





PART 3 - DATA WAREHOUSE REPLICATE & SHARDING

Perbedaan Replication & Sharding

No	Replication	Sharding
1	<p data-bbox="249 339 1151 549">Proses duplikasi data dari satu database (master) ke beberapa database lainnya. Tujuan utama replikasi adalah meningkatkan ketersediaan data (high availability), failover, dan meningkatkan kinerja baca. Semua salinan data memiliki struktur yang sama.</p> <p data-bbox="249 606 958 685">Data yang sama disalin ke beberapa node Digunakan untuk cadangan/ketersediaan tinggi</p> <div data-bbox="461 796 817 1296"><p data-bbox="542 1253 736 1296">Replication</p></div>	<p data-bbox="1192 339 2430 549">Proses pembagian data secara horizontal di antara beberapa server (shard). Setiap shard menyimpan bagian dari data keseluruhan. Tujuan utama sharding adalah meningkatkan skalabilitas sistem untuk menangani data dalam jumlah besar. Struktur data pada setiap shard bisa berbeda, tergantung pada skema sharding yang digunakan.</p> <p data-bbox="1192 606 2328 728">Data dibagi menjadi beberapa bagian dan didistribusikan ke beberapa node Digunakan untuk pemrosesan/penyimpanan terdistribusi Dapat menyimpan data yang lebih besar dari kapasitas node</p> <div data-bbox="1564 836 1951 1210"></div>

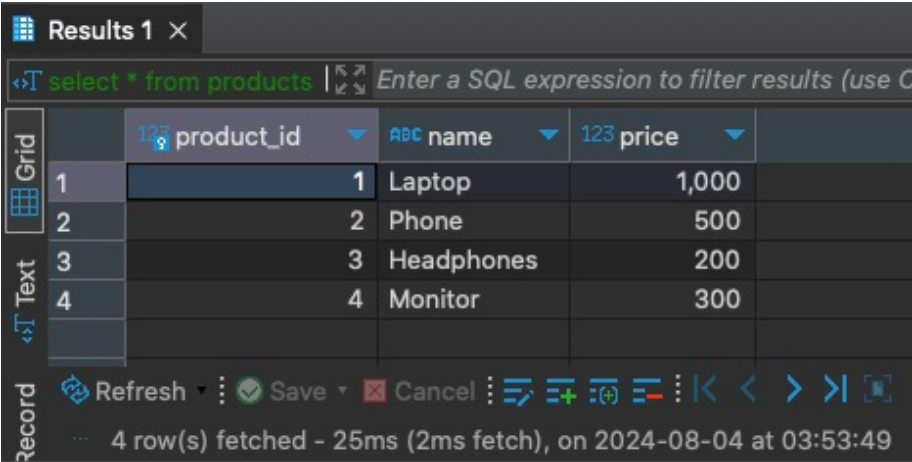
Create Reference Table + Distribute Table



No	Perintah & Deskripsi	Output
2	<p>Lakukan percobaan untuk membuat reference table + distributed table seperti pada repo https://github.com/Immersive-DataEngineer-Resource/citus-demo</p> <div></div> <p>Tabel Users (References)</p> <p>CREATE TABLE users (...): Perintah ini membuat tabel bernama "users" dengan kolom</p> <ul style="list-style-type: none">user_id: Kolom angka berurutan otomatis (SERIAL) sebagai primary key (pengenal unik) tabel.username: Kolom teks yang tidak boleh kosong (NOT NULL).email: Kolom teks yang tidak boleh kosong (NOT NULL) dan unik (UNIQUE) untuk memastikan tidak ada email duplikat. <p>SELECT create_reference_table('users');: Perintah ini (mungkin spesifik untuk Citus) membuat tabel referensi untuk tabel "users". Tabel referensi digunakan untuk manajemen data terdistribusi.</p> <p>INSERT INTO users (...) VALUES ...: Perintah ini memasukkan dua baris data ke dalam tabel "users", masing-masing dengan nama pengguna dan alamat email.</p> <p>select * from users u: Perintah ini menampilkan semua data dari tabel "users" dengan alias "u".</p>	<p>Output menampilkan hasil yang sesuai dengan perintah. Dimana table users sudah dibuat dengan 2 row dengan kolom user_id, username dan email. Dimana Tabel ini digunakan sebagai table references.</p> <div></div>

