

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA HỆ THÔNG THÔNG TIN



BÁO CÁO BÀI TẬP LỚN 2 MÔN HỌC
CƠ SỞ DỮ LIỆU PHÂN TÁN

ĐỀ TÀI
CƠ CHẾ PHÂN TÁN TRONG HỆ QUẢN TRỊ CASSANDRA

Giảng viên hướng dẫn:

KS. Nguyễn Minh Nhựt

Nhóm sinh viên thực hiện:

- | | | |
|----|--------------------|----------|
| 1. | Trần Vũ Bão | 22520124 |
| 2. | Phan Thành Công | 22520170 |
| 3. | Phan Thị Thuỷ Hiền | 22520423 |
| 4. | Nguyễn Đỗ Đức Minh | 22520872 |

TP. HỒ CHÍ MINH, NĂM 2025

LỜI CẢM ƠN

Trên thực tế không có sự thành công nào mà không gắn liền với những sự hỗ trợ, giúp đỡ dù ít hay nhiều, dù trực tiếp hay gián tiếp của người khác. Với lòng biết ơn sâu sắc nhất, đầu tiên nhóm chúng em xin gửi lời cảm ơn chân thành đến tập thể quý Thầy Cô Trường Đại học Công nghệ thông tin – Đại học Quốc gia TP.HCM và quý Thầy Cô khoa Hệ thống thông tin đã giúp cho nhóm chúng em có những kiến thức cơ bản làm nền tảng để thực hiện đề tài này.

Đặc biệt nhóm xin gửi lời cảm ơn chân thành tới anh Nguyễn Minh Nhựt - trợ giảng thực hành môn Cơ sở dữ liệu phân tán đã tận tình giúp đỡ, trực tiếp chỉ bảo, hướng dẫn nhóm chúng em trong suốt quá trình làm đồ án môn học. Nhờ đó, chúng em đã tiếp thu được nhiều kiến thức bổ ích trong việc vận dụng cũng như rèn luyện kỹ năng làm đồ án. Nếu không có những lời hướng dẫn, dạy bảo của Anh thì chúng em e rằng đồ án của nhóm rất khó để hoàn thiện được. Một lần nữa, chúng em xin chân thành cảm ơn anh Nhựt. Ngoài ra, để đồ án được hoàn thành thì không thể nào không cảm ơn đến những người đã làm ra nó, cảm ơn các thành viên trong nhóm đã chăm chỉ và chịu khó để hoàn thành nhiệm vụ đúng tiến độ.

Dựa trên những kiến thức được Anh cung cấp trên Trường kết hợp với việc tự tìm hiểu những công cụ và kiến thức mới, chúng em đã cố gắng hết sức để thực hiện đồ án một cách tốt nhất. Tuy nhiên, trong quá trình thực hiện, nhóm chúng em không thể tránh khỏi những thiếu sót. Chính vì vậy, nhóm rất mong nhận được những góp ý từ phía Anh nhằm hoàn thiện hơn những kiến thức mà nhóm đã học tập và tích luỹ được, là hành trang để chúng em thực hiện tiếp các đề tài khác trong tương lai. Cuối cùng, nhóm xin kính chúc anh Nguyễn Minh Nhựt thật dồi dào sức khỏe, vững tin để tiếp tục thực hiện sứ mệnh cao đẹp là truyền đạt kiến thức cho các bạn sinh viên.

Nhóm chúng em xin chân thành cảm ơn!

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

TP. Hồ Chí Minh, tháng 06 năm 2025

Người nhận xét

(Ký tên và ghi rõ họ tên)

MỤC LỤC

DANH MỤC HÌNH ẢNH.....	6
DANH MỤC TỪ VIẾT TẮT.....	9
CHƯƠNG 1. GIỚI THIỆU VỀ HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU NOSQL VÀ CASSANDRA.....	10
1. Tổng quan về NoSQL.....	10
1.1. Khái niệm về NoSQL.....	10
1.2. Đặc điểm.....	10
1.3. Phân loại các hệ cơ sở dữ liệu NoSQL.....	10
2. Giới thiệu về Cassandra.....	11
2.1. Nguồn gốc và lịch sử ra đời của Cassandra.....	11
2.2. Tổng quan về Cassandra.....	11
2.3. Đặc điểm.....	11
2.4. Ưu điểm và nhược điểm của Cassandra.....	13
2.4.1. Ưu điểm.....	13
2.4.2. Nhược điểm.....	13
2.5. Ứng dụng của Cassandra.....	14
3. Giới thiệu về ngôn ngữ truy vấn CQL.....	14
3.1. Kiểu dữ liệu trong CQL.....	14
3.2. Một số hạn chế của CQL.....	16
CHƯƠNG 2. CÀI ĐẶT VÀ KẾT NỐI TRONG CASSANDRA.....	18
1. Cài đặt Cassandra trên từng máy bằng Docker.....	18
1.1. Giới thiệu và các khái niệm cơ bản trong Docker.....	18
1.2. Cài đặt Cassandra bằng Docker.....	19
2. Kết nối hai máy ở hai cluster khác nhau trong Cassandra bằng Python.....	19
2.1. Mã nguồn kết nối.....	20

2.2. Màn hình kết nối thành công giữa hai máy thuộc hai cluster khác nhau.....	22
CHƯƠNG 3. THIẾT KẾ CƠ SỞ DỮ LIỆU NOSQL.....	25
1. Mô tả về cơ sở dữ liệu NoSQL.....	25
2. DDL (Data Definition Language).....	28
3. Dữ liệu mẫu của các bảng trong cơ sở dữ liệu.....	31
3.1. Bảng chi_tiet_hoa_don_theo_ma_kh.....	31
3.2. Bảng doanh_thu_moi_ngay_theo_ma_cn.....	33
3.3. Bảng kho_sp_theo_ma_cn.....	34
3.4. Bảng sl_khach_hang_moi_ngay_theo_ma_cn.....	36
3.5. Bảng doanh_thu_sp_quy_cn.....	37
3.6. Bảng doanh_thu_thang_nv_cn.....	39
CHƯƠNG 4. DI CHUYỂN DỮ LIỆU TỪ CƠ SỞ DỮ LIỆU QUAN HỆ SANG HỆ QUẢN TRỊ CASSANDRA.....	42
1. Quá trình di chuyển dữ liệu từ cơ sở dữ liệu quan hệ sang Cassandra.....	42
1.1. Giới thiệu về công cụ Apache Airflow.....	42
1.2. Di chuyển dữ liệu từ cơ sở dữ liệu quan hệ sang Cassandra.....	46
2. Ưu điểm và nhược điểm của việc di chuyển.....	100
2.1. Ưu điểm.....	100
2.2. Nhược điểm.....	102
3. Thực hiện truy vấn để so sánh hiệu suất truy vấn giữa cơ sở dữ liệu quan hệ và Cassandra.....	105
3.1. Kết quả đối với RDBMS.....	105
3.2. Kết quả đối với Cassandra.....	111
3.3. So sánh và nhận xét.....	115
CHƯƠNG 5. THAO TÁC DỮ LIỆU GIỮA CÁC MÁY.....	119
1. Thêm dữ liệu.....	119

2. Cập nhật dữ liệu (Trong code có áp dụng tính trong suốt phân tán).....	128
3. Xoá dữ liệu (Trong code có áp dụng tính trong suốt phân tán).....	137
4. Thực hiện một số câu truy vấn điển hình giữa hai máy trong Cassandra.....	145
DANH MỤC TÀI LIỆU THAM KHẢO.....	168

DANH MỤC HÌNH ẢNH

Hình 1.1 Kiến trúc của Cassandra.....	13
Hình 2.1 Minh họa về Docker.....	18
Hình 2.2 Bật Radmin VPN ở Chi nhánh 1.....	22
Hình 2.3 Bật Radmin VPN ở Chi nhánh 2.....	23
Hình 2.4 Chi nhánh 1 kết nối thành công đến Chi nhánh 2.....	23
Hình 2.5 Chi nhánh 2 kết nối thành công đến Chi nhánh 1.....	23
Hình 2.6 CN1 thực hiện truy vấn đến CN2.....	24
Hình 2.7 CN2 thực hiện truy vấn đến CN1.....	24
Hình 3.1 Lược đồ cơ sở dữ liệu quản lý cửa hàng bán cây cảnh Plant Paradise.....	25
Hình 3.2 Lược đồ NoSQL.....	28
Hình 3.3 Dữ liệu mẫu ở Oracle của bảng chi_tiet_hoa_don_theo_ma_kh (CN1).....	32
Hình 3.4 Dữ liệu mẫu ở Oracle của bảng chi_tiet_hoa_don_theo_ma_kh (CN2).....	32
Hình 3.5 Dữ liệu mẫu ở Oracle của bảng doanh_thu_moi_ngay_theo_ma_cn (CN1)....	33
Hình 3.6 Dữ liệu mẫu ở Oracle của bảng doanh_thu_moi_ngay_theo_ma_cn (CN2)....	34
Hình 3.7 Dữ liệu mẫu ở Oracle của bảng kho_sp_theo_ma_cn (CN1).....	35
Hình 3.8 Dữ liệu mẫu ở Oracle của bảng kho_sp_theo_ma_cn (CN2).....	35
Hình 3.9 Dữ liệu mẫu ở Oracle của bảng sl_khach_hang_moi_ngay_theo_ma_cn (CN1). 36	
Hình 3.10 Dữ liệu mẫu ở Oracle của bảng sl_khach_hang_moi_ngay_theo_ma_cn (CN2).....	37
Hình 3.11 Dữ liệu mẫu ở Oracle của bảng doanh_thu_sp_quy_cn (CN1).....	38
Hình 3.12 Dữ liệu mẫu ở Oracle của bảng doanh_thu_sp_quy_cn (CN2).....	39
Hình 3.13 Dữ liệu mẫu ở Oracle của bảng doanh_thu_thang_nv_cn (CN1).....	40
Hình 3.14 Dữ liệu mẫu ở Oracle của bảng doanh_thu_thang_nv_cn (CN2).....	41
Hình 4.1 Công cụ Apache Airflow.....	42

Hình 4.2 Quá trình di chuyển dữ liệu từ Oracle sang Cassandra.....	46
Hình 4.3 Cấu trúc thư mục repository di chuyển dữ liệu từ Oracle sang Cassandra.....	47
Hình 4.4 Luồng dữ liệu giữa các thư mục trong repository.....	51
Hình 4.5 Bảng Chi tiết hóa đơn theo khách hàng.....	52
Hình 4.6 Bảng Doanh thu theo chi nhánh.....	52
Hình 4.7 Bảng Kho sản phẩm theo chi nhánh.....	53
Hình 4.8 Bảng Số lượng khách hàng theo ngày.....	53
Hình 4.9 Bảng Doanh thu sản phẩm theo quý.....	54
Hình 4.10 Bảng Doanh thu nhân viên theo tháng.....	55
Hình 4.11 Pipeline toàn bộ quá trình ETL dữ liệu từ Oracle sang Cassandra.....	86
Hình 4.12 Workflow Orchestration và Task Management.....	100
Hình 4.13 Error Handling và Retry Logic.....	101
Hình 4.14 Data Flow Management với XCom.....	101
Hình 4.15 Scheduling và Monitoring.....	101
Hình 4.16 Modular và Reusable Code.....	102
Hình 4.17 XCom Storage Limitation.....	102
Hình 4.18 Resource Management Issues.....	103
Hình 4.19 Development và Debugging Complexity.....	104
Hình 4.20 Performance Bottlenecks.....	104
Hình 4.21 Kết quả EXPLAIN câu truy vấn ở RDBMS (CN1).....	106
Hình 4.22 Kết quả thời gian thực thi câu truy vấn ở RDBMS (CN1).....	108
Hình 4.23 Kết quả EXPLAIN câu truy vấn ở RDBMS (CN2).....	109
Hình 4.24 Kết quả thời gian thực thi câu truy vấn ở RDBMS (CN2).....	111
Hình 4.25 Chạy câu truy vấn ở Cassandra dùng code Python (CN1).....	112
Hình 4.26 Kết quả chạy câu truy vấn ở Cassandra dùng code Python (CN1).....	113
Hình 4.27 Chạy câu truy vấn ở Cassandra dùng code Python (CN2).....	114

Hình 4.28 Kết quả chạy câu truy vấn ở Cassandra dùng code Python (CN2).....	115
Hình 5.1 Terminal CN1 kiểm tra sự tồn tại của doanh thu ngày 2/1/2021.....	119
Hình 5.2 Terminal CN2 kiểm tra sự tồn tại của doanh thu ngày 2/1/2021.....	119
Hình 5.3 Terminal ở CN2 chạy file code Python thực hiện insert.....	122
Hình 5.4 Terminal ở CN1 kiểm tra kết quả insert của CN2.....	123
Hình 5.5 Terminal CN2 kiểm tra tính đúng đắn của thao tác insert.....	123
Hình 5.6 Terminal CN2 kiểm tra sự tồn tại của doanh thu ngày 3/1/2021.....	123
Hình 5.7 Terminal CN1 kiểm tra sự tồn tại của doanh thu ngày 3/1/2021.....	124
Hình 5.8 Terminal ở CN1 chạy file code Python thực hiện insert.....	127
Hình 5.9 Terminal ở CN2 kiểm tra kết quả insert của CN1.....	127
Hình 5.10 Terminal CN1 kiểm tra tính đúng đắn của thao tác insert.....	128
Hình 5.11 Terminal CN1 thực hiện kiểm tra sự tồn tại của dòng dữ liệu.....	131
Hình 5.12 Terminal CN2 thực hiện kiểm tra sự tồn tại của dòng dữ liệu.....	131
Hình 5.13 Terminal CN2 chạy file code Python thực hiện update.....	132
Hình 5.14 Terminal ở CN1 kiểm tra kết quả cập nhật của CN2.....	132
Hình 5.15 Terminal CN2 kiểm tra tính đúng đắn của thao tác update.....	132
Hình 5.16 Terminal CN2 thực hiện kiểm tra sự tồn tại của dòng dữ liệu.....	136
Hình 5.17 Terminal CN1 thực hiện kiểm tra sự tồn tại của dòng dữ liệu.....	136
Hình 5.18 Terminal CN1 chạy file code Python thực hiện update.....	136
Hình 5.19 Terminal ở CN2 kiểm tra kết quả update của CN1.....	136
Hình 5.20 Terminal CN1 kiểm tra tính đúng đắn của thao tác update.....	137
Hình 5.21 Terminal CN1 thực hiện kiểm tra sự tồn tại của dòng dữ liệu.....	140
Hình 5.22 Terminal CN2 thực hiện kiểm tra sự tồn tại của dòng dữ liệu.....	140
Hình 5.23 Terminal CN2 chạy file Python thực hiện delete.....	141
Hình 5.24 Terminal ở CN1 kiểm tra kết quả delete của CN2.....	141
Hình 5.25 Terminal CN2 thực hiện kiểm tra sự tồn tại của dòng dữ liệu.....	144

