

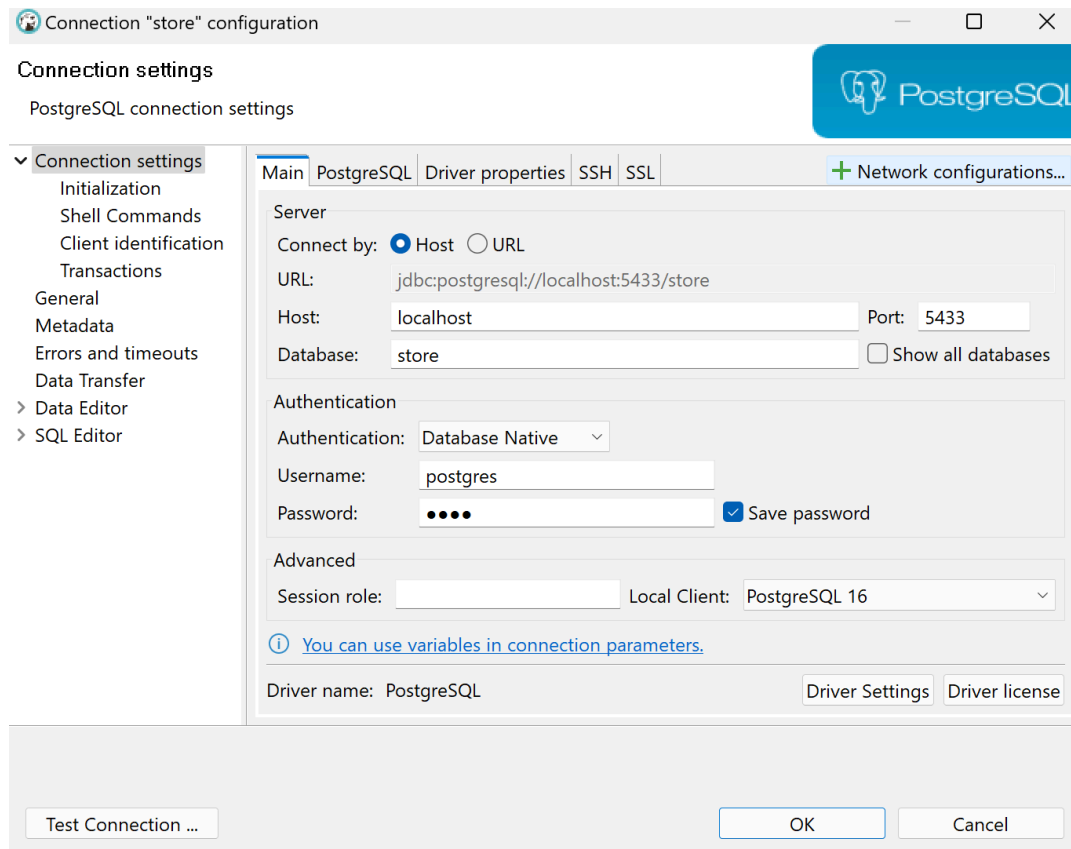
1. Jalankan citus di komputer lokal dengan menggunakan docker compose

```
Ahmad Fathoni A@Laptoptoni MINGW64 /c/Alterra
$ git clone https://github.com/Immersive-DataEngineer-Resource/citus-demo.git
Cloning into 'citus-demo'...
remote: Enumerating objects: 12, done.
remote: Counting objects: 100% (12/12), done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 12 (delta 0), reused 12 (delta 0), pack-reused 0
Receiving objects: 100% (12/12), 5.02 KiB | 734.00 KiB/s, done.
```

