PostgreSQL pgroll: Zero downtime, reversible schema changes

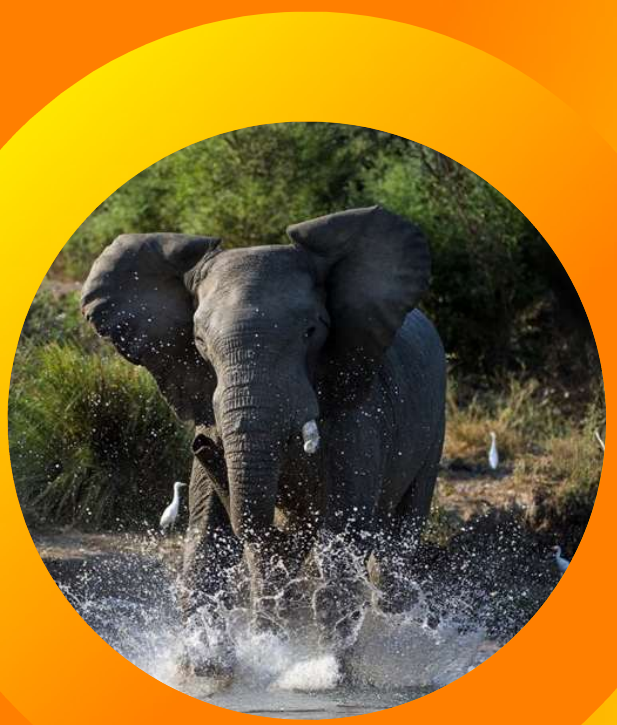
Nishchay Kothari

Technical Consultant
Fujitsu

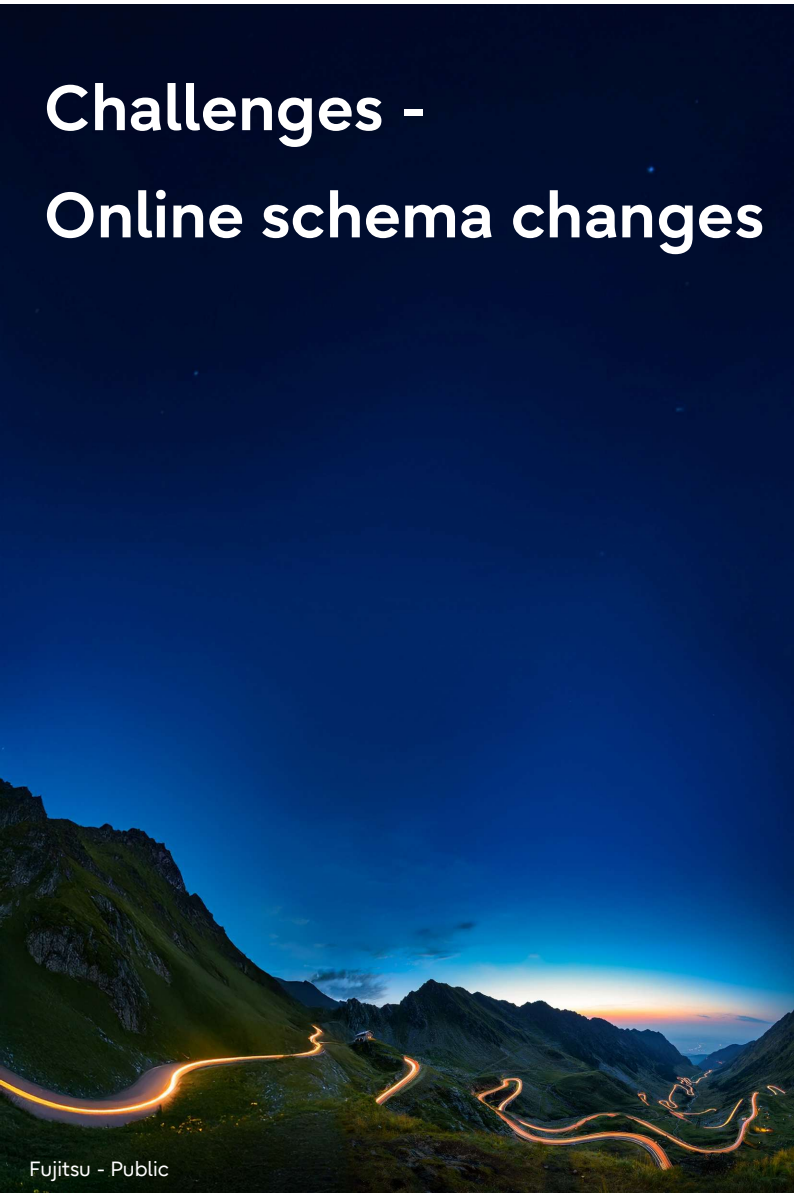
PG Down Under 2025

Agenda

- Challenges – Online schema changes
- Introduction - pgroll
- How it works
- Installation
- Usage
  - Add tables
  - Add columns
  - Add & rollback index
- Q&A



