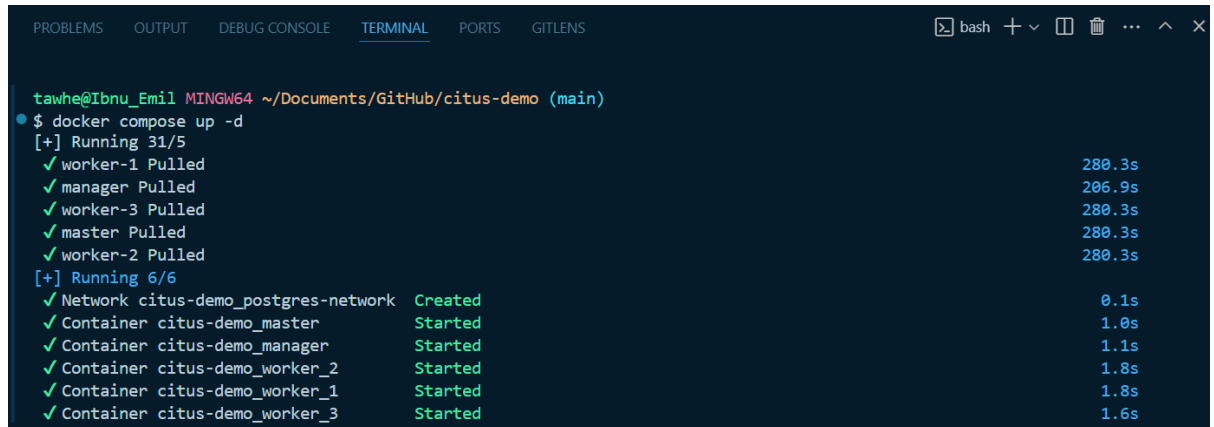


Part 1 – Introduction to Data Warehouse

1. Sebutkan perbedaan antara data warehouse dan data lake!
 - 1) Data Warehouse memiliki data yang terstruktur, dimana data tersebut dioptimalkan untuk query yang kompleks dan analisis data. Biasanya menggunakan skema seperti star atau snowflake schema. Data terstruktur yang sudah diproses dan dibersihkan, biasanya dari sistem OLTP. Tujuan dari Data Warehouse adalah untuk analisis bisnis, pelaporan, business intelligence, dan lain lain. Data Warehouse menggunakan teknologi SQL-based seperti Amazon Redshift, Google BigQuery, dan Snowflake.
 - 2) Data Lake memiliki data yang tidak terstruktur atau semi-terstruktur, dimana data tersebut dioptimalkan untuk penyerapan dan penyimpanan data dalam jumlah besar, dengan analisis yang dilakukan menggunakan alat big data. Untuk sumbernya berasal dari berbagai sumber data seperti log, sensor data, media sosial, dan lain sebagainya. Tujuan dari Data Lake adalah untuk penyimpanan data mentah yang dapat digunakan untuk analisis, machine learning, dan big data. Data Lake menggunakan teknologi Hadoop, Amazon S3, Azure Data Lake.
2. Apa yang membedakan teknologi database untuk data warehouse (OLAP) dari teknologi database konvensional (OLTP)?
 - 1) OLAP (Online Analytical Processing) digunakan untuk analisis bisnis, manajer, dan eksekutif yang bertujuan untuk analisis data historis dan pelaporan. Struktur data dari OLAP (Online Analytical Processing) adalah data terstruktur yang biasanya menggunakan skema star atau snowflake. OLAP (Online Analytical Processing) memiliki query kompleks dan analitik, biasanya melibatkan agregasi dan join. Adapun teknologi yang digunakan pada OLAP (Online Analytical Processing) adalah Amazon Redshift, Google BigQuery, dan Snowflake.
 - 2) OLTP (Online Transaction Processing) digunakan untuk pengguna operasional, maupun karyawan front-end yang bertujuan untuk pemrosesan transaksi harian, seperti insert, update, dan delete. Struktur data dari OLTP (Online Transaction Processing) adalah data terstruktur yang biasanya menggunakan skema normalisasi tinggi. OLTP (Online Transaction Processing) memiliki query sederhana dan cepat, serta . fokus pada transaksi individu. Adapun teknologi yang digunakan pada OLTP (Online Transaction Processing) adalah MySQL, PostgreSQL, dan Oracle Database.
3. Teknologi apa saja yang biasanya dipakai untuk data warehouse?
 - 1) AWS Redshift
 - 2) Google BigQuery
 - 3) Clickhouse
 - 4) Snowflake
 - 5) Databricks
 - 6) Apache Doris
 - 7) Postgre (with citus extension)
 - 8) Microsoft Azure Synapse Analytics
 - 9) Dan lain lain