Create Reference Table + Distribute Table



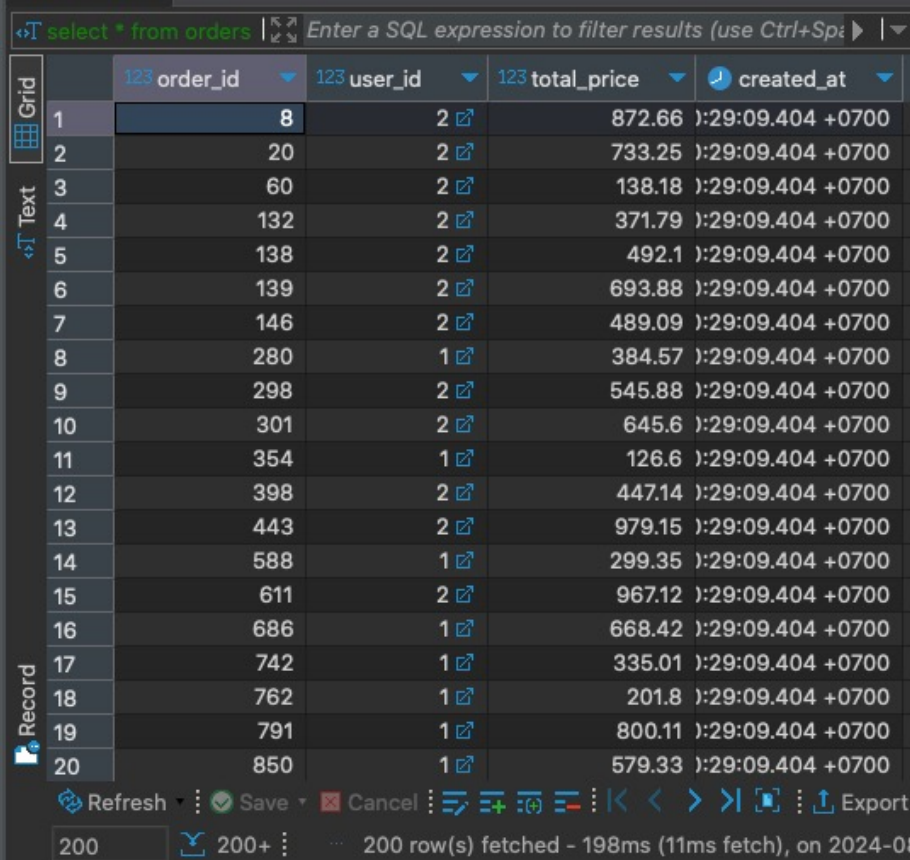
No	Perintah & Deskripsi	Output
2	<p>Lakukan percobaan untuk membuat reference table + distributed table seperti pada repo https://github.com/Immersive-DataEngineer-Resource/citus-demo</p> <pre>-- Create products table (Reference Table) CREATE TABLE products (product_id SERIAL PRIMARY KEY, name TEXT NOT NULL, price NUMERIC(10, 2) NOT NULL); INSERT INTO products (name, price) VALUES ('Laptop', 1000.00), ('Phone', 500.00), ('Headphones', 200.00), ('Monitor', 300.00); select * from products</pre> <p>Tabel Products (Tabel Referensi)</p> <p>CREATE TABLE products (...): Perintah ini membuat tabel bernama "products" dengan kolom:</p> <ul style="list-style-type: none">product_id: Kolom angka berurutan otomatis (SERIAL) sebagai primary key (pengenal unik) tabel.name: Kolom teks yang tidak boleh kosong (NOT NULL).price: Kolom angka desimal yang tidak boleh kosong (NOT NULL) dengan format 10 digit keseluruhan dan 2 digit di belakang koma. <p>INSERT INTO products (...) VALUES ...: Perintah ini memasukkan empat baris data ke dalam tabel "products", masing-masing dengan nama dan harga produk.</p> <p>select * from products: Perintah ini menampilkan semua data dari tabel "products".</p>	<p>Output menampilkan hasil yang sesuai dengan perintah. Dimana table products sudah dibuat dengan 4 row dengan kolom product_id, name dan price. Dimana Tabel ini digunakan sebagai table references.</p> 

