

**Nama : Kharisma Novi Chandramukti**

**Kelas/Batch : Data Engineering 4**

---

### **Part 3 – Replication + Sharding**

#### **1. Perbedaan antara replication dan sharding**

##### **a. Replication**

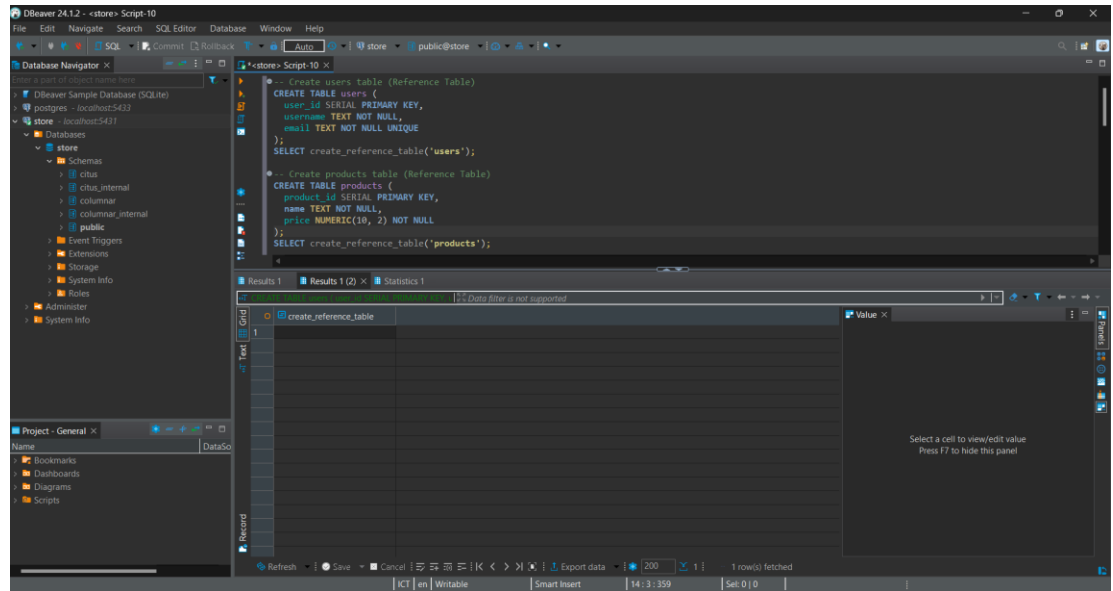
- Bertujuan untuk menciptakan dan memelihara beberapa salinan data yang sama di berbagai server untuk meningkatkan ketersediaan data, kinerja dan keandalan
- Memiliki karakteristik untuk memastikan data tetap konsisten dan terbaru di semua salinan
- Manfaatnya adalah mengurangi waktu downtime, meningkatkan ketersediaan data, dan memungkinkan pemulihan data setelah kegagalan
- Replication cocok untuk aplikasi yang memerlukan data yang selalu tersedia dan dapat diakses dari berbagai lokasi

##### **b. Sharding :**

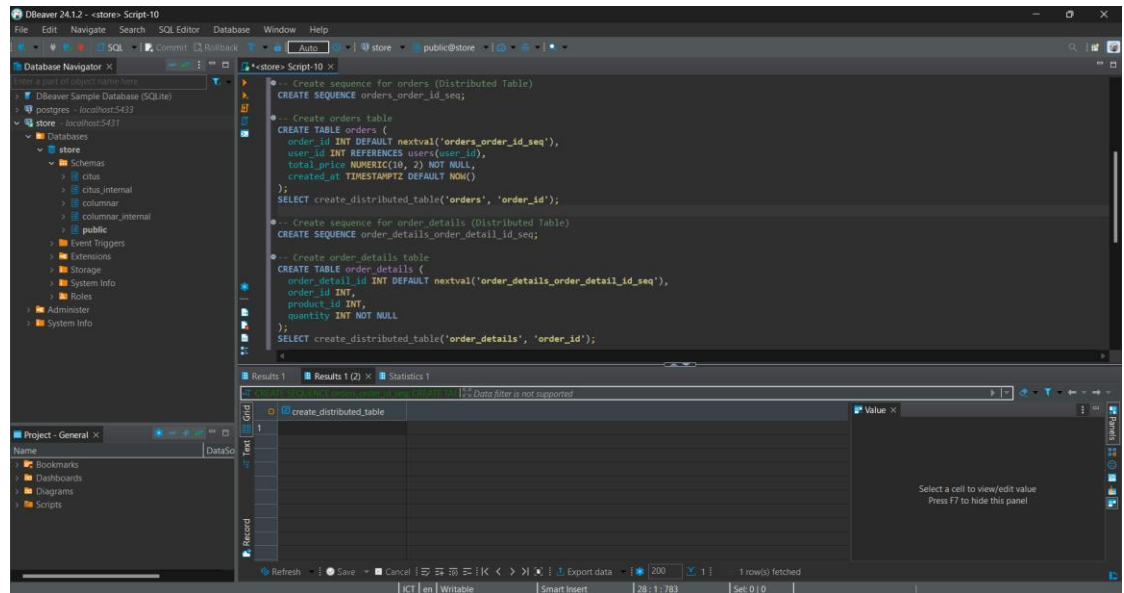
- Bertujuan untuk memecah database yang besar menjadi beberapa bagian yang lebih kecil untuk meningkatkan skalabilitas dan kinerja
- Memiliki karakteristik untuk memecah data menjadi “shards” yang disimpan di server yang berbeda, dengan setiap shard berisi Sebagian dari data asli
- Manfaatnya adalah meningkatkan skalabilitas, kinerja, dan ketersediaan data dengan memungkinkan penambahan server baru untuk menangani data yang semakin banyak
- Sharding cocok untuk aplikasi yang memerlukan skalabilitas horizontal dan dapat menangani data yang sangat besar