Hình 5.26 Terminal CN1 thực hiện kiểm tra sự tồn tại của dòng dữ liệu.....	144
Hình 5.27 Terminal CN1 chạy file Python thực hiện delete.....	145
Hình 5.28 Terminal ở CN2 kiểm tra kết quả delete của CN1.....	145
Hình 5.29 Thực thi câu truy vấn 1 (CN1).....	146
Hình 5.30 Thực thi câu truy vấn 1 (CN2).....	146
Hình 5.31 Union kết quả câu truy vấn 1 của CN1 và CN2.....	150
Hình 5.32 Thực thi câu truy vấn 2 (CN1).....	150
Hình 5.33 Thực thi câu truy vấn 2 (CN2).....	150
Hình 5.34 Union kết quả câu truy vấn 2 của CN1 và CN2.....	154
Hình 5.35 Thực thi câu truy vấn 3 (CN1).....	154
Hình 5.36 Thực thi câu truy vấn 3 (CN2).....	154
Hình 5.37 Union kết quả câu truy vấn 3 của CN1 và CN2.....	158
Hình 5.38 Thực thi câu truy vấn 4 (CN1).....	158
Hình 5.39 Thực thi câu truy vấn 4 (CN2).....	158
Hình 5.40 Union kết quả câu truy vấn 4 của CN1 và CN2.....	162
Hình 5.41 Thực thi câu truy vấn 5 (CN1).....	163
Hình 5.42 Thực thi câu truy vấn 5 (CN2).....	163
Hình 5.43 Union kết quả câu truy vấn 5 của CN1 và CN2.....	167

DANH MỤC TỪ VIẾT TẮT

STT	Từ viết tắt	Nội dung
1	CQL	Cassandra Query Language
2	DDL	Data Definition Language
3	NoSQL	Not only SQL (Structure Query Language)
4	RDBMS	Relational Database Management System
5	CSDL	Cơ sở dữ liệu
6	CN1	Chi nhánh 1
7	CN2	Chi nhánh 2
8	ETL	Extract - Transform - Load

CHƯƠNG 1. GIỚI THIỆU VỀ HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU NOSQL VÀ CASSANDRA

1. Tổng quan về NoSQL

1.1. Khái niệm về NoSQL

- NoSQL (Not Only SQL) là một thuật ngữ chỉ các hệ quản trị cơ sở dữ liệu không dùng mô hình quan hệ truyền thống.
- NoSQL được thiết kế để xử lý khối lượng dữ liệu lớn, dữ liệu phi cấu trúc hoặc bán cấu trúc, với khả năng phân tán, mở rộng ngang và độ trễ thấp.
- Thay vì sử dụng bảng (table) như trong RDBMS, NoSQL sử dụng các mô hình linh hoạt hơn như: document, key-value, column-family, graph,...

1.2. Đặc điểm

- Không có lược đồ cố định (schema-less): Dữ liệu có thể thay đổi cấu trúc linh hoạt.
- Hỗ trợ mở rộng ngang (horizontal scalability): Có thể mở rộng bằng cách thêm node thay vì nâng cấp máy chủ.
- Hiệu năng cao: Xử lý nhanh với khối lượng lớn dữ liệu và truy vấn đơn giản.
- Hỗ trợ dữ liệu phi cấu trúc: Ví dụ như JSON, XML,...
- Không tuân thủ chặt chẽ ACID, ưu tiên tính khả dụng (Availability) và phân vùng (Partitioning) — theo CAP theorem.

1.3. Phân loại các hệ cơ sở dữ liệu NoSQL

❖ Key-Value Store:

- Dữ liệu lưu dưới dạng cặp key – value. Giá trị được truy xuất thông qua key.
- Ví dụ: Redis, Riak, Amazon DynamoDB.

❖ Document Store:

- Dữ liệu (bán cấu trúc hay semi-structured) được lưu trữ và tổ chức dưới dạng một tập hợp các document. Các document này linh hoạt, mỗi document có một tập nhiều trường.
- Lưu trữ tài liệu dạng JSON, XML, BSON,...

➤ Ví dụ: MongoDB, CouchDB.

❖ **Column-Family Store (Wide Column Store):**

➤ Cơ sở dữ liệu tổ chức dưới dạng các bảng. Gần giống với mô hình RDBMS. Tuy nhiên, Chúng lưu trữ dữ liệu bởi các cột chứ không phải bằng các dòng

➤ Ví dụ: Apache Cassandra, HBase.

❖ **Graph Database:**

➤ Những CSDL này áp dụng lý thuyết đồ thị trong khoa học máy tính để lưu trữ và truy xuất dữ liệu. Chúng tập trung vào tính rời rạc giữa các phần dữ liệu. Các phần tử đơn vị dữ liệu được biểu thị như một nút (node) và liên kết với các thành phần khác bằng các cạnh (edge).

➤ Ví dụ: Neo4j, ArangoDB.

2. Giới thiệu về Cassandra

2.1. Nguồn gốc và lịch sử ra đời của Cassandra

- Được phát triển ban đầu bởi Facebook năm 2007 để phục vụ tính năng Inbox Search.
- Được công bố mã nguồn mở vào năm 2008.
- Năm 2009, dự án Cassandra được Apache Software Foundation tiếp nhận và phát triển dưới tên Apache Cassandra.
- Cassandra kết hợp mô hình dữ liệu Bigtable (Google) với Dynamo (Amazon)
→ vừa có khả năng lưu trữ theo cột, vừa có khả năng phân tán mạnh mẽ.

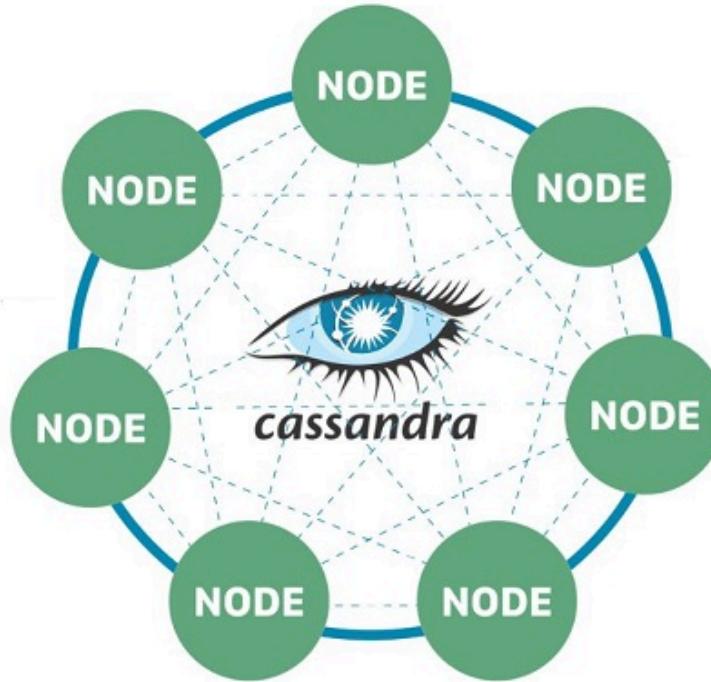
2.2. Tổng quan về Cassandra

- Apache Cassandra là một hệ quản trị cơ sở dữ liệu NoSQL phân tán, phi quan hệ, hướng cột.
- Hỗ trợ mô hình dữ liệu phi tập trung với kiến trúc ngang hàng (peer-to-peer).
- Tối ưu cho hệ thống yêu cầu khả năng mở rộng cao, sẵn sàng cao, và đảm bảo tính nhất quán eventual consistency.

2.3. Đặc điểm

Cassandra cung cấp các tính năng sau:

- *Khả năng mở rộng (Massively Scalable Architecture)*: Cassandra có một thiết kế vô cùng đặc biệt. Cụm Cassandra có thể dễ dàng thu nhỏ hoặc mở rộng. Bất kỳ số lượng các nút có thể được thêm vào hoặc xóa đi trong cụm Cassandra mà không gây ra xáo trộn nào.
- *Kiến trúc ngang hàng (Peer to peer)*: Cassandra theo một kiến trúc ngang hàng, thay vì kiến trúc client-server. Các nút trong Cassandra có vai trò tương tự nhau, đều đảm nhận việc đọc ghi dữ liệu, làm giảm nguy cơ bị bottleneck (thắt cổ chai). Do đó, không có bất cứ điểm chết nào.
- *Masterless Architecture*: Dữ liệu có thể được ghi và đọc trên bất kỳ nút nào.
- *Hiệu suất quy mô tuyến tính (Linear scale performance)*: Khi nhiều nút được thêm vào, hiệu suất của Cassandra sẽ tăng lên.
- *Kiến trúc không có SPOF (No Single Point of Failure)*: Cassandra sao chép dữ liệu trên các nút khác nhau để đảm bảo không có một điểm lỗi nào.
- *Phát hiện và khôi phục lỗi (Fault detection and recovery)*: Các nút bị lỗi có thể dễ dàng được khôi phục và phục hồi.
- *Mô hình dữ liệu linh hoạt và năng động (Flexible and dynamic data model)*: Hỗ trợ các kiểu dữ liệu ghi và đọc nhanh.
- *Bảo vệ dữ liệu (Data protection)*: Dữ liệu được bảo vệ với thiết kế nhật ký cam kết và xây dựng trong bảo mật như như các cơ chế sao lưu và khôi phục.
- *Tính nhất quán dữ liệu (Tunable data consistency)*: Hỗ trợ tính nhất quán dữ liệu mạnh mẽ trên toàn bộ phân phối kiến trúc.
- *Sao chép đa trung tâm dữ liệu (Multi-Data Center Replication)*: Cassandra cung cấp sao chép dữ liệu trên nhiều trung tâm.
- *Nén dữ liệu (Data Compression)*: Cassandra có thể nén tới 80% dữ liệu mà không cần bất kỳ chi phí nào.
- *Ngôn ngữ truy vấn Cassandra (CQL)*: Cassandra cung cấp một ngôn ngữ truy vấn tương tự ngôn ngữ SQL. Điều này giúp các nhà phát triển chuyển từ cơ sở dữ liệu quan hệ sang dùng Cassandra một cách dễ dàng.



Hình 1.1 Kiến trúc của Cassandra

2.4. Ưu điểm và nhược điểm của Cassandra

2.4.1. Ưu điểm

- Khả năng mở rộng tốt: Mỗi node có vai trò ngang nhau, dễ dàng mở rộng bằng cách thêm node mới.
- Hiệu năng cao: Thích hợp cho ứng dụng ghi nhiều.
- Đảm bảo tính sẵn sàng (Availability): Dữ liệu được sao chép nhiều nơi.
- Hỗ trợ replication theo chiến lược tùy chỉnh.
- Không bị lỗi khi một node ngừng hoạt động.
- Khả năng xử lý big data tốt.
- Hỗ trợ ngôn ngữ truy vấn CQL (Cassandra Query Language) gần giống SQL, dễ tiếp cận.

2.4.2. Nhược điểm

- Cassandra không hỗ trợ nhiều cho việc tính toán trên storage, nó không hỗ trợ các hàm JOIN, UNION, INTERSECT,... Tuy nhiên vẫn có hỗ

trợ các Aggregate functions (COUNT, MIN, MAX, SUM, AVG), ORDER, LIMIT.

- Không hỗ trợ giao dịch phức tạp (ACID) như trong các cơ sở dữ liệu quan hệ.
- Khó thiết kế mô hình dữ liệu tối ưu nếu chưa quen.
- Không hỗ trợ tính nhất quán mạnh (strong consistency) mặc định.
- Tối ưu cho việc ghi và đọc nhanh, nhưng đối với các truy vấn phức tạp thì có thể kém hiệu quả.

2.5. Ứng dụng của Cassandra

- *Mạng xã hội*: Lưu trữ dữ liệu người dùng, bài đăng, tương tác (Facebook, Instagram).
- *Thương mại điện tử*: Dữ liệu khách hàng, đơn hàng, phân tích hành vi.
- *Internet of Things (IoT)*: Lưu trữ luồng dữ liệu cảm biến theo thời gian thực.
- *Streaming và phân tích dữ liệu lớn*: Hệ thống log, clickstream, dữ liệu từ sensors.
- *Hệ thống recommendation*: Dựa trên dữ liệu người dùng và hành vi truy cập.

3. Giới thiệu về ngôn ngữ truy vấn CQL

3.1. Kiểu dữ liệu trong CQL

Tên kiểu dữ liệu	Định nghĩa
ascii	Biểu diễn cho một chuỗi ký tự ASCII.
bigint	Đại diện cho số nguyên có dấu dài 64-bit.
blob	Dùng để lưu trữ các byte tùy ý.
boolean	Lưu trữ true hoặc false.

counter	Đại diện cho một số nguyên dài 64-bit, nhưng giá trị của cột này không thể thiết lập. Chỉ có hai hoạt động trên kiểu này là tăng và giảm.
date	Đại diện cho một giá trị ngày mà không có một giá trị giờ. Ngày có thể được biểu diễn như là chuỗi trong định dạng yyyy-mm-dd.
decimal	Đại diện cho một biến - giá trị thập phân.
double	Lưu trữ một giá trị dấu chấm động dài 64-bit.
float	Lưu trữ một giá trị dấu chấm động 32-bit.
inet	Biểu diễn cho một chuỗi địa chỉ IP trong định dạng của IPv4 hoặc IPv6.
int	Biểu diễn cho một số nguyên có dấu dài 32-bit. Sử dụng chủ yếu để lưu trữ các giá trị số nguyên.
smallint	Biểu diễn cho một số nguyên 2 byte (16-bit).
text	Biểu diễn cho một chuỗi mã hoá UTF-8.
time	Biểu diễn cho một giá trị thời gian.
timestamp	Lưu trữ cả thành phần ngày và giờ với độ chính xác mili giây.
tinyint	Biểu diễn cho một số nguyên 1 byte (8 bit).
timeuuid	Lưu trữ phiên bản 1 UUID.

uuid	UUID ở định dạng chuẩn. Đây là một giá trị lớn hơn so với timeuuid.
varchar	Tương tự như text.
variant	Một giá trị số nguyên với độ chính xác tùy ý.
set	Kiểu này lưu trữ một bộ các giá trị. Các giá trị được lưu trữ không có thứ tự, nhưng CQLSH sẽ trả về dữ liệu đã được sắp xếp.
list	Một danh sách cũng lưu trữ một bộ các giá trị, nhưng lưu trữ chúng theo kiểu đã được sắp xếp, mặc định sắp theo thứ tự chèn vào.
map	Chứa một bộ các cặp khóa-giá trị ở bất cứ kiểu gì, ngoại trừ kiểu counter.