4. Tuliskan setiap perintah dari proses instalasi citus menggunakan docker compose sampai tabel terbentuk, berikan juga tangkapan layar untuk setiap langkah dan hasilnya!
- 1) Langkah pertama jalankan docker compose pada direktori docker-compose.yml dengan mengetik perintah “docker compose up -d”.



```
tawhe@Ibnu_Emil MINGW64 ~/Documents/GitHub/citus-demo (main)
$ docker compose up -d
[+] Running 31/5
✓ worker-1 Pulled                                280.3s
✓ manager Pulled                                206.9s
✓ worker-3 Pulled                                280.3s
✓ master Pulled                                280.3s
✓ worker-2 Pulled                                280.3s
[+] Running 6/6
✓ Network citus-demo_postgres-network Created    0.1s
✓ Container citus-demo_master Started           1.0s
✓ Container citus-demo_manager Started           1.1s
✓ Container citus-demo_worker_2 Started           1.8s
✓ Container citus-demo_worker_1 Started           1.8s
✓ Container citus-demo_worker_3 Started           1.6s
```

Berikut adalah tampilan bahwa sudah berhasil menjalankan perintah “docker compose up -d”

Containers [Give feedback](#)

Container CPU usage ⓘ

1.10% / 1200% (12 CPUs available)

Container memory usage ⓘ

284.8MB / 7.21GB

Show charts

Search

Only show running containers

	Name	Image	Status	Port(s)	CPU (%)	Last start	Actions
	compose						
	<div><div></div><div>⌵</div><div><div><div></div><div>citus-demo</div></div></div></div>		Running (5/5)		0.94%	1 minute	<div><div></div><div></div><div></div></div>
	<div><div></div><div><div><div></div><div>citus-demo_manager</div></div><div>51f987a95c3c</div></div></div>	citusdata/rmanager:0.51f987a95c3c	Running		0.54%	1 minute	<div><div></div><div></div><div></div></div>
	<div><div></div><div><div><div></div><div>citus-demo_master</div></div><div>789e607f34aa</div></div></div>	citusdata/c789e607f34aa	Running	5431:5432	0.36%	1 minute	<div><div></div><div></div><div></div></div>
	<div><div></div><div><div><div></div><div>citus-demo_worker_2</div></div><div>49b2e768a3b5</div></div></div>	citusdata/c49b2e768a3b5	Running	5434:5432	0.02%	1 minute	<div><div></div><div></div><div></div></div>
	<div><div></div><div><div><div></div><div>citus-demo_worker_3</div></div><div>c09fa558f76b</div></div></div>	citusdata/c09fa558f76b	Running	5435:5432	0.01%	1 minute	<div><div></div><div></div><div></div></div>
	<div><div></div><div><div><div></div><div>citus-demo_worker_1</div></div><div>449e9a5e443e</div></div></div>	citusdata/c449e9a5e443e	Running	5433:5432	0.01%	1 minute	<div><div></div><div></div><div></div></div>

Pada aplikasi Docker juga sudah ada container yang sudah dijalankan tadi. Berikut ini adalah bukti bahwa docker compose sudah berhasil.

- 2) Kemudian, langkah berikutnya adalah membuat connection ke PostgreSQL menggunakan DBeaver dengan konfigurasi seperti di bawah ini :

The screenshot shows the 'Connect to a database' dialog box in DBeaver. The title bar says 'Connect to a database'. The main section is titled 'Connection Settings' and 'PostgreSQL connection settings'. There are tabs for 'Main', 'PostgreSQL Driver properties', 'SSH', and 'SSL'. The 'Main' tab is selected. Under 'Server', 'Connect by:' has 'Host' selected. The 'URL' field contains 'jdbc:postgresql://localhost:5431/store'. The 'Host' field contains 'localhost' and the 'Port' field contains '5431'. The 'Database' field contains 'store'. There is a checkbox for 'Show all databases'. Under 'Authentication', 'Authentication:' is set to 'Database Native'. The 'Username' field contains 'postgres' and the 'Password' field is masked with dots. There is a checkbox for 'Save password'. Under 'Advanced', 'Session role:' is empty and 'Local Client:' is set to 'PostgreSQL 16'. At the bottom, there are links for 'Connection variables information', 'Database documentation', and 'Connection details (name, type, ...)'. There are also buttons for 'Driver name: PostgreSQL', 'Driver Settings', and 'Driver license'. At the very bottom, there are buttons for 'Test Connection ...', '< Back', 'Next >', 'Finish', and 'Cancel'.

The screenshot shows the 'Connection test' dialog box. It has a title bar with the DBeaver logo and the text 'Connection test'. Inside, there is an information icon and the text 'Connected (759 ms)'. Below this, it shows the server details: 'Server: PostgreSQL 15.3 (Debian 15.3-1.pgdg120+1) on x86_64-pc-linux-gnu, compiled by gcc (Debian 12.2.0-14) 12.2.0, 64-bit'. It also shows the driver: 'Driver: PostgreSQL JDBC Driver 42.7.2'. At the bottom, there are two buttons: 'OK' and 'Details >>'.

Berikut merupakan hasil connection test dan connection sudah berhasil dibuat.