Create Reference Table + Distribute Table



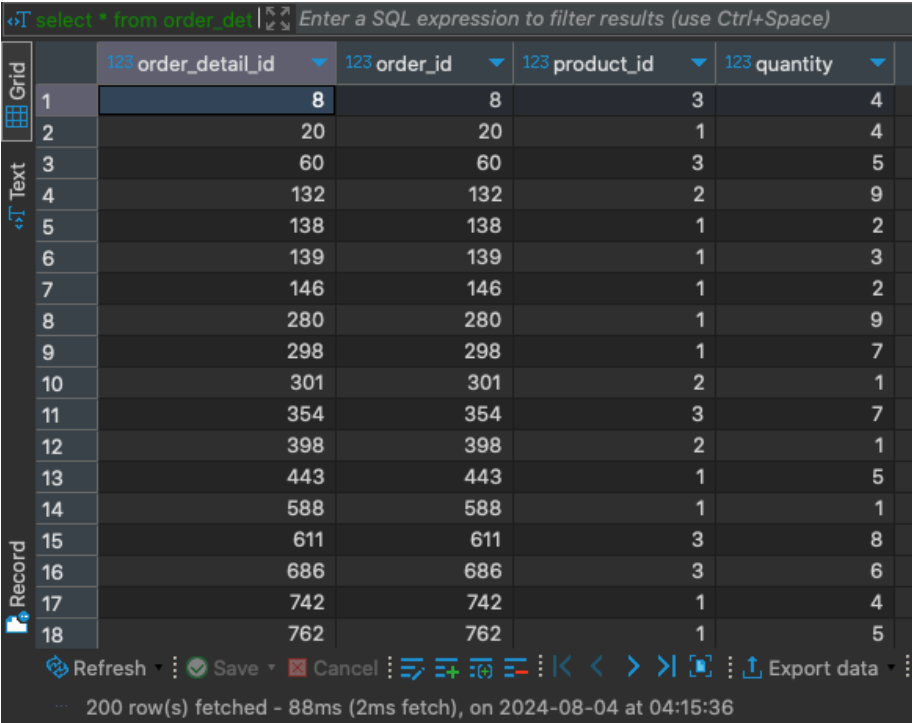
No	Perintah & Deskripsi	Output
2	<p>Lakukan percobaan untuk membuat reference table + distributed table seperti pada repo https://github.com/Immersive-DataEngineer-Resource/citus-demo</p> <pre>CREATE TABLE orders (order_id INT DEFAULT nextval('orders_order_id_seq'), user_id INT REFERENCES users(user_id), total_price NUMERIC(10, 2) NOT NULL, created_at TIMESTAMPTZ DEFAULT NOW()); SELECT create_distributed_table('orders', 'order_id'); select * from orders</pre> <p>Tabel Orders (Tabel Terdistribusi)</p> <p>CREATE SEQUENCE orders_order_id_seq;: Perintah ini membuat sequence (penghasil angka berurutan) bernama "orders_order_id_seq" yang akan digunakan untuk menghasilkan nilai unik untuk order_id.</p> <p>CREATE TABLE orders (...): Perintah ini membuat tabel bernama "orders" dengan kolom:</p> <ul style="list-style-type: none">order_id: Kolom angka yang secara default akan mengambil nilai berikutnya dari sequence "orders_order_id_seq".user_id: Kolom angka yang mereferensi ke kolom "user_id" pada tabel "users" (hubungan antar tabel).total_price: Kolom angka desimal yang tidak boleh kosong (NOT NULL) dengan format 10 digit keseluruhan dan 2 digit di belakang koma.created_at: Kolom tanggal dan waktu yang secara default akan diisi dengan waktu saat ini (NOW()). <p>SELECT create_distributed_table('orders', 'order_id');: Perintah ini (mungkin spesifik untuk Citus) membuat tabel "orders" terdistribusi di beberapa server. Kolom order_id digunakan untuk menentukan distribusi data (sharding) pada server yang berbeda.</p> <p>select * from orders: Perintah ini menampilkan semua data dari tabel "orders".</p>	<p>Hasil dibawah ini disupport dengan input perintah sebuah blok kode PL/pgSQL yang digunakan untuk memasukkan data secara otomatis ke dalam tabel orders.</p>

