0,6)	4550	0

test_db/postgres@PostgreSQL 17* × labs.person/test_d... ×

test_db/postgres@PostgreSQL 17

Query

Query History

```

1 CREATE EXTENSION IF NOT EXISTS pageinspect;
2
3 SET search_path TO labs;
4
5 select p.id, p.name, p.ctid, p.xmin, p.xmax from pe
6
7 SELECT t_xmin, t_xmax, t_ctid,
8 tuple_data_split('labs.person'::regclass, t_data,
9 t_infomask2, t_bits)
10 FROM heap_page_items(get_raw_page('labs.person',
11

```

Data Output

Messages

Notifications

SQL

	t_xmin	t_xmax	t_ctid	tuple_data_split
	xid	xid	tid	bytea[]
1	4549	0	(0,1)	[binary data]
2	4549	0	(0,2)	[binary data]
3	4549	0	(0,3)	[binary data]
4	4550	0	(0,4)	[binary data]
5	4550	0	(0,5)	[binary data]
6	4550	0	(0,6)	[binary data]

test_db/postgres@PostgreSQL 17* × labs.person/test_d... ×

test_db/postgres@PostgreSQL 17

Query

Query History

```

1 CREATE EXTENSION IF NOT EXISTS pageinspect;
2
3 SET search_path TO labs;
4
5 -- SELECT t_xmin, t_xmax, t_ctid,
6 -- tuple_data_split('labs.person'::regclass, t_data, t_
7 -- t_infomask2, t_bits)
8 -- FROM heap_page_items(get_raw_page('labs.person', 0))
9
10 -- INSERT INTO person VALUES(4, 'John');
11
12 UPDATE person set name = 'Alex' where id = 2;
13
14 select p.id, p.name, p.ctid, p.xmin, p.xmax from person
15

```

Data Output

Messages

Notifications

SQL

	id	name	ctid	xmin	xmax
	integer	character varying (15)	tid	xid	xid
1	1	Bob	(0,1)	4549	0
2	3	Robert	(0,3)	4549	0
3	1	Bob	(0,4)	4550	0
4	3	Robert	(0,6)	4550	0
5	4	John	(0,7)	4551	0
6	2	Alex	(0,8)	4552	0
7	2	Alex	(0,9)	4552	0

test_db/postgres@PostgreSQL 17* × labs.person/test_d... ×

test_db/postgres@PostgreSQL 17

Query

Query History

```

1 CREATE EXTENSION IF NOT EXISTS pageinspect;
2
3 SET search_path TO labs;
4
5 -- SELECT t_xmin, t_xmax, t_ctid,
6 -- tuple_data_split('labs.person'::regclass, t_data, t_
7 -- t_infomask2, t_bits)
8 -- FROM heap_page_items(get_raw_page('labs.person', 0))
9
10 -- INSERT INTO person VALUES(4, 'John');
11
12 UPDATE person set name = 'Alex' where id = 2;
13
14 select p.id, p.name, p.ctid, p.xmin, p.xmax from person
15

```

Data Output

Messages

Notifications

SQL

	id	name	ctid	xmin	xmax
	integer	character varying (15)	tid	xid	xid
1	1	Bob	(0,1)	4549	0
2	3	Robert	(0,3)	4549	0
3	1	Bob	(0,4)	4550	0
4	3	Robert	(0,6)	4550	0
5	4	John	(0,7)	4551	0
6	2	Alex	(0,8)	4552	0
7	2	Alex	(0,9)	4552	0

test_db/postgres@PostgreSQL 17* × labs.person/test_d... ×

test_db/postgres@PostgreSQL 17

No limit

Query

Query History

```

1 CREATE EXTENSION IF NOT EXISTS pageinspect;
2
3 SET search_path TO labs;
4
5 SELECT t_xmin, t_xmax, t_ctid,
6        tuple_data_split('labs.person'::regclass, t_data, t_infomask2, t_bits)
7        FROM heap_page_items(get_raw_page('labs.person', 0));
8
9 -- INSERT INTO person VALUES(4, 'John');
10
11 -- UPDATE person set name = 'Alex' where id = 2;
12