3.2. Một số hạn chế của CQL

- Không hỗ trợ JOIN và các truy vấn con (Subqueries) như trong SQL chuẩn. Mỗi mệnh đề SELECT chỉ được áp dụng trên một bảng.
- Không hỗ trợ khóa OR trong mệnh đề WHERE. Để xử lý trường hợp tương tự thì buộc phải tách truy vấn thành nhiều câu SELECT riêng biệt và kết hợp kết quả ở phía ứng dụng (client side).
- Chỉ các cột được định nghĩa là partition key hoặc clustering column mới được phép truy vấn trực tiếp. Nếu muốn lọc dữ liệu bằng một cột không thuộc khóa chính thì ta cần:
 - + Tạo secondary index (Secondary Index chỉ nên dùng khi dữ liệu phân tán đều và bảng không quá lớn).
 - + Hoặc là dùng ALLOW FILTERING (*rất không khuyến khích vì hiệu năng thấp*).

-- KHÔNG được nếu 'email' không phải key

```
SELECT * FROM users WHERE email = 'abc@example.com';
```

-- Cách 1: Tạo secondary index

```
CREATE INDEX ON users (email);
```

-- Cách 2 (không khuyến khích):

```
SELECT * FROM users WHERE email = 'abc@example.com' ALLOW  
FILTERING;
```

- Trong Cassandra, chỉ có thể sử dụng toán tử so sánh (>, <, >=, <=) trên cột clustering, và chỉ sau khi đã lọc theo partition key.

-- Ví dụ đúng: giả sử bảng có PRIMARY KEY (user_id, timestamp)

```
SELECT * FROM logs WHERE user_id = '123' AND timestamp >  
'2024-01-01';
```

--Ví dụ sai (timestamp không phải clustering key hoặc user_id không có):

```
SELECT * FROM logs WHERE timestamp > '2024-01-01';
```

- Không có Foreign key hay Constraint giữa các bảng.
- Không có Aggregate functions phức tạp ngoại trừ các hàm COUNT, SUM, AVG, MIN, MAX.
- Không hỗ trợ Trigger hoặc Procedure phức tạp.

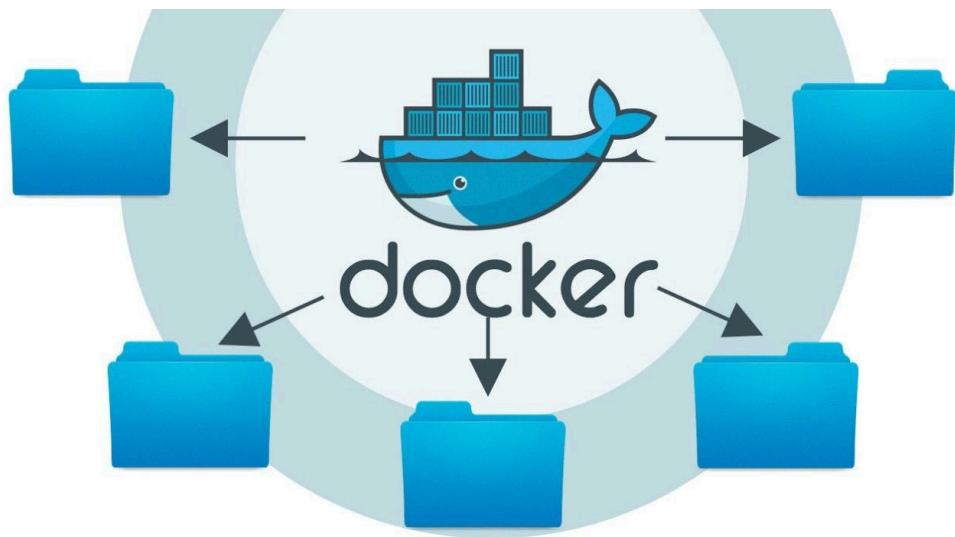
CHƯƠNG 2. CÀI ĐẶT VÀ KẾT NỐI TRONG CASSANDRA

1. Cài đặt Cassandra trên từng máy bằng Docker

1.1. Giới thiệu và các khái niệm cơ bản trong Docker

- **Docker là gì?**

- Docker là một nền tảng mã nguồn mở giúp đóng gói, phân phối, và chạy ứng dụng trong các container - một môi trường nhẹ, độc lập, và nhất quán giữa các hệ thống.
- Container hoạt động giống như một máy ảo nhẹ, nhưng nhanh hơn và hiệu quả hơn.



Hình 2.1 Minh họa về Docker

- **Lợi ích của Docker:**

- Triển khai ứng dụng nhanh chóng.
- Đảm bảo môi trường phát triển giống nhau giữa các máy.
- Tối ưu hóa tài nguyên máy tính.
- Dễ dàng tích hợp DevOps (CI/CD).

- **Các khái niệm cơ bản trong Docker:**

Khái niệm	Mô tả
-----------	-------

Image	Mẫu để tạo container. Ví dụ: cassandra:latest
Container	Phiên bản đang chạy của image.
Dockerfile	File chứa hướng dẫn để build image.
Docker Hub	Kho lưu trữ public các image (giống GitHub nhưng cho container).
Volume	Lưu trữ dữ liệu để container restart không mất dữ liệu.
Port mapping	Chuyển tiếp cổng từ máy chủ vào container (ví dụ -p 9042:9042).

1.2. Cài đặt Cassandra bằng Docker

- Trong terminal của Docker, tiến hành gõ lệnh:

```
docker pull cassandra
```

Lệnh này sẽ tải image chính thức của Cassandra từ Docker Hub về máy.

- Khởi chạy container Cassandra:

```
docker run --name cass_cluster -p 9042:9042 -d cassandra:latest
```

Giải thích:

- + --name cass_cluster: đặt tên container là cass_cluster
- + -p 9042:9042: ánh xạ cổng mặc định để truy cập CQL
- + -d: chạy container ở chế độ nền
- + cassandra:latest: sử dụng image mới tải

Nếu thấy dòng “*Starting listening for CQL clients on /0.0.0.0:9042*” thì Cassandra đã sẵn sàng.

- Mở giao diện dòng lệnh CQL (cqlsh):

```
docker exec -it cass_cluster cqlsh
```

Lúc này sẽ vào môi trường cqlsh bên trong container để truy vấn Cassandra.

2. Kết nối hai máy ở hai cluster khác nhau trong Cassandra bằng Python

2.1. Mã nguồn kết nối

File connect_2_clusters.py

```
from cassandra.cluster import Cluster
from cassandra.auth import PlainTextAuthProvider
import socket

def test_connection(host, port):
    try:
        sock = socket.create_connection((host, port), timeout=5)
        sock.close()
        return True
    except Exception as e:
        return f"Failed: {e}"

def connect_to_cluster(cluster_ip, keyspace_name):
    # auth_provider = PlainTextAuthProvider(username='congphan',
    # password='password')
    try:
        # Test connection to teammate's machine
        print(test_connection(cluster_ip, 9042))

        cluster = Cluster(
            contact_points=[cluster_ip],
            # auth_provider=auth_provider,
            port=9042,
            protocol_version=4,
            connect_timeout=10, # seconds
            control_connection_timeout=10 # seconds
    )
```

```

        session = cluster.connect()
        print(f"Connected to cluster having {cluster_ip} successfully!")

        print("\nAttempting to connect to BTL2_data keyspace...")
        session = cluster.connect(keyspace_name)
        print(f"Connected to {keyspace_name} successfully!")

    return cluster1, session
except Exception as e:
    print(f"Connecting to another cluster failed: {e}")

```

File cluster_queries.py

```

from connect_2_clusters import connect_to_cluster

cluster_ip = '26.103.246.194'
keyspace_name = 'btl2_data'

try:
    cluster, session = connect_to_cluster(cluster_ip, keyspace_name)
    print("\n=====Kiểm tra kết nối 2 máy thuộc 2 cluster=====")

    query = "SELECT * FROM doanh_thu_moi_ngay_theo_ma_cn LIMIT 5;"
    print(f"\nMáy 2 đang thực hiện câu truy vấn: {query}")
    print("\n")

    rows = session.execute(query)
    if not rows:

```

```

        print("No results found")

    else:
        for row in rows:
            print(row)

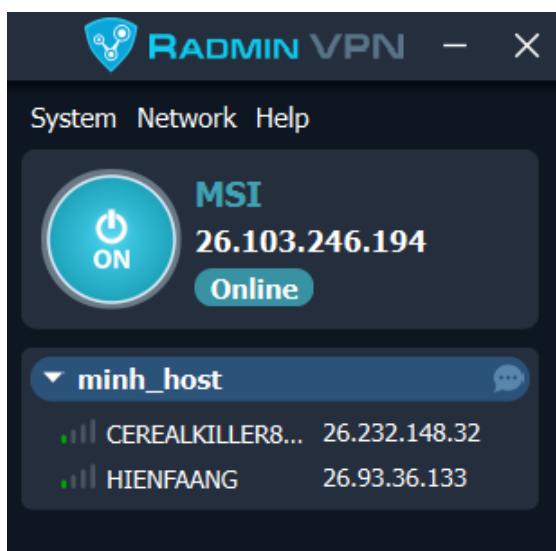
except Exception as e:
    print(f"Error when doing queries: {e}")

finally:
    # Always close connections
    if 'cluster' in locals():
        cluster.shutdown()

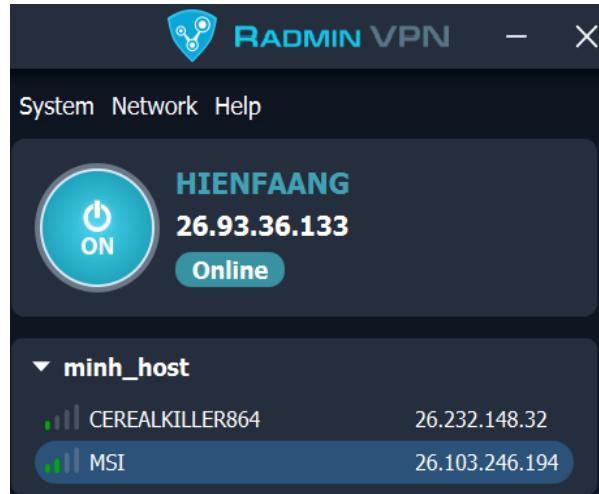
```

2.2. Màn hình kết nối thành công giữa hai máy thuộc hai cluster khác nhau

- Bật Radmin VPN để hai máy vào chung một mạng ảo “*minh_host*”.
 - + Máy 1 (Chi nhánh 1) có IP là 26.103.246.194
 - + Máy 2 (Chi nhánh 2) có IP là 26.93.36.133



Hình 2.2 Bật Radmin VPN ở Chi nhánh 1



Hình 2.3 Bật Radmin VPN ở Chi nhánh 2

- Máy 1 (Chi nhánh 1).

```
PS D:\Nam3_HK2\CoSoDuLieuPhanTan\DoAn\BTL2\migrate_oracle_cassandra> python .\cluster_queries.py
True
Connected to cluster_2 having IP address: 26.93.36.133 successfully!

Attempting to connect to BTL2_data keyspace...
Connected to btl2_data successfully!
```

Hình 2.4 Chi nhánh 1 kết nối thành công đến Chi nhánh 2

- Máy 2 (Chi nhánh 2).

```
pwsh python3 cluster_queries.py
True
Connected to cluster_1 having IP address: 26.103.246.194 successfully!

Attempting to connect to BTL2_data keyspace...
Connected to btl2_data successfully!
```

Hình 2.5 Chi nhánh 2 kết nối thành công đến Chi nhánh 1

- Kiểm tra việc thực hiện truy vấn giữa hai máy sau khi đã kết nối.
 - + Kết quả ở Máy 1.

```
D:\Nam3_HK2\CoSoDuLieuPhanTan\DoAn\BTL2\migrate_oracle_cassandra\cassandra>python cluster_queries.py
True
Connected to cluster_2 having IP address: 26.93.36.133 successfully!
Attempting to connect to BTL2_data keyspace...
Connected to btl2_data successfully!
Máy 1 đang thực hiện câu truy vấn: SELECT * FROM doanh_thu_moi_ngay_theo_ma_cn LIMIT 5;

Row(ma_chi_nhanh=2, ngay=Date(20453), tong_tien=2309644808)
Row(ma_chi_nhanh=2, ngay=Date(20452), tong_tien=2385423517)
Row(ma_chi_nhanh=2, ngay=Date(20451), tong_tien=2313346178)
Row(ma_chi_nhanh=2, ngay=Date(20450), tong_tien=2401552275)
Row(ma_chi_nhanh=2, ngay=Date(20449), tong_tien=2122912265)
```

Hình 2.6 CN1 thực hiện truy vấn đến CN2

+ Kết quả ở Máy 2.

```
pwsh python3 cluster_queries.py
True
Connected to cluster_1 having IP address: 26.103.246.194 successfully!
Attempting to connect to BTL2_data keyspace...
Connected to btl2_data successfully!
=====Kiểm tra kết nối 2 máy thuộc 2 cluster=====
Máy 2 đang thực hiện câu truy vấn: SELECT * FROM doanh_thu_moi_ngay_theo_ma_cn LIMIT 5;

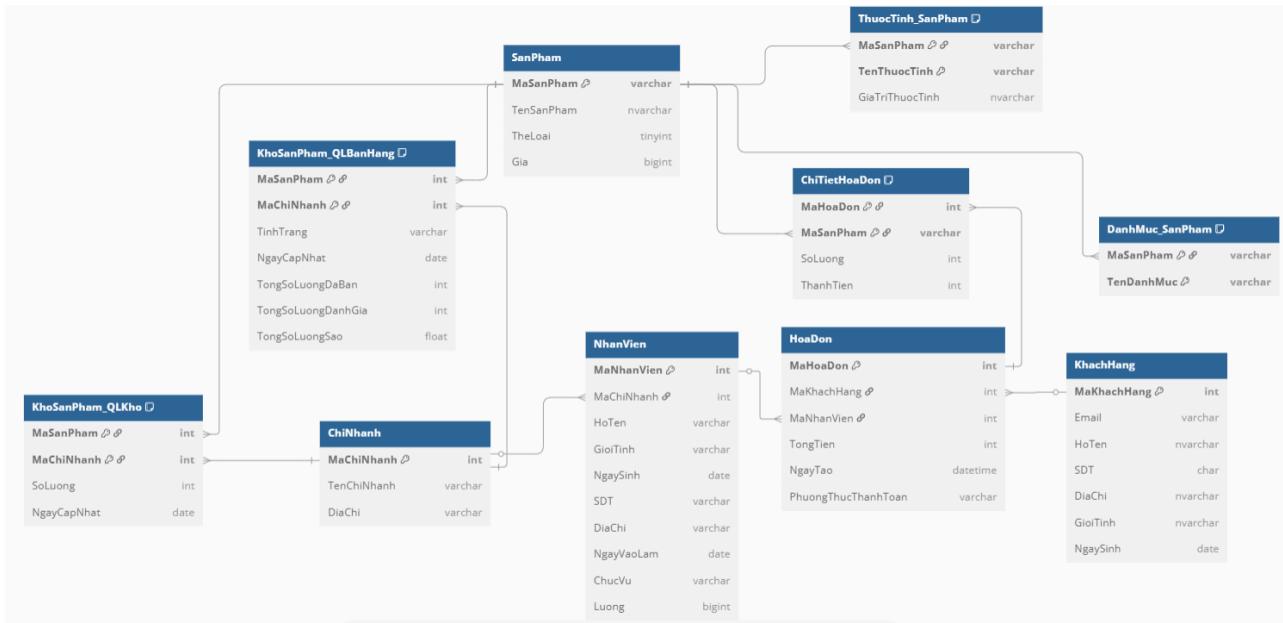
Row(ma_chi_nhanh=1, ngay=Date(20453), tong_tien=2352262351)
Row(ma_chi_nhanh=1, ngay=Date(20452), tong_tien=2452166036)
Row(ma_chi_nhanh=1, ngay=Date(20451), tong_tien=2326607572)
Row(ma_chi_nhanh=1, ngay=Date(20450), tong_tien=2494039167)
Row(ma_chi_nhanh=1, ngay=Date(20449), tong_tien=2120817846)
```

Hình 2.7 CN2 thực hiện truy vấn đến CN1

CHƯƠNG 3. THIẾT KẾ CƠ SỞ DỮ LIỆU NOSQL

1. Mô tả về cơ sở dữ liệu NoSQL

- Xuất phát từ lược đồ cơ sở dữ liệu quan hệ.