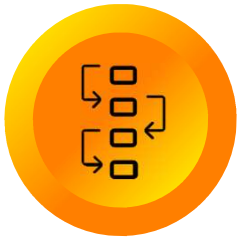
Challenges -
Online schema changes



FUJITSU

Online schema changes are painful

Breaking Applications



Multiple steps

Unexpected database locks

No easy rollbacks

Introduction - pgroll

Open-source command-line tool

Safe and reversible schema changes for PostgreSQL

Works with Postgres 14.0 or later

pg roll

FUJITSU

Features

Zero downtime changes (no DB locking, no breaking changes)

Keep old and new schema versions **working simultaneously**

Automatic **columns backfilling** when needed

Instant rollback in case of issues during change

Works against existing schemas, **no need to start from scratch**

Works with **any Postgres service**

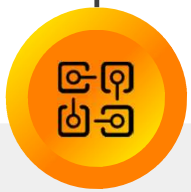
Written in Go, **cross-platform single binary** with no external dependencies

Understanding how pgroll works

FUJITSU

Virtual schemas using views

- pgroll creates virtual schemas using views instead of modifying physical tables directly
- This ensures existing users experience no disruption while database changes are being introduced



Controlled, non-disruptive changes

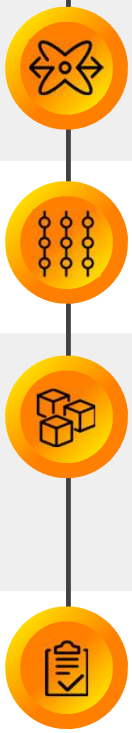
- Database structure modifications happen without breaking the existing schema
- Applications continue to function normally while schema changes are rolled out in the background

No immediate impact on users

- Unlike traditional schema changes, pgroll ensures smooth transitions by mapping new changes through views
- Client applications are not forced to upgrade immediately, allowing flexibility in rollout

pgroll - Expand/contract workflow

FUJITSU



Expand phase (safe additions)

- New tables and columns are added first, ensuring no existing functionality breaks
- Applications can start using the new schema without affecting legacy users

Handling breaking changes

- Instead of modifying existing columns directly, pgroll creates new columns
- Triggers synchronize data between old and new columns to maintain consistency

Exposing the new schema

- The updated schema is made available as a new version
- Applications can gradually transition to the new schema version
- Applications start using the new schema while the old version remains active
- This staged rollout ensures smooth adoption without downtime