```

Data Output

Messages

Notifications

SQL

	t_xmin xid	t_xmax xid	t_ctid tid	tuple_data_split bytea[]
1	4549	0	(0,1)	[binary data]
2	4549	4552	(0,8)	[binary data]
3	4549	0	(0,3)	[binary data]
4	4550	0	(0,4)	[binary data]
5	4550	4552	(0,9)	[binary data]
6	4550	0	(0,6)	[binary data]
7	4551	0	(0,7)	[binary data]
8	4552	0	(0,8)	[binary data]
9	4552	0	(0,9)	[binary data]

test_db/postgres@PostgreSQL 17* × labs.person/test_d... ×

test_db/postgres@PostgreSQL 17

No limit

Query

Query History

```

1 CREATE EXTENSION IF NOT EXISTS pageinspect;
2
3 SET search_path TO labs;
4
5 -- INSERT INTO person VALUES(4, 'John');
6 -- UPDATE person set name = 'Alex' where id = 2;
7
8 -- select p.id, p.name, p.ctid, p.xmin, p.xmax from person
9
10 DELETE FROM person WHERE id = 3;
11
12 SELECT t_xmin, t_xmax, t_ctid,
13        tuple_data_split('labs.person'::regclass, t_data, t_infomask2, t_bits)
14        FROM heap_page_items(get_raw_page('labs.person', 0));
15

```

Data Output

Messages

Notifications

SQL

	t_xmin xid	t_xmax xid	t_ctid tid	tuple_data_split bytea[]
1	4549	0	(0,1)	[binary data]
2	4549	4552	(0,8)	[binary data]
3	4549	4553	(0,3)	[binary data]
4	4550	0	(0,4)	[binary data]
5	4550	4552	(0,9)	[binary data]
6	4550	4553	(0,6)	[binary data]
7	4551	0	(0,7)	[binary data]
8	4552	0	(0,8)	[binary data]
9	4552	0	(0,9)	[binary data]

test_db/postgres@PostgreSQL 17* × labs.person/test_d... ×

test_db/postgres@PostgreSQL 17

No limit

Query

Query History

```

9
10 -- DELETE FROM person WHERE id = 3;
11
12 INSERT INTO person VALUES(999, 'Test');
13
14 SELECT t_xmin, t_xmax, t_ctid,
15        tuple_data_split('labs.person'::regclass, t_data, t_infomask2, t_bits)
16        FROM heap_page_items(get_raw_page('labs.person', 0));
17
18
19

```

Data Output

Messages

Notifications

SQL

	t_xmin xid	t_xmax xid	t_ctid tid	tuple_data_split bytea[]
1	4549	0	(0,1)	[binary data]
2	4549	4552	(0,8)	[binary data]
3	4549	4553	(0,3)	[binary data]
4	4550	0	(0,4)	[binary data]
5	4550	4552	(0,9)	[binary data]
6	4550	4553	(0,6)	[binary data]
7	4551	0	(0,7)	[binary data]
8	4552	0	(0,8)	[binary data]
9	4552	0	(0,9)	[binary data]
10	4554	0	(0,10)	[binary data]

test_db/postgres@PostgreSQL 17* × labs.person/test_d... ×

test_db/postgres@PostgreSQL 17

No limit

Query

Query History

```

12 -- INSERT INTO person VALUES(999, 'Test');
13
14 DELETE FROM person WHERE id = 999;
15
16 SELECT t_xmin, t_xmax, t_ctid,
17        tuple_data_split('labs.person'::regclass, t_data, t_infomask2, t_bits)
18        FROM heap_page_items(get_raw_page('labs.person', 0));
19
20