Create Reference Table + Distribute Table

No	Perintah & Deskripsi	Output
2	<p>Lakukan percobaan untuk membuat reference table + distributed table seperti pada repo https://github.com/Immersive-DataEngineer-Resource/citus-demo</p> <pre>-- Create order_details table CREATE TABLE order_details (order_detail_id INT DEFAULT nextval('order_details_order_detail_id_seq'), order_id INT, product_id INT, quantity INT NOT NULL); SELECT create_distributed_table('order_details', 'order_id'); select * from order_details</pre> <p>Membuat Tabel Detail Pesanan (order_details) CREATE SEQUENCE order_details_order_detail_id_seq:: Membuat urutan angka (sequence) untuk menghasilkan ID detail pesanan secara otomatis. CREATE TABLE order_details (...): Membuat tabel baru bernama "order_details" untuk menyimpan detail pesanan seperti ID detail pesanan, ID pesanan, ID produk, dan jumlah produk yang dipesan. SELECT create_distributed_table('order_details', 'order_id');:: Membuat tabel "order_details" menjadi tabel terdistribusi, menggunakan kolom "order_id" sebagai kunci distribusi.</p>	<p>Hasil dibawah ini disupport dengan input perintah sebuah blok kode PL/pgSQL yang digunakan untuk memasukkan data secara otomatis ke dalam tabel order_details.</p> 

Create Reference Table + Distribute Table

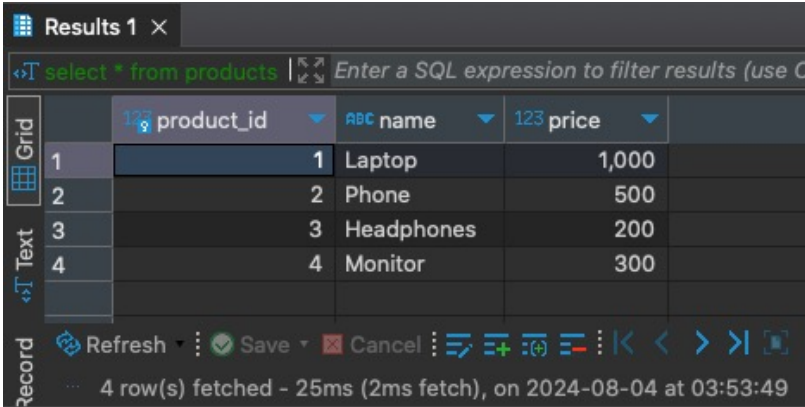
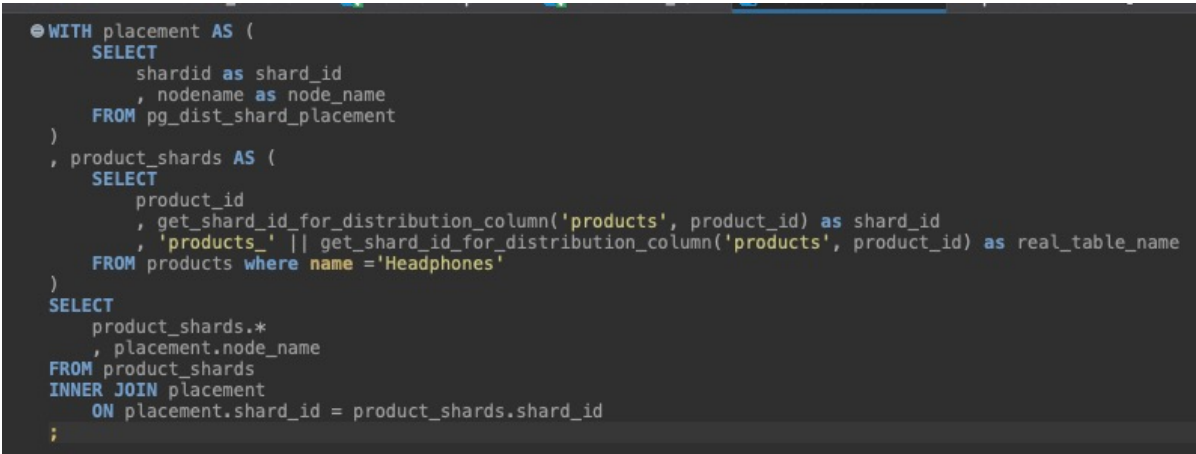