Hình 3.1 Lược đồ cơ sở dữ liệu quản lý cửa hàng bán cây cảnh Plant Paradise

Lược đồ CSDL quản lý cửa hàng bán cây cảnh Plant Paradise như sau:

CHINHANH (MACHINHANH, TENCHINHANH, DIACHI)

Tân từ: Mỗi chi nhánh có mã chi nhánh (MACHINHANH) dùng để phân biệt các chi nhánh với nhau, ngoài ra còn có tên chi nhánh (TENCHINHANH) và địa chỉ của chi nhánh đó (DIACHI). Địa chỉ chi nhánh sẽ là một trong ba thành phố: Hà Nội, Đà Nẵng hoặc TPHCM.

KHACHHANG (MAKHACHHANG, EMAIL, HOTEN, SDT, DIACHI, GIOITINH, NGAYSINH)

Tân từ: Mỗi khách hàng có mã khách hàng (MAKHACHHANG) dùng để định danh. Các thông tin lưu kèm bao gồm email (EMAIL), họ tên (HOTEN), số điện thoại (SDT), địa chỉ (DIACHI), giới tính (GIOITINH) và ngày sinh (NGAYSINH).

NHANVIEN (MANHANVIEN, MACHINHANH, HOTEN, GIOITINH, NGAYSINH, SDT, DIACHI, NGAYVAOLAM, CHUCVU, LUONG)

Tân từ: Mỗi nhân viên có mã nhân viên (MANHANVIEN) để phân biệt. Mỗi nhân viên làm việc tại một chi nhánh (MACHINHANH) cụ thể, kèm theo các thông tin cá nhân như họ tên (HOTEN), giới tính (GIOITINH), ngày sinh (NGAYSINH), số điện thoại (SDT), địa chỉ (DIACHI), ngày vào làm (NGAYVAOLAM), chức vụ (CHUCVU) và mức lương (LUONG).

SANPHAM (MASANPHAM, TENSANPHAM, THELOAI, GIA)

Tân từ: Mỗi sản phẩm có mã sản phẩm (MASANPHAM) duy nhất để phân biệt. Các thông tin kèm theo gồm tên sản phẩm (TENSANPHAM), thể loại sản phẩm (THELOAI) với giá trị mặc định là 0 đối với sản phẩm là *cây* và 1 đối với sản phẩm là *chậu*, và giá bán (GIA).

THUOCTINH_SANPHAM (MASANPHAM, TENTHUOCTINH, GIATRITHUOCTINH)

Tân từ: Bảng này lưu trữ các thuộc tính chi tiết của sản phẩm, bao gồm mã sản phẩm (MASANPHAM), tên thuộc tính (TENTHUOCTINH) và giá trị của thuộc tính đó (GIATRITHUOCTINH). Mỗi sản phẩm có thể có nhiều thuộc tính khác nhau.

DANHMUC_SANPHAM (MASANPHAM, TENDANHMUC)

Tân từ: Bảng này liên kết sản phẩm với các danh mục phân loại, gồm mã sản phẩm (MASANPHAM) và tên danh mục (TENDANHMUC). Một sản phẩm có thể thuộc nhiều danh mục.

HOADON (MAHOADON, MAKHACHHANG, MANHANVIEN, TONGTIEN, NGAYTAO, PHUONGTHUCTHANHTOAN)

Tân từ: Mỗi hóa đơn có mã hóa đơn (MAHOADON) để phân biệt. Thông tin hóa đơn bao gồm mã khách hàng (MAKHACHHANG), mã nhân viên lập hóa đơn (MANHANVIEN),

tổng tiền hóa đơn (TONGTIEN), ngày tạo hóa đơn (NGAYTAO), và phương thức thanh toán (PHUONGTHUC THANHTOAN).

CHITIETHOADON (MAHOADON, MASANPHAM, SOLUONG, THANHTIEN)

Tân từ: Bảng chi tiết hóa đơn ghi lại các sản phẩm được bán trong từng hóa đơn, gồm mã hóa đơn (MAHOADON), mã sản phẩm (MASANPHAM), số lượng sản phẩm bán ra (SOLUONG) và thành tiền tương ứng (THANHTIEN).

KHOSANPHAM_QLKHO (MASANPHAM, MACHINHANH, SOLUONG, NGAYCAPNHAT)

Tân từ: Bảng này quản lý việc nhập kho cũng như số lượng tồn kho của sản phẩm tại từng chi nhánh, gồm mã sản phẩm (MASANPHAM), mã chi nhánh (MACHINHANH), số lượng tồn (SOLUONG) và ngày cập nhật tồn kho (NGAYCAPNHAT).

KHOSANPHAM_QLBANHANG (MASANPHAM, MACHINHANH, TINHTRANG, NGAYCAPNHAT, TONGSOLUONGDABAN, TONGSOLUONGDANHGIA, TONGSOLUONGSAO)

Tân từ: Bảng này quản lý tình trạng bán hàng của sản phẩm tại các chi nhánh, gồm mã sản phẩm (MASANPHAM), mã chi nhánh (MACHINHANH), tình trạng còn hàng hoặc hết hàng (TINHTRANG), ngày cập nhật (NGAYCAPNHAT), tổng số lượng sản phẩm đã bán (TONGSOLUONGDABAN), tổng số lượng đánh giá sản phẩm (TONGSOLUONGDANHGIA) và tổng số sao đánh giá trung bình (TONGSOLUONGSAO).

- ❖ Thiết kế schema trong Cassandra theo hướng **Query-driven** – nghĩa là thiết kế bảng theo truy vấn.

doanh_thu_moi_ngay_theo_ma_cn		sl_khach_hang_moi_ngay_theo_ma_cn	
ma_chi_nhanh	int	ma_chi_nhanh	int
ngay	date	ngay	date
tong_tien	bigint	so_luong_khach_hang	int
chi_tiet_hoa_don_theo_ma_kh		doanh_thu_thang_nv_cn	
ma_khach_hang	int	ma_chi_nhanh	int
ma_hoa_don	int	ma_nhan_vien	int
ma_san_pham	text	nam	int
so_luong	int	thang	int
thanh_tien	bigint	tong_doanh_thu	bigint
tong_tien	bigint		
ngay_tao	timestamp		
phuong_thuc_thanh_toan	text		
ma_nhan_vien	text		
doanh_thu_sp_quy_cn		kho_sp_theo_ma_cn	
ma_chi_nhanh	int	ma_chi_nhanh	int
ma_san_pham	text	ma_san_pham	text
nam	int	ten_san_pham	text
quy	int	tinh_trang	text
tong_doanh_thu	bigint	tong_sl_danh_gia	int
		tong_sl_da_ban	int
		tong_sl_ton_kho	int

Hình 3.2 Lược đồ NoSQL

2. DDL (Data Definition Language)

```
CREATE KEYSPACE IF NOT EXISTS BTL2_data
```

```
WITH REPLICATION = {
    'class': 'SimpleStrategy',
    'replication_factor': 1
};
```

```
USE BTL2_data;
```

```
CREATE TABLE IF NOT EXISTS chi_tiet_hoa_don_theo_ma_kh (
    ma_khach_hang int,
    ma_hoa_don int,
    ma_san_pham text,
    so_luong int,
    thanh_tien bigint,
    tong_tien bigint,
    ngay_tao timestamp,
    phuong_thuc_thanh_toan text,
    ma_nhan_vien int,
    PRIMARY KEY ((ma_khach_hang), ngay_tao, ma_hoa_don)
) WITH CLUSTERING ORDER BY (ngay_tao DESC);
```

```
CREATE TABLE IF NOT EXISTS doanh_thu_moi_ngay_theo_ma_cn (
    ma_chi_nhanh int,
    ngay date,
    tong_tien bigint,
    PRIMARY KEY ((ma_chi_nhanh), ngay)
) WITH CLUSTERING ORDER BY (ngay DESC);
```

```
CREATE TABLE IF NOT EXISTS kho_sp_theo_ma_cn (
    ma_chi_nhanh int,
    ma_san_pham text,
    ten_san_pham text,
    tinh_trang text,
    tong_so_luong_danh_gia int,
    tong_so_luong_da_ban int,
    tong_so_luong_ton_kho int,
    PRIMARY KEY ((ma_chi_nhanh), ma_san_pham, tong_so_luong_ton_kho)
```

```
);
```

```
CREATE TABLE IF NOT EXISTS sl_khach_hang_moi_ngay_theo_ma_cn (
    ma_chi_nhanh int,
    ngay date,
    so_luong_khach_hang int,
    PRIMARY KEY ((ma_chi_nhanh), ngay)
) WITH CLUSTERING ORDER BY (ngay DESC);
```

```
CREATE TABLE doanh_thu_sp_quy_cn (
    ma_chi_nhanh int,
    ma_san_pham text,
    nam int,
    quy int,
    tong_doanh_thu bigint,
    PRIMARY KEY ((ma_chi_nhanh, ma_san_pham), nam, quy)
) WITH CLUSTERING ORDER BY (nam ASC, quy ASC);
```

```
CREATE TABLE doanh_thu_thang_nv_cn (
    ma_chi_nhanh int,
    ma_nhan_vien int,
    nam int,
    thang int,
    tong_doanh_thu bigint,
    PRIMARY KEY ((ma_chi_nhanh, ma_nhan_vien), nam, thang)
) WITH CLUSTERING ORDER BY (nam ASC, thang ASC);
```

Giải thích về chiến lược nhân bản được sử dụng:

- 'class': 'SimpleStrategy'

- Chiến lược nhân bản SimpleStrategy: Đây là chiến lược nhân bản đơn giản nhất trong Cassandra.

- Cách hoạt động: SimpleStrategy được thiết kế để sử dụng cho một trung tâm dữ liệu (single data center) duy nhất và các cụm thử nghiệm/phát triển. Nó không nhận biết về cấu trúc topo phức tạp như các trung tâm dữ liệu hay rack khác nhau. Khi sử dụng SimpleStrategy, bản sao đầu tiên được đặt trên node do Partitioner xác định. Các bản sao tiếp theo được đặt trên các node tiếp theo theo chiều kim đồng hồ trên vòng (ring) mà không quan tâm đến vị trí vật lý (rack hay data center) của các node đó.
- **'replication_factor': 1**
- Hệ số nhân bản (Replication Factor): Tham số này xác định tổng số bản sao của mỗi dòng dữ liệu sẽ được lưu trữ trong cụm Cassandra.
 - Giá trị 1: Với replication_factor: 1, chỉ có một bản sao duy nhất của mỗi dòng dữ liệu được lưu trữ trên toàn bộ cụm.

3. Dữ liệu mẫu của các bảng trong cơ sở dữ liệu

Để thêm dữ liệu vào các bảng đã được định nghĩa trong schema của CSDL Cassandra, tiến hành viết các truy vấn SQL để trích xuất dữ liệu từ CSDL Oracle, chuẩn bị cho quá trình đổ dữ liệu (Sử dụng công cụ ETL Apache Airflow - sẽ được trình bày chi tiết trong chương 4) vào các bảng trong Cassandra.

3.1. Bảng chi_tiet_hoa_don_theo_ma_kh

```
SELECT
    hd."MaKhachHang",
    cthd."MaHoaDon",
    cthd."MaSanPham",
    cthd."SoLuong",
    cthd."ThanhTien",
    hd."TongTien",
    hd."NgayTao",
    hd."PhuongThucThanhToan",
    hd."MaNhanVien"
```

```

FROM BTL1."ChiTietHoaDon" cthd
JOIN BTL1."HoaDon" hd ON cthd."MaHoaDon" = hd."MaHoaDon"
ORDER BY BTL1."MaHoaDon", BTL1."NgayTao" DESC;

```

- Dữ liệu mẫu:

MaKhachHang NUMBER(38)	MaHoaDon NUMBER(38)	MaSanPham NVARCHAR2(50)	SoLuong NUMBER(38)	ThanhTien NUMBER(38)	TongTien NUMBER(38)	NgayTao DATE	PhuongThucThanhToan NVARCHAR2(50)	MaNhanVien NUMBER(38)
1	3	C CNPPLT0347	3	1260000	2830000	07/22/ 2025 05:47:01 AM	Ngân Hàng	398
1	3	C CNPPLT0317	7	1120000	2830000	07/22/ 2025 05:47:01 AM	Ngân Hàng	398
1	3	C CNPPLT0307	30	450000	2830000	07/22/ 2025 05:47:01 AM	Ngân Hàng	398
1	4	C CNPPLT0412	10	1600000	1600000	12/14/ 2023 09:59:20 AM	Ngân Hàng	182
1	4	C CNPPLT0307	12	123000	1600000	12/14/ 2023 09:59:20 AM	Ngân Hàng	182
1	9	C CNPPLT0307	1	123000	1660000	05/12/ 2023 09:54:48 PM	Tiền Mặt	108
1	12	C CNPPLT0186	1	180000	7190000	03/24/ 2023 05:22:37 PM	Ngân Hàng	137
1	12	C CNPPLT0052	8	2960000	7190000	03/24/ 2023 05:22:37 PM	Ngân Hàng	137
1	12	C CNPPLT0397	3	2850000	7190000	03/24/ 2023 05:22:37 PM	Ngân Hàng	137
1	12	C CNPPLT0645	4	480000	7190000	03/24/ 2023 05:22:37 PM	Ngân Hàng	137
1	12	C CNPOT0159	6	720000	7190000	03/24/ 2023 05:22:37 PM	Ngân Hàng	137
1	13	C CNPPLT0410	8	1440000	18990000	07/12/ 2025 05:31:40 AM	Tiền Mặt	296
1	13	C CNPPLT0405	1	550000	18990000	07/12/ 2025 05:31:40 AM	Tiền Mặt	296
1	13	C CNPPLT0213	9	3240000	18990000	07/12/ 2025 05:31:40 AM	Tiền Mặt	296
1	13	C CNPPLT0402	9	4680000	18990000	07/12/ 2025 05:31:40 AM	Tiền Mặt	296