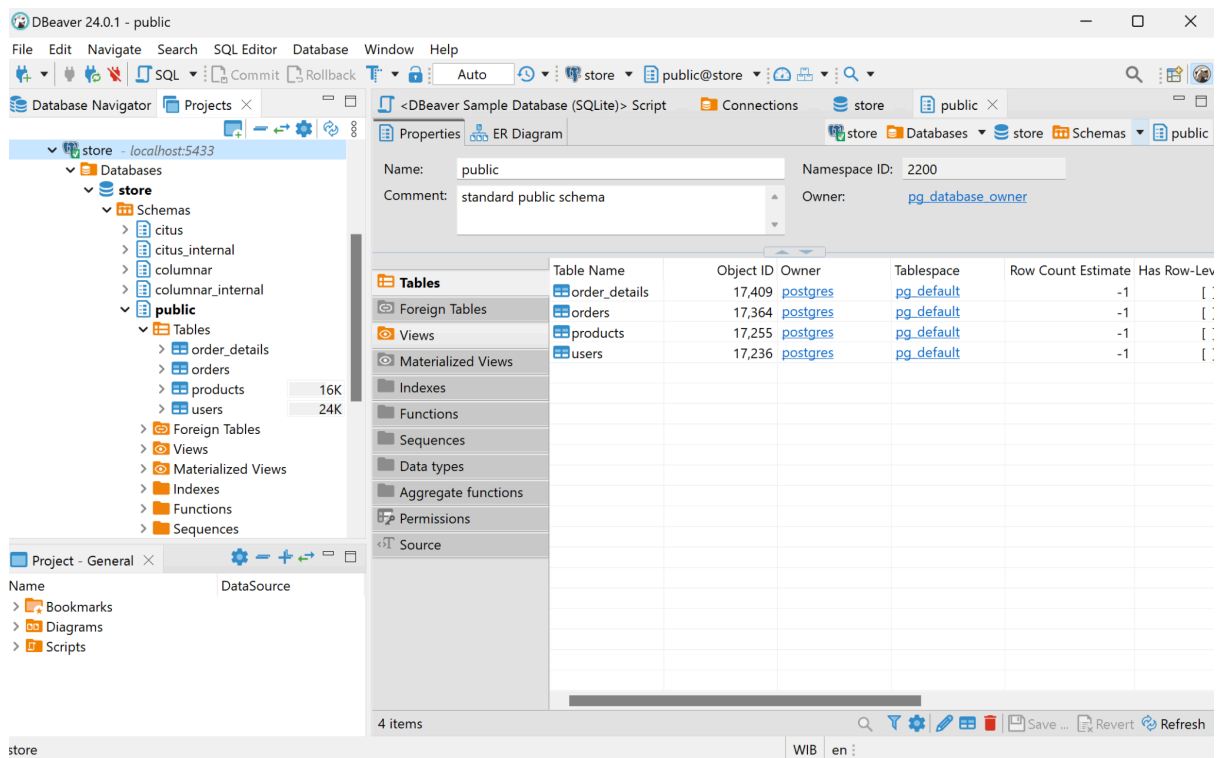
Melakukan git clone terhadap repo citus-demo ke dalam folder Alterra

```
Ahmad Fathoni A@Laptoptoni MINGW64 /c/Alterra/citus-demo (main)
$ docker compose up -d
worker-3 Pulling
worker-1 Pulling
worker-2 Pulling
master Pulling
faef57eae888 Pulling fs layer
a33c10a72186 Pulling fs layer
d662a43776d2 Pulling fs layer
a3ba86413420 Pulling fs layer
a627f37e9916 Pulling fs layer
424bade69494 Pulling fs layer
dd8d4fcd466b Pulling fs layer
a3ba86413420 waiting
a627f37e9916 waiting
424bade69494 waiting
03d0efeea592 Pulling fs layer
dd8d4fcd466b waiting
4f27e1518a67 Pulling fs layer
03d0efeea592 waiting
0c8ac8b8eb90 Pulling fs layer
4f27e1518a67 waiting
c08e79653ad2 Pulling fs layer
d5724e8c22af Pulling fs layer
0c8ac8b8eb90 waiting
```

Melakukan docker compose up terhadap folder citus demo untuk menjalankan aplikasi citus-demo



Mengkoneksikan docker compose citus-demo dengan dbeaver



docker compose citus-demo sudah terkoneksi, dan tabel sudah muncul

2. Tuliskan perintah untuk membuat tabel

a. tabel biasa/head method

```
create table events_row as select * from events_columnar;
```

Membuat table events_row yang isinya (baris dan data) sama seperti dengan events_columnar

b. tabel columnar

```
CREATE TABLE events_columnar (  
    device_id bigint,  
    event_id bigserial,  
    event_time timestamp default now(),  
    data jsonb not null  
)  
USING columnar;
```

Membuat table events_columnar, yang didalamnya terdapat device_id, event_id, event_time dan data, menggunakan metode columnar

3. Masukkan 100 baris ke dalam tabel biasa dan tabel columnar

```
INSERT INTO events_row (device_id, data)  
SELECT d, '{"hello":"columnar"}' FROM generate_series(1, 10000) d;
```

```
INSERT INTO events_columnar (device_id, data)  
SELECT d, '{"hello":"columnar"}' FROM generate_series(1, 10000) d;
```







Memasukkan data ke dalam events_row dan events_columnar dimana akan etrdapat 10000 baris

4. Tampilkan perbedaan ukuran antara table biasa dengan tabel columnar

```
select count(*) from events_columnar;  
  
select count(*) from events_row;  
  
select * from pg_catalog.pg_class;  
  
select * from pg_catalog.pg_am;  
  
select pg_class.relname as "Table Name", am.amname as "Method Access"  
from pg_catalog.pg_class as pg_class  
join pg_catalog.pg_am as am  
on pg_class.relam = am."oid"  
where  
    pg_class.relname in ('events_columnar', 'events_row');
```

Membandingkan antara tabel events_columnar dengan tabel events_row terhadap metode yang dipakai, hasilnya terlihat seperti dibawah

	ABC Table Name ▼	ABC Method Access ▼	
1	events_columnar	columnar	
2	events_row	heap	

▼	 public	
▼	 Tables	
>	 events_columnar	64K
>	 events_row	712K
>	 order_details	
>	 orders	

Ukuran tabel events_columnar lebih kecil ketimbang tabel events_row

5. Tuliskan kesimpulannya

Dengan menggunakan metode columnar baris di dalam table akan disusun secara berkelompok berdasarkan kolom sehingga ukuran data akan lebih kecil ketimbang dengan hanya menggunakan heap method, karena ukuran data lebih kecil maka data columnar lebih mudah untuk diakses dan kinerja nya lebih cepat

Pada contoh-contoh diatas kita dapat melihat bahwa kedua tabel dimasukkan 100 baris, akan tetapi table yang menggunakan metode columnar memiliki ukuran yang lebih kecil ketimbang table yang menggunakan metode heap