No	Perintah & Deskripsi	Output
2	<p>Lakukan percobaan untuk membuat reference table + distributed table seperti pada repo https://github.com/Immersive-DataEngineer-Resource/citus-demo</p> <pre>-- Create order_details table CREATE TABLE order_details (order_detail_id INT DEFAULT nextval('order_details_order_detail_id_seq'), order_id INT, product_id INT, quantity INT NOT NULL); SELECT create_distributed_table('order_details', 'order_id'); select * from order_details</pre> <p>Membuat Tabel Detail Pesanan (order_details) CREATE SEQUENCE order_details_order_detail_id_seq:: Membuat urutan angka (sequence) untuk menghasilkan ID detail pesanan secara otomatis. CREATE TABLE order_details (...): Membuat tabel baru bernama "order_details" untuk menyimpan detail pesanan seperti ID detail pesanan, ID pesanan, ID produk, dan jumlah produk yang dipesan. SELECT create_distributed_table('order_details', 'order_id'):: Membuat tabel "order_details" menjadi tabel terdistribusi, menggunakan kolom "order_id" sebagai kunci distribusi.</p>	<p>Hasil dibawah ini disupport dengan input perintah sebuah blok kode PL/pgSQL yang digunakan untuk memasukkan data secara otomatis ke dalam tabel order_details.</p> 

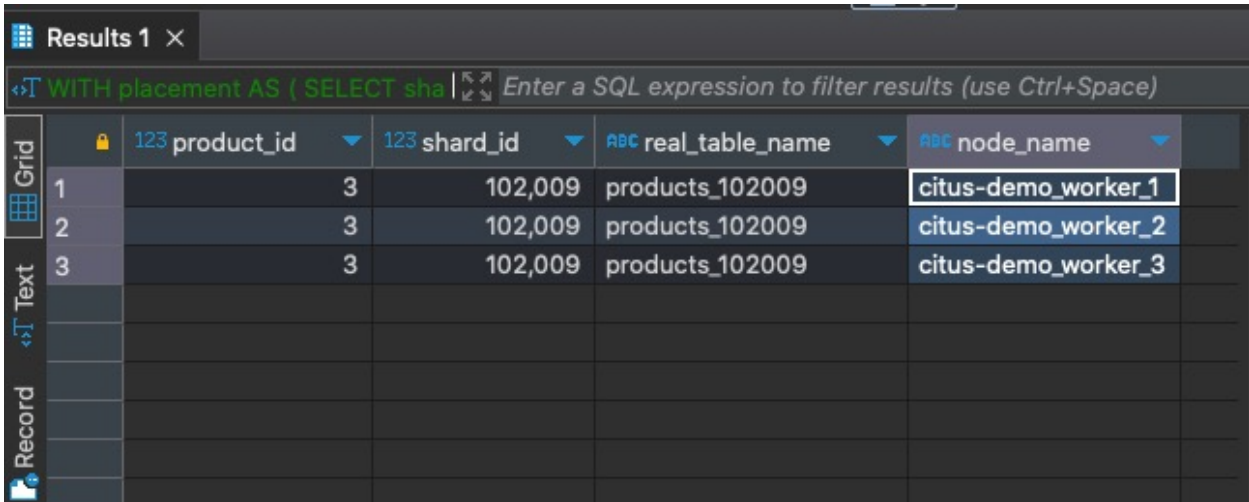
Create Reference Table + Distribute Table