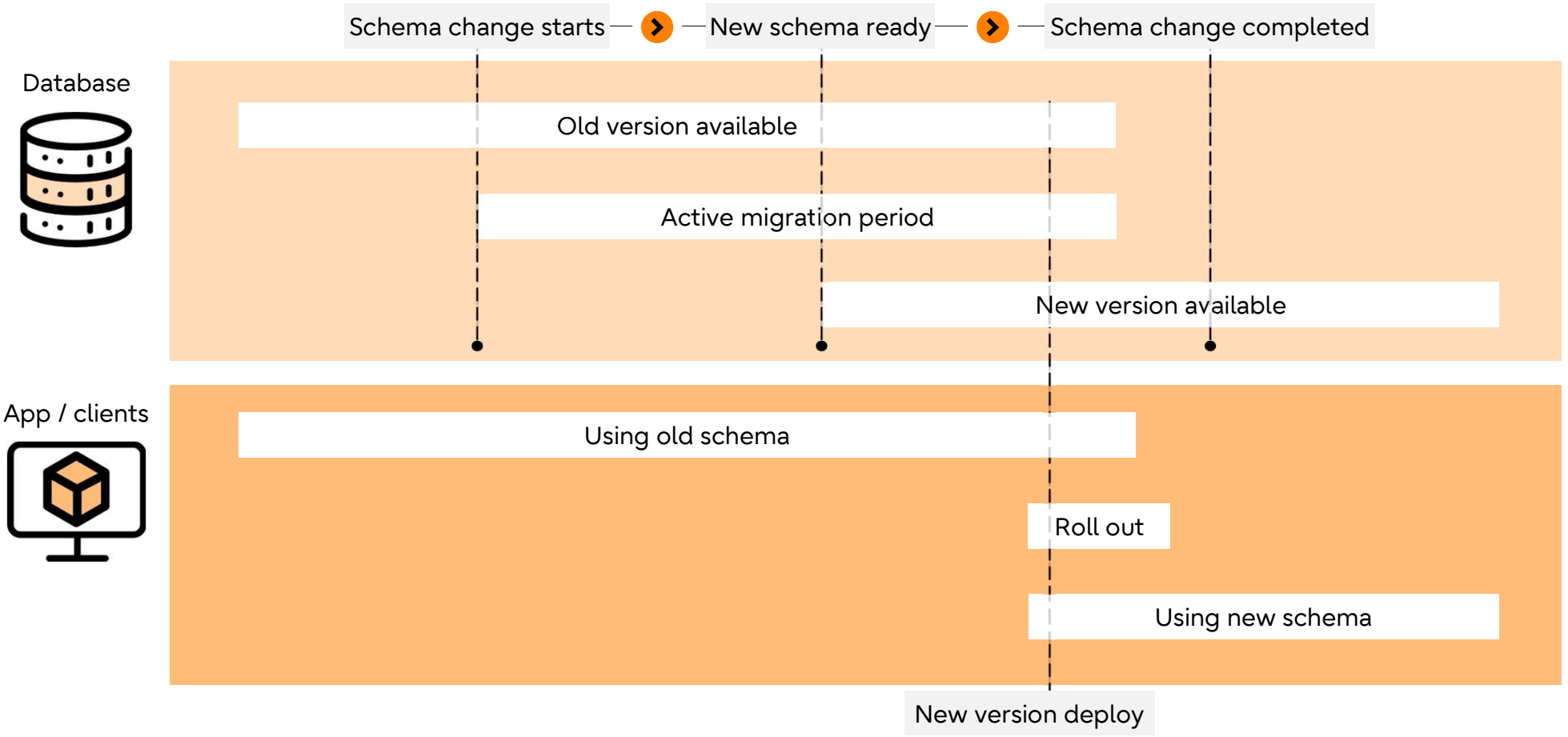
Completing the change

- When all applications have transitioned, the old schema is removed
- Unused tables and columns are cleaned up, and final names are assigned
- The system remains stable, ensuring continuous operation