Hình 3.3 Dữ liệu mẫu ở Oracle của bảng chi_tiet_hoa_don_theo_ma_kh (CN1)

MaKhachHang NUMBER(38)	MaHoaDon NUMBER(38)	MaSanPham NVARCHAR2(50)	SoLuong NUMBER(38)	ThanhTien NUMBER(38)	TongTien NUMBER(38)	NgayTao DATE	PhuongThucThanhToan NVARCHAR2(50)	MaNhanVien NUMBER(38)
1	2	C CNPPLT0113	9	1260000	1420000	03/29/ 2025 01:10:10 AM	Ngân Hàng	537
1	2	C CNPPLT0365	1	160000	1420000	03/29/ 2025 01:10:10 AM	Ngân Hàng	537
1	8	C CNPPLT0591	2	180000	18595000	03/26/ 2023 06:19:48 PM	Tiền Mặt	695
1	8	C CNPPLT0196	9	1620000	18595000	03/26/ 2023 06:19:48 PM	Tiền Mặt	695
1	8	C CNPPLT0454	5	7750000	18595000	03/26/ 2023 06:19:48 PM	Tiền Mặt	695
1	8	C CNPOT0021	9	8775000	18595000	03/26/ 2023 06:19:48 PM	Tiền Mặt	695
1	8	C CNPPLT0563	9	270000	18595000	03/26/ 2023 06:19:48 PM	Tiền Mặt	695
1	10	C CNPPLT0303	1	120000	14850000	10/23/ 2022 02:47:32 PM	Ngân Hàng	532
1	10	C CNPPLT0064	4	8800000	14850000	10/23/ 2022 02:47:32 PM	Ngân Hàng	532
1	10	C CNPPLT0262	10	1850000	14850000	10/23/ 2022 02:47:32 PM	Ngân Hàng	532
1	10	C CNPPLT0646	8	1280000	14850000	10/23/ 2022 02:47:32 PM	Ngân Hàng	532
1	10	C CNPOT0076	2	2800000	14850000	10/23/ 2022 02:47:32 PM	Ngân Hàng	532
1	11	C CNPOT0180	6	1500000	7950000	11/02/ 2025 06:45:06 PM	Ngân Hàng	940
1	11	C CNPPLT0097	1	220000	7950000	11/02/ 2025 06:45:06 PM	Ngân Hàng	940
1	11	C CNPOT0008	5	2250000	7950000	11/02/ 2025 06:45:06 PM	Ngân Hàng	940
1	11	C CNPPLT0626	7	1400000	7950000	11/02/ 2025 06:45:06 PM	Ngân Hàng	940
1	11	C CNPPLT0232	1	180000	7950000	11/02/ 2025 06:45:06 PM	Ngân Hàng	940
1	11	C CNPPLT0215	5	2400000	7950000	11/02/ 2025 06:45:06 PM	Ngân Hàng	940
1	17	C CNPPLT0502	7	1260000	4465000	09/25/ 2025 05:00:40 PM	Ngân Hàng	528
1	17	C CNPPLT0474	4	240000	4465000	09/25/ 2025 05:00:40 PM	Ngân Hàng	528
1	17	C CNPPLT0372	1	125000	4465000	09/25/ 2025 05:00:40 PM	Ngân Hàng	528
1	17	C CNPPLT0150	8	2800000	4465000	09/25/ 2025 05:00:40 PM	Ngân Hàng	528
1	17	C CNPPLT0604	2	40000	4465000	09/25/ 2025 05:00:40 PM	Ngân Hàng	528
1	21	C CNPPLT0675	9	2250000	2910000	04/09/ 2025 02:19:19 AM	Tiền Mặt	855
1	21	C CNPPLT0300	3	660000	2910000	04/09/ 2025 02:19:19 AM	Tiền Mặt	855

Hình 3.4 Dữ liệu mẫu ở Oracle của bảng chi_tiet_hoa_don_theo_ma_kh (CN2)

3.2. Bảng doanh_thu_moi_ngay_theo_ma_cn

```
SELECT  
    cn."MaChiNhanh",  
    TRUNC(hd."NgayTao") AS Ngay,  
    SUM(hd."TongTien") AS TongTien  
FROM "HoaDon" hd  
    RIGHT JOIN "NhanVien" nv ON hd."MaNhanVien" = nv."MaNhanVien"  
    JOIN "ChiNhanh" cn ON cn."MaChiNhanh" = nv."MaChiNhanh"  
GROUP BY cn."MaChiNhanh", TRUNC(hd."NgayTao")  
ORDER BY Ngay DESC;
```

- Dữ liệu mẫu:

MaChiNhanh NUMBER(38)	NGAY DATE	TONGTIEN NUMBER
1	12/31/ 2025 12:00:00 AM	2352262351
1	12/30/ 2025 12:00:00 AM	2452166036
1	12/29/ 2025 12:00:00 AM	2326607572
1	12/28/ 2025 12:00:00 AM	2494039167
1	12/27/ 2025 12:00:00 AM	2120817846
1	12/26/ 2025 12:00:00 AM	2598029503
1	12/25/ 2025 12:00:00 AM	2409097008
1	12/24/ 2025 12:00:00 AM	2513628818
1	12/23/ 2025 12:00:00 AM	2384233355
1	12/22/ 2025 12:00:00 AM	2307391615
1	12/21/ 2025 12:00:00 AM	2400590748
1	12/20/ 2025 12:00:00 AM	2584688258
1	12/19/ 2025 12:00:00 AM	2111992851
1	12/18/ 2025 12:00:00 AM	2385967147
1	12/17/ 2025 12:00:00 AM	1941671765
1	12/16/ 2025 12:00:00 AM	2726351883
1	12/15/ 2025 12:00:00 AM	2182630066

Hình 3.5 Dữ liệu mẫu ở Oracle của bảng doanh_thu_moi_ngay_theo_ma_cn (CN1)

MaChiNhanh NUMBER(38)	NGAY DATE	TONGTIEN NUMBER
2	01/01/ 2022 12:00:00 AM	2474411570
2	01/02/ 2022 12:00:00 AM	2212405946
2	01/03/ 2022 12:00:00 AM	2553255865
2	01/04/ 2022 12:00:00 AM	2201154496
2	01/05/ 2022 12:00:00 AM	2289334331
2	01/06/ 2022 12:00:00 AM	2400397617
2	01/07/ 2022 12:00:00 AM	2331795915
2	01/08/ 2022 12:00:00 AM	2353894903
2	01/09/ 2022 12:00:00 AM	2658002156
2	01/10/ 2022 12:00:00 AM	2208021188
2	01/11/ 2022 12:00:00 AM	2271402375
2	01/12/ 2022 12:00:00 AM	2506278383
2	01/13/ 2022 12:00:00 AM	2561088882
2	01/14/ 2022 12:00:00 AM	2357290247
2	01/15/ 2022 12:00:00 AM	1973712056
2	01/16/ 2022 12:00:00 AM	2087217512
2	01/17/ 2022 12:00:00 AM	2466577369
2	01/18/ 2022 12:00:00 AM	2301329159
2	01/19/ 2022 12:00:00 AM	2232012262
2	01/20/ 2022 12:00:00 AM	2277265970
2	01/21/ 2022 12:00:00 AM	2068016092
2	01/22/ 2022 12:00:00 AM	2389749225
2	01/23/ 2022 12:00:00 AM	2408868090
2	01/24/ 2022 12:00:00 AM	2573833551
2	01/25/ 2022 12:00:00 AM	2104931043

Hình 3.6 Dữ liệu mẫu ở Oracle của bảng doanh_thu_moi_ngay_theo_ma_cn (CN2)

3.3. Bảng kho_sp_theo_ma_cn

```

SELECT
KSPQ."MaChiNhanh",
KSPQ."MaSanPham",
SP."TenSanPham",
KSPQH."TinhTrang",
KSPQH."TongSoLuongDanhGia",
KSPQH."TongSoLuongDaBan",
KSPQ."SoLuong"
FROM "KhoSanPham_QLBanHang" kspqh, "KhoSanPham_QLKho" kspq, "SanPham"
sp

```

```

WHERE KSPQ."MaSanPham" = KSPQH."MaSanPham" AND SP."MaSanPham" =
KSPQ."MaSanPham"

ORDER BY KSPQ."MaSanPham", KSPQ."SoLuong";

```

- Dữ liệu mẫu:

MaChiNhanh NUMBER(38)	MaSanPham NVARCHAR2(50)	TenSanPham NVARCHAR2(100)	TinhTrang NVARCHAR2(50)	TongSoLuongDanhGia NUMBER(38)	TongSoLuongDaBan NUMBER(38)	SoLuong NUMBER(38)
1	C CNPLT0079	Cây cầu nguyên (màu nhạt)	Hết hàng	46	83	0
1	C CNPLT0080	Cây cầu nguyên (nhieu rẽ)	Còn hàng	96	231	97
1	C CNPLT0081	Cây cầu nguyên (compact)	Còn hàng	10	171	21
1	C CNPLT0082	Cây cầu nguyên (giống nhập)	Hết hàng	50	308	0
1	C CNPLT0083	Cây chuối mỏ két đỗ / Thiên điểu cầm thạch	Hết hàng	48	18	0
1	C CNPLT0085	Cây cỏ lan chi / cây dây nhện (đất thịt)	Hết hàng	35	21	0
1	C CNPLT0086	Cây cỏ lan chi / cây dây nhện (cao)	Còn hàng	66	411	71
1	C CNPLT0087	Cây cỏ lan chi / cây dây nhện (dáng xòe)	Còn hàng	18	42	9
1	C CNPLT0089	Cây cọ Nhật / Kè Nhật	Còn hàng	10	6	34
1	C CNPLT0091	Cây cỏ tông đuôi lươn	Hết hàng	70	199	0
1	C CNPLT0092	Cây cỏ tông lá mít	Hết hàng	23	395	0
1	C CNPLT0093	Cây cung đàn xanh	Còn hàng	99	5	14
1	C CNPLT0094	Cây cung điện vàng (tán tròn)	Còn hàng	34	28	44
1	C CNPLT0096	Cây cung điện vàng (bụi to)	Còn hàng	21	253	13
1	C CNPLT0097	Cây Cung Điện Vàng (lá đốm)	Còn hàng	32	432	1
1	C CNPLT0098	Cây đại lộc / vạn lộc / ngọc ngân / đỗ la	Còn hàng	96	7	53
1	C CNPLT0099	Cây đại phú gia / Đỗ la / May mắn	Còn hàng	34	144	53

Hình 3.7 Dữ liệu mẫu ở Oracle của bảng kho_sp_theo_ma_cn (CN1)

MaChiNhanh NUMBER(38)	MaSanPham NVARCHAR2(50)	TenSanPham NVARCHAR2(100)	TinhTrang NVARCHAR2(50)	TongSoLuongDanhGia NUMBER(38)	TongSoLuongDaBan NUMBER(38)	SoLuong NUMBER(38)
2	C CNPLT0001	Bạch mã hoàng tử (trồng trong nhà)	Hết hàng	73	316	0
2	C CNPLT0003	Bàng cầm thạch (nhiều lá)	Còn hàng	52	471	52
2	C CNPLT0004	Bàng cầm thạch (nhỏ gọn)	Còn hàng	80	210	99
2	C CNPLT0006	Bàng Đài Loan cầm thạch (ấm ướt)	Còn hàng	28	211	74
2	C CNPLT0007	Bàng Đài Loan cầm thạch (có quả)	Hết hàng	41	465	0
2	C CNPLT0008	Bàng Đài Loan cầm thạch (chậu treo)	Hết hàng	89	108	0
2	C CNPLT0009	Bàng Đài Loan cầm thạch (trồng chậu treo)	Còn hàng	44	20	34
2	C CNPLT0010	Bàng Đài Loan cầm thạch (tươi tốt)	Hết hàng	68	42	0
2	C CNPLT0012	Bàng Singapore / Sung tỳ bà (chậu nhựa)	Hết hàng	82	371	0
2	C CNPLT0013	Bàng Singapore / Sung tỳ bà (giống nhập)	Còn hàng	65	462	40
2	C CNPLT0014	Bàng Singapore / Sung tỳ bà (đất sạch)	Hết hàng	65	423	0
2	C CNPLT0015	Bàng Singapore / Sung tỳ bà (lá dài)	Hết hàng	20	299	0
2	C CNPLT0016	Bàng Singapore / Sung tỳ bà (ngắn ngày)	Còn hàng	82	99	85
2	C CNPLT0017	Bàng Singapore / Sung tỳ bà (tán rộng)	Còn hàng	91	221	44
2	C CNPLT0019	Bàng Singapore / Sung tỳ bà (tươi)	Còn hàng	16	36	99
2	C CNPLT0020	Bàng Singapore / Sung tỳ bà (lâu năm)	Còn hàng	40	195	96
2	C CNPLT0022	Bàng Singapore / Sung tỳ bà (có quả)	Còn hàng	47	498	46
2	C CNPLT0024	Bàng Singapore / Sung tỳ bà (tươi tốt)	Còn hàng	28	422	81
2	C CNPLT0025	Bàng singapore / Sung tỳ bà (lá xoắn)	Hết hàng	84	55	0
2	C CNPLT0026	Bàng singapore / Sung tỳ bà (dáng tròn)	Còn hàng	98	251	32
2	C CNPLT0027	Bàng Singapore / Sung tỳ bà (ít sâu bệnh)	Còn hàng	55	396	87
2	C CNPLT0029	Caladium Aaron	Còn hàng	22	485	67
2	C CNPLT0033	Cánh bướm xanh	Còn hàng	45	455	82
2	C CNPLT0034	Cau Hawaii (thân cao)	Còn hàng	61	27	69
2	C CNPLT0035	Cau Hawaii (uña nắng)	Hết hàng	100	443	0

Hình 3.8 Dữ liệu mẫu ở Oracle của bảng kho_sp_theo_ma_cn (CN2)

3.4. Bảng sl_khach_hang_moi_ngay_theo_ma_cn

```
SELECT  
    NV."MaChiNhanh",  
    TRUNC(HD."NgayTao") AS Ngay,  
    COUNT(DISTINCT HD."MaKhachHang") AS SoLuongKhachHang  
FROM "NhanVien" NV  
JOIN "HoaDon" HD ON NV."MaNhanVien" = HD."MaNhanVien"  
GROUP BY NV."MaChiNhanh", TRUNC(HD."NgayTao")  
ORDER BY Ngay DESC;
```

- Dữ liệu mẫu:

MaChiNhanh NUMBER(38)	NGAY DATE	SOLUONGKHACHHANG NUMBER
1	12/31/ 2025 12:00:00 AM	309
1	12/30/ 2025 12:00:00 AM	306
1	12/29/ 2025 12:00:00 AM	316
1	12/28/ 2025 12:00:00 AM	358
1	12/27/ 2025 12:00:00 AM	311
1	12/26/ 2025 12:00:00 AM	335
1	12/25/ 2025 12:00:00 AM	327
1	12/24/ 2025 12:00:00 AM	307
1	12/23/ 2025 12:00:00 AM	304
1	12/22/ 2025 12:00:00 AM	333
1	12/21/ 2025 12:00:00 AM	313
1	12/20/ 2025 12:00:00 AM	324
1	12/19/ 2025 12:00:00 AM	291
1	12/18/ 2025 12:00:00 AM	313
1	12/17/ 2025 12:00:00 AM	272
1	12/16/ 2025 12:00:00 AM	350

Hình 3.9 Dữ liệu mẫu ở Oracle của bảng sl_khach_hang_moi_ngay_theo_ma_cn (CN1)

MaChiNhanh NUMBER(38)	NGAY DATE	SOLUONGKHACHHANG NUMBER
2	12/31/ 2025 12:00:00 AM	322
2	12/30/ 2025 12:00:00 AM	319
2	12/29/ 2025 12:00:00 AM	299
2	12/28/ 2025 12:00:00 AM	297
2	12/27/ 2025 12:00:00 AM	300
2	12/26/ 2025 12:00:00 AM	334
2	12/25/ 2025 12:00:00 AM	310
2	12/24/ 2025 12:00:00 AM	310
2	12/23/ 2025 12:00:00 AM	324
2	12/22/ 2025 12:00:00 AM	291
2	12/21/ 2025 12:00:00 AM	326
2	12/20/ 2025 12:00:00 AM	327
2	12/19/ 2025 12:00:00 AM	305
2	12/18/ 2025 12:00:00 AM	293
2	12/17/ 2025 12:00:00 AM	321
2	12/16/ 2025 12:00:00 AM	316
2	12/15/ 2025 12:00:00 AM	298
2	12/14/ 2025 12:00:00 AM	308
2	12/13/ 2025 12:00:00 AM	341
2	12/12/ 2025 12:00:00 AM	297
2	12/11/ 2025 12:00:00 AM	335
2	12/10/ 2025 12:00:00 AM	288
2	12/09/ 2025 12:00:00 AM	310
2	12/08/ 2025 12:00:00 AM	290
2	12/07/ 2025 12:00:00 AM	301

Hình 3.10 Dữ liệu mẫu ở Oracle của bảng sl_khach_hang_moi_ngay_theo_ma_cn (CN2)

3.5. Bảng doanh_thu_sp_quy_cn

```

SELECT
    KQL."MaChiNhanh",
    SP."MaSanPham",
    EXTRACT(YEAR FROM HD."NgayTao") AS Nam,
    TO_NUMBER(TO_CHAR(HD."NgayTao", 'Q')) AS Quy,
    SUM(CT."ThanhTien") AS DoanhThu
FROM
    "HoaDon" HD
    JOIN "ChiTietHoaDon" CT ON HD."MaHoaDon" = CT."MaHoaDon"

```

```

JOIN "SanPham" SP ON SP."MaSanPham" = CT."MaSanPham"
JOIN "NhanVien" NV ON NV."MaNhanVien" = HD."MaNhanVien"
JOIN "ChiNhanh" KQL ON NV."MaChiNhanh" = KQL."MaChiNhanh"
GROUP BY
    KQL."MaChiNhanh",
    SP."MaSanPham",
    EXTRACT(YEAR FROM HD."NgayTao"),
    TO_NUMBER(TO_CHAR(HD."NgayTao", 'Q'))
ORDER BY
    KQL."MaChiNhanh", SP."MaSanPham", Nam, Quy;

```

- **Dữ liệu mẫu:**

MaChiNhanh NUMBER(38)	MaSanPham NVARCHAR2(50)	NAM NUMBER	QUY NUMBER	DOANHTHU NUMBER
1	CCNPLT0000	2022	1	90400000
1	CCNPLT0000	2022	2	99600000
1	CCNPLT0000	2022	3	104600000
1	CCNPLT0000	2022	4	124000000
1	CCNPLT0000	2023	1	108800000
1	CCNPLT0000	2023	2	114800000
1	CCNPLT0000	2023	3	134800000
1	CCNPLT0000	2023	4	106600000
1	CCNPLT0000	2024	1	93200000
1	CCNPLT0000	2024	2	111200000
1	CCNPLT0000	2024	3	109600000
1	CCNPLT0000	2024	4	108000000
1	CCNPLT0000	2025	1	84400000
1	CCNPLT0000	2025	2	106400000
1	CCNPLT0000	2025	3	113200000
1	CCNPLT0000	2025	4	116200000

Hình 3.11 Dữ liệu mẫu ở Oracle của bảng doanh_thu_sp_quy_cn (CN1)

MaChiNhanh NUMBER(38)	MaSanPham NVARCHAR2(50)	NAM NUMBER	QUY NUMBER	DOANHThu NUMBER
2	CCNPLT0000	2022	1	113200000
2	CCNPLT0000	2022	2	125600000
2	CCNPLT0000	2022	3	88800000
2	CCNPLT0000	2022	4	93800000
2	CCNPLT0000	2023	1	138600000
2	CCNPLT0000	2023	2	113800000
2	CCNPLT0000	2023	3	127200000
2	CCNPLT0000	2023	4	103400000
2	CCNPLT0000	2024	1	109000000
2	CCNPLT0000	2024	2	111800000
2	CCNPLT0000	2024	3	95600000
2	CCNPLT0000	2024	4	106400000
2	CCNPLT0000	2025	1	101000000
2	CCNPLT0000	2025	2	115800000
2	CCNPLT0000	2025	3	80600000
2	CCNPLT0000	2025	4	98200000
2	CCNPLT0001	2022	1	204300000
2	CCNPLT0001	2022	2	246150000
2	CCNPLT0001	2022	3	270000000
2	CCNPLT0001	2022	4	272250000
2	CCNPLT0001	2023	1	220500000
2	CCNPLT0001	2023	2	231300000
2	CCNPLT0001	2023	3	238950000
2	CCNPLT0001	2023	4	227250000

Hình 3.12 Dữ liệu mẫu ở Oracle của bảng doanh_thu_sp_quy_cn (CN2)

3.6. Bảng doanh_thu_thang_nv_cn

```

SELECT
    NV."MaChiNhanh",
    NV."MaNhanVien",
    EXTRACT(YEAR FROM HD."NgayTao") AS Nam,
    EXTRACT(MONTH FROM HD."NgayTao") AS Thang,
    SUM(CT."ThanhTien") AS DoanhThu
FROM
    "HoaDon" HD
    JOIN "ChiTietHoaDon" CT ON HD."MaHoaDon" = CT."MaHoaDon"
    JOIN "NhanVien" NV ON HD."MaNhanVien" = NV."MaNhanVien"
GROUP BY

```

```

NV."MaChiNhanh",
NV."MaNhanVien",
EXTRACT(YEAR FROM HD."NgayTao"),
EXTRACT(MONTH FROM HD."NgayTao")
ORDER BY
NV."MaChiNhanh", NV."MaNhanVien", Nam, Thang;

```

- **Dữ liệu mẫu:**

MaChiNhanh NUMBER(38)	MaNhanVien NUMBER(38)	NAM NUMBER	THANG NUMBER	DOANHThu NUMBER
1	1	2022	1	74875653
1	1	2022	2	114406380
1	1	2022	3	84758688
1	1	2022	4	98233434
1	1	2022	5	119555882
1	1	2022	6	65796093
1	1	2022	7	145889492
1	1	2022	8	129838683
1	1	2022	9	179399121
1	1	2022	10	140884904
1	1	2022	11	188366296
1	1	2022	12	94173070
1	1	2023	1	105006812
1	1	2023	2	188448719
1	1	2023	3	123340603
1	1	2023	4	228232861

Hình 3.13 Dữ liệu mẫu ở Oracle của bảng doanh_thu_thang_nv_cn (CN1)

MaChiNhanh NUMBER(38)	MaNhanVien NUMBER(38)	NAM NUMBER	THANG NUMBER	DOANHThu NUMBER
2	501	2022	1	130992994
2	501	2022	2	87201257
2	501	2022	3	144118520
2	501	2022	4	80889932
2	501	2022	5	156779710
2	501	2022	6	181091094
2	501	2022	7	149424122
2	501	2022	8	214396519
2	501	2022	9	232639617
2	501	2022	10	115214442
2	501	2022	11	110695866
2	501	2022	12	201604555
2	501	2023	1	99855000
2	501	2023	2	133514800
2	501	2023	3	223460057
2	501	2023	4	152174832
2	501	2023	5	149138624
2	501	2023	6	140335918
2	501	2023	7	87115355
2	501	2023	8	134402596
2	501	2023	9	101606453
2	501	2023	10	101059323
2	501	2023	11	200303746
2	501	2023	12	149395534

Hình 3.14 Dữ liệu mẫu ở Oracle của bảng doanh_thu_thang_nv_cn (CN2)

CHƯƠNG 4. DI CHUYỂN DỮ LIỆU TỪ CƠ SỞ DỮ LIỆU QUAN HỆ SANG HỆ QUẢN TRỊ CASSANDRA

1. Quá trình di chuyển dữ liệu từ cơ sở dữ liệu quan hệ sang Cassandra

1.1. Giới thiệu về công cụ Apache Airflow

- *Apache Airflow* là một nền tảng mã nguồn mở mạnh mẽ, được dùng để lập trình (programmatically author), lên lịch (schedule), và giám sát (monitor) các workflow (quy trình làm việc). Nó được viết bằng Python và ban đầu được phát triển bởi Airbnb.



Hình 4.1 Công cụ Apache Airflow

- *Tại sao cần Apache Airflow?*

Trong thế giới dữ liệu hiện đại, các quy trình xử lý dữ liệu (như ETL/ELT), huấn luyện mô hình Machine Learning, hoặc các tác vụ tự động hóa khác thường rất phức tạp, bao gồm nhiều bước phụ thuộc lẫn nhau. Việc chạy các tác vụ này thủ công hoặc chỉ bằng cron jobs truyền thống gặp phải nhiều hạn chế:

- Thiếu sự phụ thuộc: Khó xác định và quản lý các tác vụ phải chạy sau khi tác vụ khác hoàn thành.
- Giám sát kém: Khó biết tác vụ nào đang chạy, tác vụ nào đã thất bại.
- Khó sửa lỗi (debugging): Khi một tác vụ lỗi, việc tìm ra nguyên nhân và khởi động lại từ điểm lỗi rất khó khăn.

- Không có giao diện trực quan: Thiếu cái nhìn tổng quan về trạng thái của các quy trình.
- Khó mở rộng: Việc quản lý hàng trăm, hàng nghìn tác vụ trở nên bất khả thi.

Airflow ra đời để giải quyết tất cả những vấn đề này, cung cấp một giải pháp toàn diện cho việc tự động hóa và quản lý workflow.

- *Các khái niệm cốt lõi trong Airflow:*

- DAG (Directed Acyclic Graph):
 - Là "trái tim" của Airflow.
 - Mỗi DAG đại diện cho một quy trình làm việc hoàn chỉnh (ví dụ: một pipeline ETL từ đầu đến cuối).
 - Nó bao gồm các "tasks" (nhiệm vụ) và mối quan hệ phụ thuộc có hướng giữa chúng (Task A phải hoàn thành trước Task B).
 - "Acyclic" có nghĩa là không có vòng lặp (không có task nào chạy lại một task trước đó trong cùng một DAG run).
 - Toàn bộ DAG được định nghĩa bằng code Python.
- Task (Nhiệm vụ):
 - Là các đơn vị công việc độc lập trong một DAG.
 - Mỗi Task đại diện cho một hành động cụ thể, ví dụ:
 - Chạy một đoạn script SQL.
 - Gọi một API.
 - Tải dữ liệu từ S3.
 - Gửi một email.
- Operator (Bộ điều hành):
 - Là các template đã được định nghĩa sẵn để tạo Task. Airflow cung cấp rất nhiều Operator tích hợp sẵn, giúp dễ dàng thực hiện các tác vụ phổ biến mà không cần viết code từ đầu.
 - Ví dụ phổ biến:
 - BashOperator: Chạy lệnh shell/bash.
 - PythonOperator: Thực thi một hàm Python.

- SqlOperator: Chạy câu lệnh SQL.
 - S3Hook, PostgresOperator, GCSOperator, KubernetesPodOperator, v.v. (để tương tác với các hệ thống khác).
- Sensor (Bộ cảm biến):
 - Là một loại Operator đặc biệt, được thiết kế để chờ đợi một điều kiện cụ thể xảy ra trước khi cho phép các tác vụ phụ thuộc tiếp tục.
 - Ví dụ:
 - FileSensor: Chờ đợi một file xuất hiện trong một thư mục cụ thể.
 - S3KeySensor: Chờ đợi một key (object) xuất hiện trong một bucket S3.
 - HttpSensor: Chờ đợi một phản hồi HTTP thành công từ một URL.
- *Các tính năng nổi bật của Apache Airflow:*
 - Định nghĩa workflow bằng Python: Cung cấp sự linh hoạt cao, cho phép sử dụng toàn bộ sức mạnh của Python để định nghĩa logic, biến, và phụ thuộc.
 - Giao diện người dùng Web trực quan (Web UI):
 - Cho phép giám sát trạng thái của tất cả các DAG và Task.
 - Xem lịch sử chạy, log của từng Task.
 - Kích hoạt/dừng DAG, điều chỉnh cấu hình.
 - Xem biểu đồ DAG để hiểu rõ mối quan hệ giữa các Task.
 - Lập lịch linh hoạt: Hỗ trợ các kiểu lập lịch khác nhau (theo khoảng thời gian, theo cron-like, theo sự kiện).
 - Khả năng phục hồi (Resilience):
 - Retry: Tự động thử lại các Task bị lỗi một số lần nhất định.