No	Perintah & Deskripsi
2	<p>Lakukan percobaan untuk membuat reference table + distributed table seperti pada repo https://github.com/Immersive-DataEngineer-Resource/citus-demo</p> <p>Kode ini dibawah digunakan untuk menghasilkan 1000 baris data sampel untuk tabel orders dan order_details dengan nilai-nilai acak untuk kolom-kolom tertentu. Ini berguna untuk pengujian atau mengisi data awal dalam database.</p> <pre data-bbox="244 515 2379 852">DO \$\$ DECLARE i INTEGER := 0; BEGIN WHILE i < 1000 LOOP -- insert 1000 rows INSERT INTO orders (user_id, total_price) VALUES (floor(random() * 2 + 1)::INT, random() * 1000); INSERT INTO order_details (order_id, product_id, quantity) VALUES (i + 1, floor(random() * 4 + 1)::INT, floor(random() * 10 + 1)::INT); i := i + 1; END LOOP; END \$\$;</pre> <p>Kode tersebut merupakan sebuah blok kode PL/pgSQL yang digunakan untuk memasukkan data secara otomatis ke dalam tabel orders dan order_details.</p> <p>Langkah-langkah Kode:</p> <p>Deklarasi Blok Kode: DO \$\$ memulai sebuah blok kode anonim dalam PostgreSQL.</p> <p>Deklarasi Variabel: DECLARE i INTEGER := 0; mendeklarasikan sebuah variabel bernama i dengan tipe data integer dan memberikan nilai awal 0.</p> <p>Perulangan: BEGIN WHILE i < 1000 LOOP memulai sebuah perulangan yang akan berulang sebanyak 1000 kali.</p> <p>Memasukkan Data Pesanan: INSERT INTO orders (user_id, total_price) VALUES (floor(random() * 2 + 1)::INT, random() * 1000); memasukkan sebuah baris data ke dalam tabel orders. Nilai user_id adalah bilangan bulat acak antara 1 dan 2, dan total_price adalah bilangan desimal acak antara 0 dan 1000.</p> <p>Memasukkan Data Detail Pesanan: INSERT INTO order_details (order_id, product_id, quantity) VALUES (i + 1, floor(random() * 4 + 1)::INT, floor(random() * 10 + 1)::INT); memasukkan sebuah baris data ke dalam tabel order_details. Nilai order_id adalah nilai i ditambah 1, product_id adalah bilangan bulat acak antara 1 dan 4, dan quantity adalah bilangan bulat acak antara 1 dan 10.</p> <p>Increment Variabel: i := i + 1; meningkatkan nilai i sebesar 1 untuk iterasi berikutnya.</p> <p>Akhir Perulangan: END LOOP; mengakhiri perulangan.</p> <p>Akhir Blok Kode: END \$\$; mengakhiri blok kode anonim.</p>

Di node/worker mana saja product "Headphones" tersimpan?