3) Buatlah query seperti gambar dibawah ini :

```
create table events_columnar (
  device_id bigint,
  event_id bigserial,
  event_time timestampz default now (),
  data jsonb not null
)
using columnar;

-- insert some data
insert into events_columnar (device_id, data)
select d, '{"hello": "columnar"}' from generate_series(1,100000) d;

-- create a row-based table to compare
create table events_row as select * from events_columnar;
```

Name	Value
Updated Rows	100000
Query	create table events_row as select * from events_columnar
Start time	Wed Aug 07 23:20:18 WIB 2024
Finish time	Wed Aug 07 23:20:19 WIB 2024

Dari gambar diatas, dapat dilihat bahwa query berjalan dengan semestinya.

Tables	
events_columnar	288K
events_row	8.1M

Berikut ini adalah perbedaan ukuran yang sangat signifikan dari table_columnar dan table events_row.

Grd	device_id	event_id	event_time	data
1	1	1	2024-08-07 23:20:14.899 +0700	{"hello": "columnar"}
2	2	2	2024-08-07 23:20:14.899 +0700	{"hello": "columnar"}
3	3	3	2024-08-07 23:20:14.899 +0700	{"hello": "columnar"}
4	4	4	2024-08-07 23:20:14.899 +0700	{"hello": "columnar"}
5	5	5	2024-08-07 23:20:14.899 +0700	{"hello": "columnar"}
6	6	6	2024-08-07 23:20:14.899 +0700	{"hello": "columnar"}
7	7	7	2024-08-07 23:20:14.899 +0700	{"hello": "columnar"}
8	8	8	2024-08-07 23:20:14.899 +0700	{"hello": "columnar"}
9	9	9	2024-08-07 23:20:14.899 +0700	{"hello": "columnar"}
10	10	10	2024-08-07 23:20:14.899 +0700	{"hello": "columnar"}
11	11	11	2024-08-07 23:20:14.899 +0700	{"hello": "columnar"}
12	12	12	2024-08-07 23:20:14.899 +0700	{"hello": "columnar"}
13	13	13	2024-08-07 23:20:14.899 +0700	{"hello": "columnar"}
14	14	14	2024-08-07 23:20:14.899 +0700	{"hello": "columnar"}
15	15	15	2024-08-07 23:20:14.899 +0700	{"hello": "columnar"}
16	16	16	2024-08-07 23:20:14.899 +0700	{"hello": "columnar"}
17	17	17	2024-08-07 23:20:14.899 +0700	{"hello": "columnar"}
18	18	18	2024-08-07 23:20:14.899 +0700	{"hello": "columnar"}
19	19	19	2024-08-07 23:20:14.899 +0700	{"hello": "columnar"}
20	20	20	2024-08-07 23:20:14.899 +0700	{"hello": "columnar"}
21	21	21	2024-08-07 23:20:14.899 +0700	{"hello": "columnar"}
22	22	22	2024-08-07 23:20:14.899 +0700	{"hello": "columnar"}
23	23	23	2024-08-07 23:20:14.899 +0700	{"hello": "columnar"}
24	24	24	2024-08-07 23:20:14.899 +0700	{"hello": "columnar"}
25	25	25	2024-08-07 23:20:14.899 +0700	{"hello": "columnar"}
26	26	26	2024-08-07 23:20:14.899 +0700	{"hello": "columnar"}
27	27	27	2024-08-07 23:20:14.899 +0700	{"hello": "columnar"}
28	28	28	2024-08-07 23:20:14.899 +0700	{"hello": "columnar"}
29	29	29	2024-08-07 23:20:14.899 +0700	{"hello": "columnar"}

1600 row(s) fetched - 0.007s (0.002s fetch), on 2024-08-07 at 23:26:44

Berikut ini adalah tampilan tabel citus yang berhasil dibuat.

5. Jelaskan perbedaan antara access method heap dan columnar pada citus!
- 1) Access Method Heap (Row-Oriented Storage) memiliki kinerja yang optimal untuk query yang mengakses seluruh baris, seperti transaksi per-baris, aplikasi OLTP, dan skenario dengan banyak modifikasi data. Untuk struktur datanya disimpan dalam baris dan memiliki keuntungan untuk transaksi OLTP yang melibatkan banyak insert, update, dan delete.
 - 2) Columnar (Column-Oriented Storage) memiliki kinerja yang optimal untuk operasi baca besar-besaran dan agregasi data seperti analisis data, laporan bisnis, dan skenario dengan akses data terfokus pada beberapa kolom. Untuk struktur datanya disimpan dalam kolom dan memiliki keuntungan untuk query analitik OLAP yang sering melakukan agregasi dan seleksi di sejumlah besar baris tetapi hanya di beberapa kolom.

Jadi kesimpulannya, perbedaan utama adalah pada cara penyimpanan dan optimasi kinerja untuk jenis beban kerja yang berbeda. Jika Access Method Heap (Row-Oriented Storage) untuk transaksi cepat dan modifikasi data, sedangkan jika Columnar (Column-Oriented Storage) untuk analisis data besar dan query yang kompleks.