How it works

FUJITSU

Schema changes flow

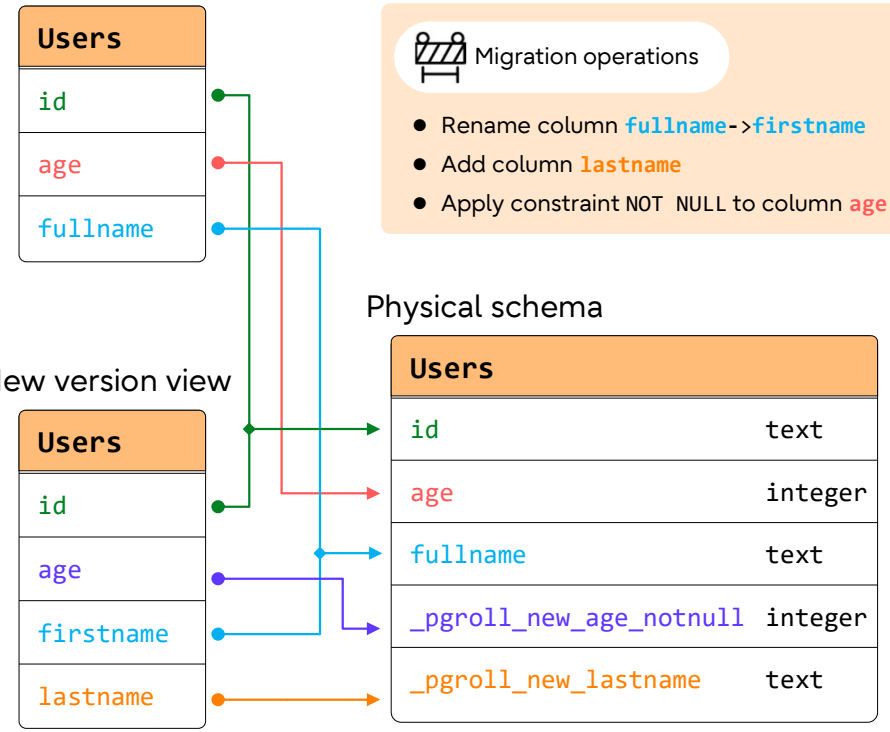


How it works

FUJITSU

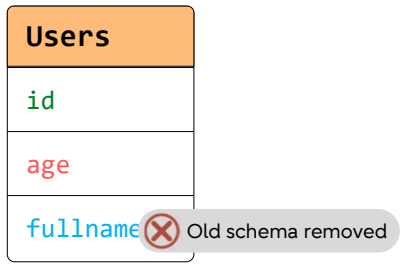
Active migration period

Old version view



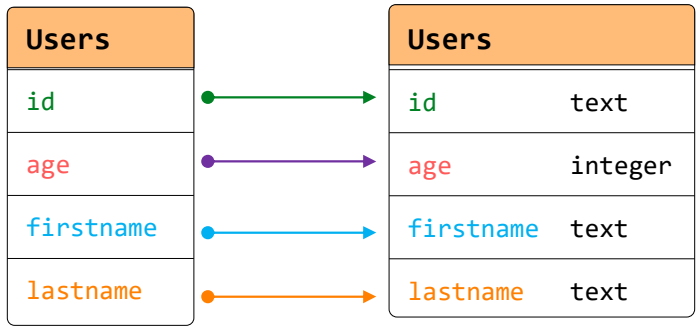
Migration complete

Old version view



New version view

Physical schema



How to install

FUJITSU

- Prerequisite: Go 1.24 or later

- To install pgroll from the source:

```
go install github.com/xataio/pgroll@latest
```

- The pgroll binary will be in `/root/go/bin`

- Copy the pgroll binary to `/usr/bin`

How to use

FUJITSU

- Prepare the database
  - Store some internal state in the database (Table & Functions)

SSL setup

```
pgroll init --postgres-url postgres://user:password@host:port/dbname
```

No SSL setup

```
pgroll init --postgres-url postgres://user:password@host:port/dbname?sslmode=disable
```

```
[fepuser@fep17-rhel9 ~]$ pgroll --postgres-url
postgres://fepuser:train@1987@localhost:27500/test_pgroll?sslmode=disable init
 SUCCESS  Initialization complete
[fepuser@fep17-rhel9 ~]$ psql -d test_pgroll
psql (17.0)
Type "help" for help.

test_pgroll=# \dn
      List of schemas
 Name   | Owner
--------+--------------------
 pgroll | fepuser
 public | pg_database_owner
(2 rows)

test_pgroll=# \dt pgroll.*
              List of relations
 Schema |      Name      | Type  |  Owner
--------+----------------+-------+---------
 pgroll | migrations     | table | fepuser
 pgroll | pgroll_version | table | fepuser
(2 rows)
```

How to add tables

FUJITSU

- Create a change file
  - You can check various examples at https://github.com/xataio/pgroll/tree/main/examples

```
vi create_users_table.json
{
 "operations": [
   {
     "create_table": {
       "name": "users",
       "columns": [
         {
           "name": "id",
           "type": "serial",
           "pk": true
         },
         {
           "name": "username",
           "type": "varchar(50)",
           "nullable": false
         },
         {
           "name": "email",
           "type": "varchar(100)",
           "nullable": true
         }
       ]
     }
   }
 ]
}
```

Users	
id	serial
username	varchar(50)
email	varchar(100)