No	Soal	Output
3	<p>Perlu diketahui bahwa Headphones mempunyai id(product_id) = 3</p>  <p>Berikut merupakan perintah untuk menjawab soal tersebut</p> 	<p>Kode SQL di atas bertujuan untuk mencari tahu di mana data produk "Headphones" disimpan dalam sebuah sistem database terdistribusi. Ini dilakukan dengan menentukan ID shard dan nama node tempat data produk tersebut berada.</p> <p>Breakdown Kode:</p> <p>Tabel Sementara placement:</p> <p>Membuat tabel sementara bernama placement yang berisi informasi tentang ID shard dan nama node untuk setiap shard dalam sistem. Tabel ini diambil dari informasi sistem pg_dist_shard_placement.</p> <p>Tabel Sementara product_shards:</p> <p>Membuat tabel sementara bernama product_shards yang berisi informasi tentang ID produk, ID shard, dan nama tabel fisik untuk produk dengan nama "Headphones". Fungsi get_shard_id_for_distribution_column digunakan untuk menentukan ID shard berdasarkan kolom product_id dalam tabel products.</p> <p>Query Utama:</p> <p>Melakukan join antara tabel sementara product_shards dan placement berdasarkan ID shard untuk mendapatkan informasi lengkap tentang produk "Headphones", termasuk ID produk, ID shard, nama tabel fisik, dan nama node tempat data produk disimpan.</p> <p>Kesimpulan</p> <p>Kode ini akan menghasilkan hasil yang menunjukkan:</p> <p>product_id: ID produk "Headphones".</p> <p>shard_id: ID shard di mana data produk tersebut disimpan.</p> <p>real_table_name: Nama tabel fisik di mana data produk disimpan.</p> <p>node_name: Nama node di mana shard tersebut berada.</p>

Di node/worker mana saja product "Headphone" tersimpan?

No	Output																				
3	<div>Di node/worker mana saja product "Headphone" tersimpan? Tunjukkan shard id nya</div> <div><table><tr><th>Grid</th><th>product_id</th><th>shard_id</th><th>real_table_name</th><th>node_name</th></tr><tr><td>1</td><td>3</td><td>102,009</td><td>products_102009</td><td>citus-demo_worker_1</td></tr><tr><td>2</td><td>3</td><td>102,009</td><td>products_102009</td><td>citus-demo_worker_2</td></tr><tr><td>3</td><td>3</td><td>102,009</td><td>products_102009</td><td>citus-demo_worker_3</td></tr></table></div> <p>product_id: Identifikasi unik untuk sebuah produk. Dalam contoh ini, semua baris memiliki nilai product_id yang sama, yaitu 3. Ini menunjukkan bahwa kita sedang mencari informasi tentang produk dengan ID 3.</p> <p>shard_id: Identifikasi unik untuk sebuah shard (pecahan data) dalam sistem database terdistribusi. Nilai shard_id yang sama untuk semua baris mengindikasikan bahwa data produk dengan ID 3 tersebar di beberapa shard yang sama, yaitu shard dengan ID 102009.</p> <p>real_table_name: Nama tabel fisik di mana data produk disimpan. Dalam hal ini, data produk dengan ID 3 disimpan dalam tabel products_102009.</p> <p>node_name: Nama node (server) di mana shard tersebut berada. Hasil menunjukkan bahwa shard dengan ID 102009 direplikasi di tiga node yang berbeda: citus-demo_worker_1, citus-demo_worker_2, dan citus-demo_worker_3.</p>	Grid	product_id	shard_id	real_table_name	node_name	1	3	102,009	products_102009	citus-demo_worker_1	2	3	102,009	products_102009	citus-demo_worker_2	3	3	102,009	products_102009	citus-demo_worker_3
Grid	product_id	shard_id	real_table_name	node_name																	
1	3	102,009	products_102009	citus-demo_worker_1																	
2	3	102,009	products_102009	citus-demo_worker_2																	
3	3	102,009	products_102009	citus-demo_worker_3																	

Di node/worker mana saja Order id 13 tersimpan?