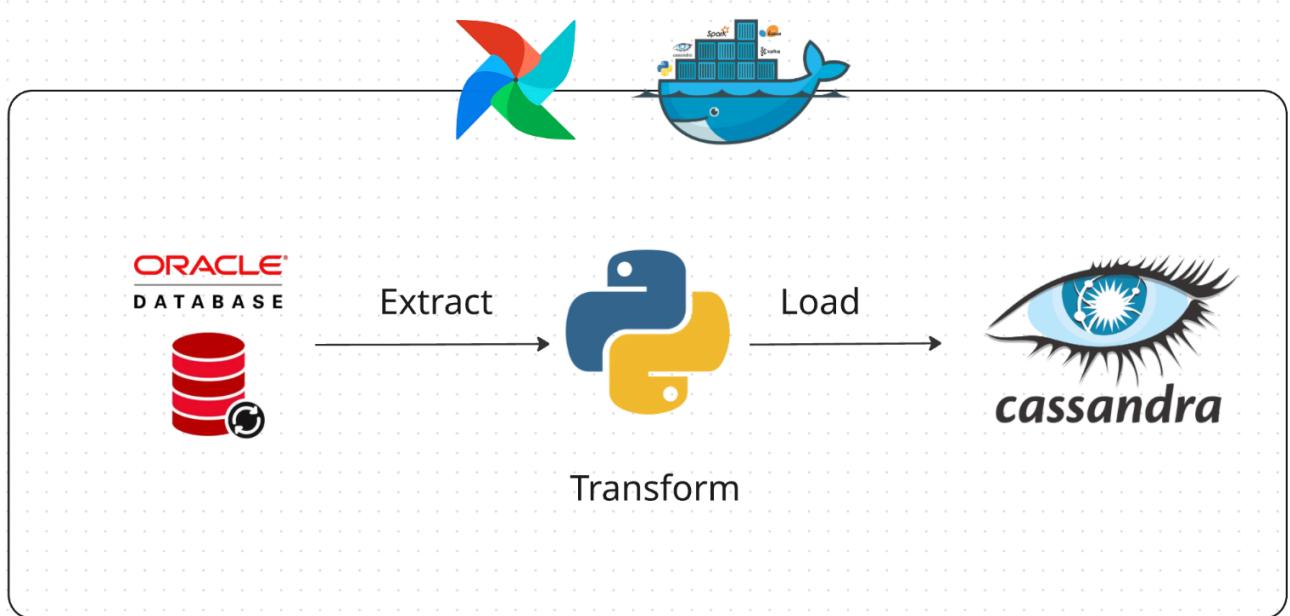
- Backfilling: Chạy lại các DAG cho các khoảng thời gian trong quá khứ.
 - Idempotency: Thiết kế để đảm bảo rằng việc chạy lại một Task nhiều lần sẽ cho cùng một kết quả cuối cùng.
 - Khả năng mở rộng (Scalability): Có thể mở rộng để xử lý hàng nghìn DAG và hàng triệu Task bằng cách sử dụng các Executor khác nhau (Local, Celery, Kubernetes).
 - Mã nguồn mở và cộng đồng lớn: Được hỗ trợ bởi một cộng đồng lớn mạnh, liên tục phát triển và cải tiến.
 - Tính mở rộng (Extensibility): Dễ dàng tạo các Operator, Hook, Plugin tùy chỉnh để tích hợp với các hệ thống mới hoặc các logic phức tạp.
- *Kiến trúc cơ bản của Airflow:*
- Web Server: Cung cấp giao diện người dùng web.
 - Scheduler: Liên tục quét các DAG mới và đã thay đổi, kiểm tra trạng thái của các Task và kích hoạt các Task đến hạn chạy.
 - Worker: Thực thi các Task thực tế. Số lượng worker có thể mở rộng tùy theo khối lượng công việc.
 - Database (Metastore): Lưu trữ tất cả metadata về các DAG, trạng thái Task, lịch sử chạy, v.v. (PostgreSQL, MySQL, SQLite).
 - Executor: Là cơ chế mà Scheduler sử dụng để gửi các Task đến Worker. Các Executor phổ biến bao gồm LocalExecutor, CeleryExecutor, KubernetesExecutor.
- *Các trường hợp sử dụng phổ biến:*
- Data Engineering (ETL/ELT): Xây dựng các pipeline để thu thập, chuyển đổi, tải dữ liệu từ nhiều nguồn khác nhau vào data warehouse hoặc data lake.
 - Machine Learning Pipelines: Tự động hóa các bước từ thu thập dữ liệu, tiền xử lý, huấn luyện mô hình, đánh giá, đến triển khai.

- DevOps Task Automation: Tự động hóa các tác vụ quản lý hệ thống, triển khai ứng dụng, tạo báo cáo.
- Reporting & Analytics: Tự động tạo và phân phối các báo cáo định kỳ.

1.2. Di chuyển dữ liệu từ cơ sở dữ liệu quan hệ sang Cassandra

Tổng quan

Thực hiện việc chuyển đổi dữ liệu từ cơ sở dữ liệu **Oracle sang Apache Cassandra** thông qua một **pipeline ETL** (Extract, Transform, Load) được xây dựng bằng **Apache Airflow**. Hệ thống được **đóng gói bằng Docker** để đảm bảo tính nhất quán và dễ dàng triển khai.



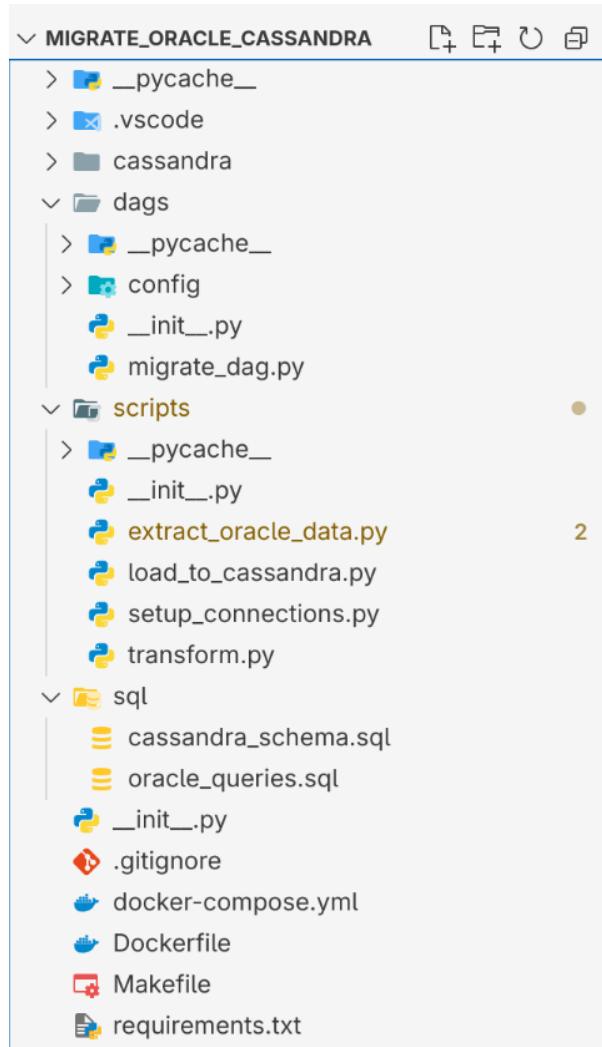
Hình 4.2 Quá trình di chuyển dữ liệu từ Oracle sang Cassandra

Kiến trúc hệ thống

1. Các thành phần chính

- Apache Airflow: Orchestration tool để quản lý các tác vụ ETL.
- Oracle Database: Nguồn dữ liệu gốc (BTL1 schema).
- Apache Cassandra: Cơ sở dữ liệu đích (BTL2_data keyspace).
- PostgreSQL: Metadata database cho Airflow.
- Docker: Container platform để đóng gói và triển khai.

2. Cấu trúc thư mục dự án



Hình 4.3 Cấu trúc thư mục repository di chuyển dữ liệu từ Oracle sang Cassandra

Trong đó:

- Thư mục **sql/**: Lưu trữ các file SQL và CQL (Cassandra Query Language).
 - **cassandra_schema.sql**: Định nghĩa schema cho Cassandra.
 - Tạo keyspace **BTL2_data**.
 - Định nghĩa các bảng đích với partition keys và clustering keys.
 - Cấu hình replication strategy.

- `oracle_queries.sql`: Định nghĩa các câu truy vấn SQL để extract dữ liệu từ Oracle cho các bảng tương ứng trong Cassandra.
- Thư mục `scripts/`: Chứa file Python scripts thực hiện logic ETL.
 - `extract_oracle_data.py`
 - Kết nối và trích xuất dữ liệu từ Oracle.
 - Thực thi các câu truy vấn.
 - Export dữ liệu ra pandas DataFrame.
 - `transform.py`
 - Chuyển đổi và chuẩn hóa dữ liệu có được từ quá trình extract.
 - Mapping tên cột từ Oracle sang định dạng của Cassandra.
 - `load_to_cassandra.py`
 - `load_invoice_details_data_optimized()`: Load dữ liệu chi tiết hóa đơn.
 - `load_revenue_data_optimized()`: Load dữ liệu doanh thu.
 - `load_wh_data_optimized()`: Load dữ liệu kho.
 - `load_cus_data_optimized()`: Load số lượng khách hàng.
 - `load_doanhthu_sp_data_optimized()`: Load doanh thu sản phẩm theo quý.
 - `load_doanhthu_nv_data_optimized()`: Load doanh thu nhân viên theo tháng.

Tính năng nổi bật:

- Tự động đóng connection đến Cassandra khi hoàn thành tác vụ.

- Xử lý dữ liệu theo Batch (mỗi batch gồm 1000 records).
- Xử lý 25 tác vụ đồng thời cho hiệu suất cao.

```
# Batch processing với 1000 records/batch
BATCH_SIZE = 1000

# Context manager cho session management
@contextmanager
def get_cassandra_session():
    # Tự động đóng connection khi hoàn thành

# Concurrent execution cho hiệu suất cao
def load_data_in_batches(params_list, session, prepared, concurrency=25):
    # Xử lý 25 operations đồng thời
```

- Thư mục **dag/**: Chứa Airflow DAGs định nghĩa workflow ETL.

- Định nghĩa workflow:

```
# DAG configuration
dag = DAG(
    'oracle_to_cassandra_migration',
    default_args={
        'owner': 'data_team',
        'retries': 3,
        'retry_delay': timedelta(minutes=5)
    },
    schedule_interval=None, # Manual trigger
    catchup=False
)
```

- Task dependencies:

```
setup_connections
  ↓
  extract_invoice_details - transform_invoice - load_invoice ↴
  |   extract_revenue ----- transform_revenue ----- load_revenue ↴
  |   extract_warehouse ----- transform_warehouse ----- load_warehouse ↴
  |   extract_customers ----- transform_customers ----- load_customers ↴
  |   extract_product_revenue - transform_prod_rev - load_prod_rev ↴
  |   extract_staff_revenue ----- transform_staff_rev ----- load_staff_rev ↴
  ↓
  verification_task
```

- Task types trong DAG:

1. *Setup tasks:*

```
setup_connections = PythonOperator(
    task_id='setup_connections',
    python_callable=setup_database_connections
)
```

2. *Extract tasks:*

```
extract_invoice = PythonOperator(  
    task_id='extract_invoice_details',  
  
    python_callable=extract_oracle_data.extract_invoi  
    ce_details  
)
```

3. *Transform tasks:*

```
transform_invoice = PythonOperator(  
    task_id='transform_invoice_data',
```

```
    python_callable=transform.transform_invoice_dat  
    a  
)
```

4. *Load tasks:*

```
load_invoice = PythonOperator(  
    task_id='load_invoice_to_cassandra',
```

```
    python_callable=load_to_cassandra.load_invoice_  
    details_data_optimized  
)
```

- Cấu hình kết nối databases

`dags/config/connections.yml`

oracle:

host: oracle_host

port: 1521

service_name: XE

username: BTL1

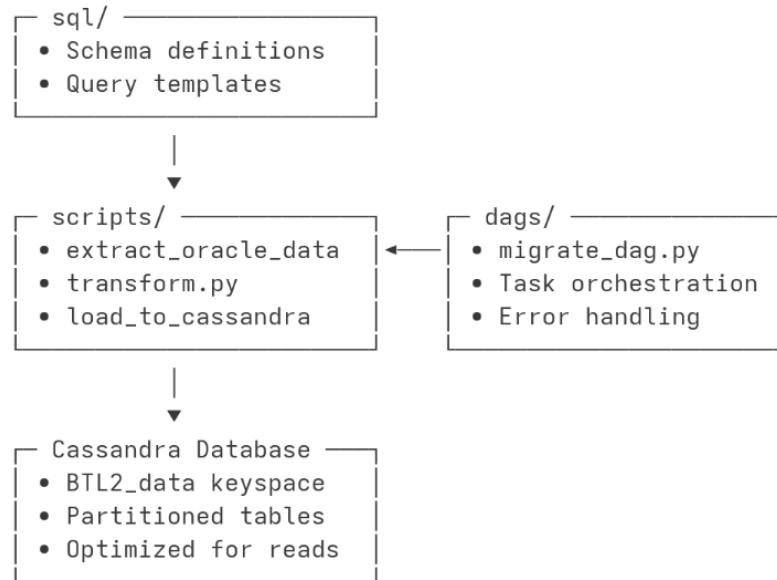
password: \${ORACLE_PASSWORD}

```

cassandra:
    hosts: ['cassandra']
    port: 9042
    keyspace: BTL2_data
    protocol_version: 4

```

3. Luồng dữ liệu giữa các thư mục



Hình 4.4 Luồng dữ liệu giữa các thư mục trong repository

4. Quá trình ETL chi tiết

Phase 1: Extract (Trích xuất dữ liệu)

- Nguồn dữ liệu: Oracle Database (schema BTL1)
- File thực thi: `extract_oracle_data.py`
- Các bảng được trích xuất:
 1. Chi tiết hóa đơn theo khách hàng
 - Query từ các bảng: ChiTietHoaDon, HoaDon, KhachHang.
 - Dữ liệu: Thông tin chi tiết hóa đơn bao gồm mã khách hàng, sản phẩm, số lượng, thành tiền.

```
-- chi_tiet_hoa_don_theo_ma_kh
SELECT
    hd."MaKhachHang",
    cthd."MaHoaDon",
    cthd."MaSanPham",
    cthd."SoLuong",
    cthd."ThanhTien",
    hd."TongTien",
    hd."NgayTao",
    hd."PhuongThucThanhToan",
    hd."MaNhanVien"
FROM BTL1."ChiTietHoaDon" cthd
JOIN BTL1."HoaDon" hd ON cthd."MaHoaDon" = hd."MaHoaDon"
RIGHT JOIN BTL1."KhachHang" kh ON HD."MaKhachHang" = KH."MaKhachHang"
ORDER BY BTL1."MaHoaDon", BTL1."NgayTao" DESC;
```

Hình 4.5 Bảng Chi tiết hóa đơn theo khách hàng

2. Doanh thu theo chi nhánh

- Query từ các bảng: HoaDon, NhanVien, ChiNhanh.
- Dữ liệu: Tổng doanh thu theo ngày và chi nhánh.

```
-- doanh_thu_moi_ngay_theo_ma_cn
SELECT
    cn."MaChiNhanh",
    TRUNC(hd."NgayTao") AS Ngay,
    SUM(hd."TongTien") AS TongTien
FROM "HoaDon" hd
RIGHT JOIN "NhanVien" nv ON hd."MaNhanVien" = nv."MaNhanVien"
JOIN "ChiNhanh" cn ON cn."MaChiNhanh" = nv."MaChiNhanh"
GROUP BY cn."MaChiNhanh", TRUNC(hd."NgayTao")
ORDER BY Ngay DESC;
```

Hình 4.6 Bảng Doanh thu theo chi nhánh

3. Kho sản phẩm theo chi nhánh

- Query từ các bảng: KhoSanPham_QLBanHang, KhoSanPham_QLKho, SanPham.
- Dữ liệu: Thông tin tồn kho, tình trạng sản phẩm.

```
-- kho_sp_theo_ma_cn
SELECT
    KSPQ."MaChiNhanh",
    KSPQ."MaSanPham",
    SP."TenSanPham",
    KSPQH."TinhTrang",
    KSPQH."TongSoLuongDanhGia",
    KSPQH."TongSoLuongDaBan",
    KSPQ."SoLuong"
FROM "KhoSanPham_QLBanHang" kspqh, "KhoSanPham_QLKho" kspq, "SanPham" sp
WHERE KSPQ."MaSanPham" = KSPQH."MaSanPham" AND SP."MaSanPham" = KSPQ."MaSanPham"
ORDER BY KSPQ."MaSanPham", KSPQ."SoLuong";
```

Hình 4.7 Bảng Kho sản phẩm theo chi nhánh

4. Số lượng khách hàng theo ngày

- Query từ các bảng: NhanVien, HoaDon.
- Dữ liệu: Thông kê khách hàng theo ngày và chi nhánh.

```
-- sl_khach_hang_moi_ngay_theo_ma_cn
SELECT
    NV."MaChiNhanh",
    TRUNC(HD."NgayTao") AS Ngay,
    COUNT(DISTINCT HD."MaKhachHang") AS SoLuongKhachHang
FROM "NhanVien" NV
    JOIN "HoaDon" HD ON NV."MaNhanVien" = HD."MaNhanVien"
GROUP BY NV."MaChiNhanh", TRUNC(HD."NgayTao")
ORDER BY Ngay DESC;
```

Hình 4.8 Bảng Số lượng khách hàng theo ngày

5. Doanh thu sản phẩm theo quý

- Query từ các bảng: HoaDon, ChiTietHoaDon, SanPham, NhanVien, ChiNhanh.