How to add tables

FUJITSU

- Start the changes

```
[fepuser@fep17-rhel9 ~]$ pgroll --postgres-url postgres://fepuser:train@1987@localhost:27500/test_pgroll?sslmode=disable start create_users_table.json
 SUCCESS  New version of the schema available under the postgres "public_create_users_table" schema

[fepuser@fep17-rhel9 ~]$ psql -d test_pgroll
psql (17.0)
Type "help" for help.

test_pgroll=# \dt pgroll.*
           List of relations
 Schema |     Name   | Type  | Owner
--------+------------+-------+----------
 pgroll | migrations | table | fepuser
 (1 row)

test_pgroll=# SELECT * FROM pgroll.migrations;
 schema |       name         |                migration            | created at    | updated at    | parent | done | resulting schema | migration_type
--------+--------------------+-------------------------------------+---------------+---------------+--------+------+------------------+----------------
 public | create_users_table | {" operations": [{"create_table":... | 2025-09-15... | 2025-09-15... |        |   f  | {}               | pgroll
(1 row)

test_pgroll=# \dt public.*
              List of relations
 Schema | Name  | Type  |  Owner
--------+-------+-------+---------
 public | users | table | fepuser
(1 row)
```

How to add tables

FUJITSU

- Check the status and complete the changes

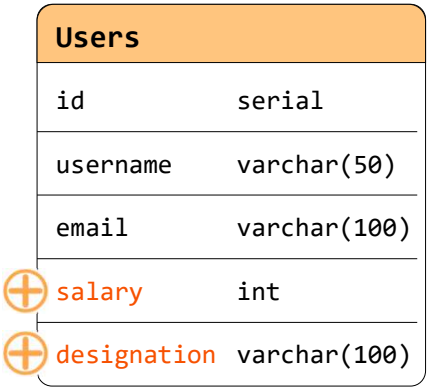
```
[fepuser@fep17-rhel9 ~]$ pgroll --postgres-url postgres://fepuser:train@1987@localhost:27500/test_pgroll?sslmode=disable status
{
  "schema": "public",
  "version": "create_users_table",
❯ "status": "In progress"
}

[fepuser@fep17-rhel9 ~]$ pgroll --postgres-url postgres://fepuser:train@1987@localhost:27500/test_pgroll?sslmode=disable complete
 SUCCESS  Migration successful!

[fepuser@fep17-rhel9 ~]$ psql -d test_pgroll
psql (17.0)
Type "help" for help.

test_pgroll=# \dt public.users
        List of relations
 Schema | Name  | Type  |  Owner
--------+-------+-------+---------
 public | users | table | fepuser
(1 row)

[fepuser@fep17-rhel9 ~]$ pgroll --postgres-url postgres://fepuser:train@1987@localhost:27500/test_pgroll?sslmode=disable status
{
  "schema": "public",
  "version": "create_users_table",
❯ "status": "Complete"
}
```

How to add columns

FUJITSU

- Create a change file

```
vi add_salary_and_designation_to_users.json
{
"operations": [
    {
      "add_column": {
        "table": "users",
        "column": {
          "name": "salary",
          "type": "int",
          "nullable": false,
          "default": "5000"
        }
      }
    },
    {
      "add_column": {
        "table": "users",
        "column": {
          "name": "designation",
          "type": "varchar(100)",
          "nullable": true
        }
      }
    }
  ]
}
```



How to add columns

FUJITSU

- Start the changes and check the status

```
[fepuser@fep17-rhel9 ~]$ pgroll --postgres-url postgres://fepuser:train@1987@localhost:27500/test_pgroll?sslmode=disable start add_salary_and_designation_to_users.json
 SUCCESS  New version of the schema available under the postgres "public_add_salary_and_designation_to_users" schema

[fepuser@fep17-rhel9 ~]$ pgroll --postgres-url postgres://fepuser:train@1987@localhost:27500/test_pgroll?sslmode=disable status
{
  "schema": "public",
  "version": "add_salary_and_designation_to_users",
> "status": "In progress"
}

[fepuser@fep17-rhel9 ~]$ psql -d test_pgroll
psql (17.0)
Type "help" for help.

test_pgroll=# \d public.users
                                        Table "public.users"
         Column          |         Type           | Collation | Nullable |                    Default
-------------------------+------------------------+-----------+----------+-----------------------------------------------
 id                      | integer                |           | not null | nextval('_pgroll_new_users_id_seq'::regclass)
 username                | character varying(50)  |           | not null |
 email                   | character varying(100) |           |          |
> _pgroll_new_salary      | integer                |           | not null | 5000
> _pgroll_new_designation | character varying(100) |           |          |
Indexes:
    "users_pkey" PRIMARY KEY, btree (id)
```