No	Perintah	Deskripsi
4	<p>Perintah untuk menunjukan di node/worker mana saja order dengan id 13 tersimpan? Tunjukkan shard id nya</p> <pre data-bbox="285 501 1327 1019"> 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 15) , order_shards AS (SELECT order_id , get_shard_id_for_distribution_column('orders', order_id) as shard_id , 'orders_' get_shard_id_for_distribution_column('orders', order_id) as real_table_name FROM order_ids where order_id = 13) SELECT order_shards.* , placement.node_name FROM order_shards INNER JOIN placement ON placement.shard_id = order_shards.shard_id </pre>	<p>Bagian 1: Pendefinisian Tabel Sementara placement AS (SELECT shardid as shard_id, nodename as node_name FROM pg_dist_shard_placement): Membuat tabel sementara bernama placement yang berisi informasi tentang ID shard dan nama node untuk setiap shard dalam sistem. Tabel ini diambil dari informasi sistem pg_dist_shard_placement.</p> <p>Bagian 2: Pendefinisian Tabel Sementara order_ids AS (SELECT order_id FROM orders ORDER BY order_id LIMIT 15): Membuat tabel sementara bernama order_ids yang berisi 15 ID pesanan pertama dari tabel orders.</p> <p>Bagian 3: Pendefinisian Tabel Sementara order_shards AS (SELECT order_id, get_shard_id_for_distribution_column('orders', order_id) as shard_id, 'orders_' get_shard_id_for_distribution_column('orders', order_id) as real_table_name FROM order_ids 1. github.com github.com where order_id = 13): Membuat tabel sementara bernama order_shards yang berisi informasi tentang ID pesanan, ID shard, dan nama tabel fisik untuk pesanan dengan ID 13. Fungsi get_shard_id_for_distribution_column digunakan untuk menentukan ID shard berdasarkan kolom order_id dalam tabel orders.</p> <p>Bagian 4: Query Utama SELECT order_shards.*, placement.node_name FROM order_shards INNER JOIN placement ON placement.shard_id = order_shards.shard_id:: Melakukan join antara tabel sementara order_shards dan placement berdasarkan ID shard untuk mendapatkan informasi lengkap tentang pesanan dengan ID 13, termasuk ID pesanan, ID shard, nama tabel fisik, dan nama node tempat data pesanan disimpan.</p>

Di node/worker mana saja Order id 13 tersimpan?

No

Output

4 Perintah untuk menunjukan di node/worker mana saja order dengan id 13 tersimpan? Tunjukkan shard id nya

Results 1 ×

WITH placement AS (SELECT shardid as s | Enter a SQL expression to filter results (use Ctrl

	123 order_id	123 shard_id	ABC real_table_name	ABC node_name
Grid	1	13	orders_102033	citus-demo_worker_3
Text				
Record				

Refresh Save Cancel

Export data

Dimana dalam kasus ini bahwa Pesanan dengan ID 13 (order_id) disimpan di **shard dengan ID 102033**.

Dimana Data pesanan ini secara real disimpan dalam **tabel orders_102033**.

Lokasi Node: Shard yang berisi pesanan ini berada di node **citus-demo_worker_3**.

Kapan sebaiknya kita menggunakan Replication?

No	Answer
5	<p>Skenario penggunaan antara lain pada saat:</p> <p>Ketersediaan tinggi: Ketika downtime tidak dapat ditoleransi, replikasi memastikan adanya salinan data yang dapat diambil alih jika terjadi kegagalan pada server utama.</p> <p>Baca kinerja tinggi: Dengan banyak salinan data, beban baca dapat didistribusikan ke beberapa server, meningkatkan kinerja query baca.</p> <p>Backup dan recovery: Replikasi dapat digunakan sebagai mekanisme backup untuk memulihkan data jika terjadi kerusakan atau kehilangan data.</p>

Kapan sebaiknya kita menggunakan Sharding?

No	Answer
6	<p>Skalabilitas horizontal: Ketika jumlah data terus meningkat dan satu server tidak lagi mampu menampungnya, sharding memungkinkan pembagian data ke beberapa server.</p> <p>Kinerja tinggi: Dengan membagi data ke beberapa shard, operasi dapat dilakukan secara paralel, meningkatkan kinerja keseluruhan. Pada kasus ini mengacu pada kemampuan sistem database untuk memproses operasi penambahan data (tuliskan) dengan sangat cepat, bahkan ketika jumlah data yang ditangani sangat besar.</p> <p>Analisis data besar: Sharding dapat memudahkan pembagian data untuk analisis paralel menggunakan framework seperti Hadoop atau Spark.</p>

THANK YOU 😊