## 2. Membuat reference table + distributed tabel seperti pada repo github

### - Reference table



### - Distributed table



### 3. Node/worker pada product “Headphone” dan tunjukkan shard id nya

```
WITH placement AS (  
    SELECT  
        shardid as shard_id,  
        nodename as node_name  
    FROM pg_dist_shard_placement  
)  
  
*product_shard AS (  
    SELECT  
        3 as product_id,  
        get_shard_id_for_distribution_column('products', 3) as shard_id,  
        'products_' || get_shard_id_for_distribution_column('products', 3) as real_table_name  
    FROM products  
)  
  
*SELECT  
    product_shard.product_id,  
    product_shard.shard_id,  
    product_shard.real_table_name,  
    placement.node_name  
FROM product_shard  
JOIN placement ON placement.shard_id = product_shard.shard_id;
```

product_id	shard_id	real_table_name	node_name
3	102.009	products_102009	citius-demo_worker_1
3	102.009	products_102009	citius-demo_worker_1
3	102.009	products_102009	citius-demo_worker_1
3	102.009	products_102009	citius-demo_worker_1
3	102.009	products_102009	citius-demo_worker_2
3	102.009	products_102009	citius-demo_worker_2
3	102.009	products_102009	citius-demo_worker_2
3	102.009	products_102009	citius-demo_worker_2
3	102.009	products_102009	citius-demo_worker_3
3	102.009	products_102009	citius-demo_worker_3
3	102.009	products_102009	citius-demo_worker_3
3	102.009	products_102009	citius-demo_worker_3

### 4. Node/worker pada order dengan id 13 dan tunjukkan shard id nya

```
WITH placement AS (  
    SELECT  
        shardid as shard_id,  
        nodename as node_name  
    FROM pg_dist_shard_placement  
)  
  
*order_ids AS (  
    SELECT order_id  
    FROM orders  
    ORDER BY order_id  
    LIMIT 1  
)  
  
*order_shards AS (  
    SELECT  
        13 as order_id,  
        get_shard_id_for_distribution_column('orders', 13) as shard_id,  
        'orders_' || get_shard_id_for_distribution_column('orders', 13) as real_table_name  
    FROM order_ids  
)  
  
*SELECT  
    order_shards.*  
    , placement.node_name  
FROM order_shards  
INNER JOIN placement  
    ON placement.shard_id = order_shards.shard_id
```

order_id	shard_id	real_table_name	node_name
13	102.033	orders_102033	citius-demo_worker_3

5. Kapan sebaiknya kita menggunakan replication?

- Ketersediaan data : replication sangat berguna untuk memastikan data tetap tersedia meskipun salah satu server mengalami kegagalan
- Kinerja : dapat meningkatkan kinerja dengan membagi beban kerja antara beberapa server
- Pemulihan data : memungkinkan pemulihan data setelah kegagalan dengan menggunakan salinan data yang ada
- Distribusi data : memungkinkan data diakses dari berbagai lokasi, yang berguna untuk aplikasi global

6. Kapan sebaiknya kita menggunakan sharding?

- Skalabilitas horizontal : sharding berguna untuk memecah database yang besar menjadi beberapa bagian yang lebih kecil, memungkinkan penambahan server baru untuk menangani data yang semakin banyak
- Kinerja : dapat meningkatkan kinerja dengan membagi data dan beban kerja antara beberapa server
- Distribusi data : memungkinkan data diakses dari berbagai lokasi, yang berguna untuk aplikasi global
- Manajemen sumber daya : memungkinkan manajemen sumber daya seperti CPU, memory, dan storage lebih efisien dengan membagi data ke beberapa server