```

Data Output

Messages

Notifications

SQL

	t_xmin xid	t_xmax xid	t_ctid tid	tuple_data_split bytea[]
1	4549	0	(0,1)	[binary data]
2	4549	4552	(0,8)	[binary data]
3	4549	4553	(0,3)	[binary data]
4	4550	0	(0,4)	[binary data]
5	4550	4552	(0,9)	[binary data]
6	4550	4553	(0,6)	[binary data]
7	4551	0	(0,7)	[binary data]
8	4552	0	(0,8)	[binary data]
9	4552	0	(0,9)	[binary data]
10	4554	4555	(0,10)	[binary data]

Key Takeaways:

- `pageinspect` shows internal row versioning, helping to understand MVCC behavior.
- PostgreSQL does not update rows directly. Instead, it creates a new row version (UPDATE behaves like an INSERT + DELETE).
- Deleted rows remain in the database but are marked as obsolete (xmax is assigned).
- New inserts always get a fresh xmin, representing their transaction ID.

5.

test_db/postgres@PostgreSQL 17*
labs.person/test_d..

test_db/postgres@PostgreSQL 17

Query
Query History

```

1 vacuum labs.person;
2
3 SELECT t_xmin, t_xmax, t_ctid,
4 tuple_data_split('labs.person'::regclass, t_
5 t_infomask2, t_bits)
6 FROM heap_page_items(get_raw_page('labs.perso

```

Data Output
Messages
Notifications

	t_xmin xid	t_xmax xid	t_ctid tid	tuple_data_split bytea[]
1	4549	0	(0,1)	[binary data]
2	[null]	[null]	[null]	[null]
3	[null]	[null]	[null]	[null]
4	4550	0	(0,4)	[binary data]
5	[null]	[null]	[null]	[null]
6	[null]	[null]	[null]	[null]
7	4551	0	(0,7)	[binary data]
8	4552	0	(0,8)	[binary data]
9	4552	0	(0,9)	[binary data]

test_db/postgres@PostgreSQL 17*
labs.person/test_d..

test_db/postgres@PostgreSQL 17

Query
Query History

```

1 SET search_path TO labs;
2
3 INSERT INTO person VALUES(5, 'Sarah');
4
5 SELECT t_xmin, t_xmax, t_ctid,
6 tuple_data_split('labs.person'::regclass, t_
7 t_infomask2, t_bits)
8 FROM heap_page_items(get_raw_page('labs.persc

```

Data Output
Messages
Notifications

	t_xmin xid	t_xmax xid	t_ctid tid	tuple_data_split bytea[]
1	4549	0	(0,1)	[binary data]
2	[null]	[null]	[null]	[null]
3	4556	0	(0,3)	[binary data]
4	4550	0	(0,4)	[binary data]
5	[null]	[null]	[null]	[null]
6	[null]	[null]	[null]	[null]
7	4551	0	(0,7)	[binary data]
8	4552	0	(0,8)	[binary data]
9	4552	0	(0,9)	[binary data]

test_db/postgres@PostgreSQL 17*
labs.person/test..

test_db/postgres@PostgreSQL 17

Query
Query History

```

1 vacuum full labs.person;
2 SELECT t_xmin, t_xmax, t_ctid,
3 tuple_data_split('labs.person'::regclass, t_
4 t_infomask2, t_bits)
5 FROM heap_page_items(get_raw_page('labs.per

```

Data Output
Messages
Notifications

	t_xmin xid	t_xmax xid	t_ctid tid	tuple_data_split bytea[]
1	4549	0	(0,1)	[binary data]
2	4556	0	(0,2)	[binary data]
3	4550	0	(0,3)	[binary data]
4	4551	0	(0,4)	[binary data]
5	4552	0	(0,5)	[binary data]
6	4552	0	(0,6)	[binary data]

Key Takeaways:

- Before VACUUM: Deleted and updated rows exist as dead tuples.
- After VACUUM: Dead tuples are marked as reusable space.
- After VACUUM FULL: Dead tuples are completely removed, reducing table size.
- Inserting new rows after VACUUM: PostgreSQL reuses free space instead of expanding the table file.