How to add columns

FUJITSU

- Complete the changes

```
[fepuser@fep17-rhel9 ~]$ pgroll --postgres-url postgres://fepuser:train@1987@localhost:27500/test_pgroll?sslmode=disable complete
 SUCCESS  Migration successful!
[fepuser@fep17-rhel9 ~]$ psql -d test_pgroll
psql (17.0)
Type "help" for help.

test_pgroll=# \d public.users
 Table "public.users"
   Column    |         Type           | Collation | Nullable |                    Default
-------------+------------------------+-----------+----------+-----------------------------------------------
 id          | integer                |           | not null | nextval('_pgroll_new_users_id_seq'::regclass)
 username    | character varying(50)  |           | not null |
 email       | character varying(100) |           |          |
>salary      | integer                |           | not null | 5000
>designation | character varying(100) |           |          |
Indexes:
    "users_pkey" PRIMARY KEY, btree (id)
```

How to add indexes

FUJITSU

- Create a change file

```
vi add_create_index_to_users.json
{
  "operations": [
    {
      "create_index":
              {
        "name": "idx_salary_column",
        "table": "users",
        "columns":{
          "salary": {}
      }
              }
    }
  ]
}
```

Users		
id	serial	
username	varchar(50)	
email	varchar(100)	
salary	int	idx_salary_column
designation	varchar(100)	

How to add indexes

FUJITSU

- Start the changes and check the status

```
[fepuser@fep17-rhel9 ~]$ pgroll --postgres-url postgres://fepuser:train@1987@localhost:27500/test_pgroll?sslmode=disable start add_index_salary_column.json
SUCCESS New version of the schema available under the postgres "public_add_create_index_to_users" schema

[fepuser@fep17-rhel9 ~]$ pgroll --postgres-url postgres://fepuser:train@1987@localhost:27500/test_pgroll?sslmode=disable status
{
  "schema": "public",
  "version": "add_create_index_to_users",
❯ "status": "In progress"
}

[fepuser@fep17-rhel9 ~]$ psql -d test_pgroll
psql (17.0)
Type "help" for help.

test_pgroll=# \d public.users
                                        Table "public.users"
   Column    |         Type           | Collation | Nullable |                    Default
-------------+------------------------+-----------+----------+-----------------------------------------------
 id          | integer                |           | not null | nextval('_pgroll_new_users_id_seq'::regclass)
 username    | character varying(50)  |           | not null |
 email       | character varying(100) |           |          |
 salary      | integer                |           | not null | 5000
 designation | character varying(100) |           |          |
Indexes:
    "users_pkey" PRIMARY KEY, btree (id)
  ❯ "idx_salary_column" btree (salary)
```

How to add indexes

FUJITSU

- Roll back

```
[fepuser@fep17-rhel9 ~]$ pgroll --postgres-url postgres://fepuser:train@1987@localhost:27500/test_pgroll?sslmode=disable rollback
 SUCCESS  Migration rolled back. Changes made since the last version have been reverted

[fepuser@fep17-rhel9 ~]$ psql -d test_pgroll
psql (17.0)
Type "help" for help.

test_pgroll=# \d public.users
                                      Table "public.users"
   Column    |         Type           | Collation | Nullable |                    Default
-------------+------------------------+-----------+----------+-----------------------------------------------
 id          | integer                |           | not null | nextval('_pgroll_new_users_id_seq'::regclass)
 username    | character varying(50)  |           | not null |
 email       | character varying(100) |           |          |
 salary      | integer                |           | not null | 5000
 designation | character varying(100) |           |          |
Indexes:
  > "users_pkey" PRIMARY KEY, btree (id)

test_pgroll=# \q

[fepuser@fep17-rhel9 ~]$ pgroll --postgres-url postgres://fepuser:train@1987@localhost:27500/test_pgroll?sslmode=disable status
{
  "schema": "public",
  "version": "add_create_index_to_users",
> "status": "Complete"
}
```

Comparison with other tools

FUJITSU

	 pgroll	Liquibase Flyway	schemachange	DBeaver pgAdmin
	pgroll	**Liquibase & Flyway**	**Schemachange**	**Dbeaver & pgAdmin**
Purpose	Zero downtime, reversible schema changes	Schema versioning and migrations as part of CI/CD pipelines	Snowflake-specific migrations	Database management
Zero downtime	Ensures uninterrupted database operations	Not inherently designed for zero downtime, requires additional manual steps for the same	Not a primary focus	Not applicable

Q&A

Thank you

Nishchay Kothari

Technical Consultant
Fujitsu

SCAN ME

SCAN ME