- Dữ liệu: Doanh thu từng sản phẩm theo quý và chi nhánh.

```
--Doanh thu của mỗi sản phẩm tại mỗi chi nhánh, theo từng quý và từng năm
--doanh_thu_sp_quy_cn
SELECT
    KQL."MaChiNhanh",
    SP."MaSanPham",
    EXTRACT(YEAR FROM HD."NgayTao") AS Nam,
    EXTRACT(QUARTER FROM HD."NgayTao") AS Quy,
    SUM(CT."ThanhTien") AS DoanhThu
FROM
    "HoaDon" HD
    JOIN "ChiTietHoaDon" CT ON HD."MaHoaDon" = CT."MaHoaDon"
    JOIN "SanPham" SP ON SP."MaSanPham" = CT."MaSanPham"
    JOIN "NhanVien" NV ON NV."MaNhanVien" = HD."MaNhanVien"
    JOIN "ChiNhanh" KQL ON NV."MaChiNhanh" = KQL."MaChiNhanh"
GROUP BY
    KQL."MaChiNhanh",
    SP."MaSanPham",
    EXTRACT(YEAR FROM HD."NgayTao"),
    EXTRACT(QUARTER FROM HD."NgayTao")
ORDER BY
    KQL."MaChiNhanh", SP."MaSanPham", Nam, Quy;
```

Hình 4.9 Bảng Doanh thu sản phẩm theo quý

6. Doanh thu nhân viên theo tháng

- Query từ các bảng: HoaDon, ChiTietHoaDon, NhanVien.
- Dữ liệu: Doanh thu từng nhân viên theo tháng.

```

--Doanh thu theo tháng của mỗi nhân viên ở từng chi nhánh
--doanh_thu_thang_nv_cn
SELECT
    NV."MaChiNhanh",
    NV."MaNhanVien",
    EXTRACT(YEAR FROM HD."NgayTao") AS Nam,
    EXTRACT(MONTH FROM HD."NgayTao") AS Thang,
    SUM(CT."ThanhTien") AS DoanhThu
FROM
    "HoaDon" HD
    JOIN "ChiTietHoaDon" CT ON HD."MaHoaDon" = CT."MaHoaDon"
    JOIN "NhanVien" NV ON HD."MaNhanVien" = NV."MaNhanVien"
GROUP BY
    NV."MaChiNhanh",
    NV."MaNhanVien",
    EXTRACT(YEAR FROM HD."NgayTao"),
    EXTRACT(MONTH FROM HD."NgayTao")
ORDER BY
    NV."MaChiNhanh", NV."MaNhanVien", Nam, Thang;

```

Hình 4.10 Bảng Doanh thu nhân viên theo tháng

extract_oracle_data.py

```

import pandas as pd
import json
from airflow.providers.oracle.hooks.oracle import OracleHook
from airflow.configuration import conf

# Enable pickle for XCom serialization
conf.set('core', 'enable_xcom_pickling', 'True')

def get_oracle_hook():

```

```

try:
    return OracleHook(oracle_conn_id='oracle_BTL2')
except Exception as e:
    print(f'Error when getting oracle hook: {e}')

oracle_hook = get_oracle_hook()

def execute_query(query):
    try:
        data = pd.read_sql(
            query,
            oracle_hook.get_conn()
        )

        print(f"Successfully extracted {len(data)} rows from Oracle")
        if len(data) > 0:
            print(f"Columns: {data.columns.tolist()}")
            print(f"Sample data: {data.head(2).to_dict('records')}")

        return data
    except Exception as e:
        print(f"Error extracting Oracle data: {e}")
        return pd.DataFrame()

def extract_invoice_data():
    if not oracle_hook:
        print("Oracle hook not available for invoice data")
    return pd.DataFrame()

```

```

invoice_query = """
SELECT
    hd."MaKhachHang",
    cthd."MaHoaDon",
    cthd."MaSanPham",
    cthd."SoLuong",
    cthd."ThanhTien",
    hd."TongTien",
    hd."NgayTao",
    hd."PhuongThucThanhToan",
    hd."MaNhanVien"
FROM "BTL1"."ChiTietHoaDon" cthd
    JOIN "BTL1"."HoaDon" hd ON cthd."MaHoaDon" = hd."MaHoaDon"
    JOIN "BTL1"."KhachHang" kh ON hd."MaKhachHang" =
kh."MaKhachHang"
    ORDER BY cthd."MaHoaDon", hd."NgayTao" DESC
"""

print("Extracting invoice data...")
df = execute_query(invoice_query)
return {
    'invoice_data': df
}

def extract_revenue_branch_data():
    if not oracle_hook:
        print("Oracle hook not available for revenue data")
    return pd.DataFrame()

```

```

revenue_query = """
    SELECT
        cn."MaChiNhanh",
        TRUNC(hd."NgayTao") AS "Ngay",
        SUM(hd."TongTien") AS "TongTien"
    FROM "BTL1"."HoaDon" hd
        JOIN "BTL1"."NhanVien" nv ON hd."MaNhanVien" = nv."MaNhanVien"
        JOIN "BTL1"."ChiNhanh" cn ON cn."MaChiNhanh" = nv."MaChiNhanh"
    GROUP BY cn."MaChiNhanh", TRUNC(hd."NgayTao")
    ORDER BY TRUNC(hd."NgayTao") DESC
"""

print("Extracting revenue data...")
df = execute_query(revenue_query)
return {
    'revenue_branch_data': df
}

def extract_warehouse_data():
    if not oracle_hook:
        print("Oracle hook not available for warehouse data")
    return pd.DataFrame()

warehouse_query = """
    SELECT
        kspq."MaChiNhanh",
        kspq."MaSanPham",
        sp."TenSanPham",
        kspqh."TinhTrang",
"""

```

```

kspqh."TongSoLuongDanhGia",
kspqh."TongSoLuongDaBan",
kspq."SoLuong"
FROM "BTL1"."KhoSanPham_QLBanHang" kspqh,
"BTL1"."KhoSanPham_QLKho" kspq,
"BTL1"."SanPham" sp
WHERE kspqh."MaSanPham" = kspq."MaSanPham"
AND sp."MaSanPham" = kspq."MaSanPham"
ORDER BY kspq."MaSanPham", kspq."SoLuong"
"""

```

```

print("Extracting warehouse data...")
df = execute_query(warehouse_query)
return {
    'warehouse_data': df
}

```

```

def extract_customer_data():
    if not oracle_hook:
        print("Oracle hook not available for customer data")
        return pd.DataFrame()

    cus_query = """
        SELECT
            nv."MaChiNhanh",
            TRUNC(hd."NgayTao") AS Ngay,
            COUNT(DISTINCT hd."MaKhachHang") AS SoLuongKhachHang
        FROM "BTL1"."NhanVien" nv
        JOIN "BTL1"."HoaDon" hd ON nv."MaNhanVien" = hd."MaNhanVien"
    """

```

GROUP BY nv."MaChiNhanh", TRUNC(hd."NgayTao")

ORDER BY Ngay DESC

"""

```
print("Extracting customer data...")
```

```
df = execute_query(cus_query)
```

```
return {
```

```
'customer_data': df
```

```
}
```

```
def extract_doanh_thu_sp_quy_cn():
```

```
if not oracle_hook:
```

```
    print("Oracle hook not available for customer data")
```

```
    return pd.DataFrame()
```

```
cus_query = """
```

```
SELECT
```

```
    KQL."MaChiNhanh",
```

```
    SP."MaSanPham",
```

```
    EXTRACT(YEAR FROM HD."NgayTao") AS Nam,
```

```
    TO_NUMBER(TO_CHAR(HD."NgayTao", 'Q')) AS Quy,
```

```
    SUM(CT."ThanhTien") AS DoanhThu
```

```
FROM
```

```
    "HoaDon" HD
```

```
    JOIN "ChiTietHoaDon" CT ON HD."MaHoaDon" = CT."MaHoaDon"
```

```
    JOIN "SanPham" SP ON SP."MaSanPham" = CT."MaSanPham"
```

```
    JOIN "NhanVien" NV ON NV."MaNhanVien" = HD."MaNhanVien"
```

```
    JOIN "ChiNhanh" KQL ON NV."MaChiNhanh" = KQL."MaChiNhanh"
```

```
GROUP BY
```

```

KQL."MaChiNhanh",
SP."MaSanPham",
EXTRACT(YEAR FROM HD."NgayTao"),
TO_NUMBER(TO_CHAR(HD."NgayTao", 'Q'))
ORDER BY
KQL."MaChiNhanh", SP."MaSanPham", Nam, Quy
""""

print("Extracting doanh_thu_sp_quy_cn data...")
df = execute_query(cus_query)
return {
    'doanh_thu_sp_quy_cn_data': df
}

def extract_doanh_thu_thang_nv_cn():
    if not oracle_hook:
        print("Oracle hook not available for customer data")
        return pd.DataFrame()

cus_query = """"
SELECT
NV."MaChiNhanh",
NV."MaNhanVien",
EXTRACT(YEAR FROM HD."NgayTao") AS Nam,
EXTRACT(MONTH FROM HD."NgayTao") AS Thang,
SUM(CT."ThanhTien") AS DoanhThu
FROM
"HoaDon" HD
JOIN "ChiTietHoaDon" CT ON HD."MaHoaDon" = CT."MaHoaDon"

```

```
JOIN "NhanVien" NV ON HD."MaNhanVien" = NV."MaNhanVien"
```

```
GROUP BY
```

```
    NV."MaChiNhanh",
```

```
    NV."MaNhanVien",
```

```
    EXTRACT(YEAR FROM HD."NgayTao"),
```

```
    EXTRACT(MONTH FROM HD."NgayTao")
```

```
ORDER BY
```

```
    NV."MaChiNhanh", NV."MaNhanVien", Nam, Thang
```

```
""""
```

```
print("Extracting doanh_thu_thang_nv_cn data...")
```

```
df = execute_query(cus_query)
```

```
return {
```

```
    'doanh_thu_thang_nv_cn_data': df
```

```
}
```

Phase 2: Transform (Chuyển đổi dữ liệu)

- File thực thi: **transform.py**
- Các bước chuyển đổi:
 1. Chuẩn hóa tên cột: Chuyển từ PascalCase sang snake_case.
 2. Chuyển đổi kiểu dữ liệu:
 - Datetime columns: Chuyển sang pandas datetime.
 - Numeric columns: Chuyển sang numeric, fill NaN = 0.
 - String columns: Trim whitespace.
 3. Mapping tên cột theo schema Cassandra.

```
transform.py
```

```

import pandas as pd
import traceback

def transform_data(input_data, k_name, num_cols, str_cols, datetime_cols,
cols_mapping):
    try:
        if input_data is None or not isinstance(input_data, dict):
            print("Invalid separated data received")
            return {}

        transformed_df = {}

        if k_name in input_data:
            tmp_df = pd.DataFrame(input_data[k_name].copy())
            tmp_df.columns = tmp_df.columns.str.lower()

            tmp_df.rename(columns=cols_mapping, inplace=True)

        if datetime_cols:
            for col in datetime_cols:
                if col in tmp_df.columns:
                    tmp_df[col] = pd.to_datetime(tmp_df[col],
errors='coerce')

        if num_cols:
            for col in num_cols:
                if col in tmp_df.columns:
                    tmp_df[col] = pd.to_numeric(tmp_df[col],
errors='coerce').fillna(0)

```

```

        if str_cols:
            for col in str_cols:
                if col in tmp_df.columns:
                    tmp_df[col] = tmp_df[col].astype(str).str.strip()

        transformed_df[k_name] = tmp_df

    return transformed_df

except Exception as e:
    print(f"Error transforming {k_name} data: {e}")
    import traceback
    traceback.print_exc()
    return {}

def transform_invoice_data(invoice_df):
    invoice_mapping = {
        'makhachhang': 'ma_khach_hang',
        'mahoadon': 'ma_hoa_don',
        'masanpham': 'ma_san_pham',
        'soluong': 'so_luong',
        'thanhtien': 'thanh_tien',
        'tongtien': 'tong_tien',
        'ngaytao': 'ngay_tao',
        'phuongthucthanhtoan': 'phuong_thuc_thanh_toan',
        'manhanvien': 'ma_nhan_vien'
    }

```

```

        datetime_cols = ['ngay_tao']

        num_cols = [
            'so_luong',
            'thanh_tien',
            'tong_tien',
            'ma_khach_hang',
            'ma_hoa_don'
        ]

        return transform_data(invoice_df, 'invoice_data', num_cols, None, datetime_cols,
                           invoice_mapping)

def transform_revenue_branch_data(revenue_df):
    revenue_mapping = {
        'machinhanh': 'ma_chi_nhanh',
        'ngay': 'ngay',
        'tongtien': 'tong_tien'
    }

    datetime_cols = ['ngay']
    num_cols = ['tong_tien']

    return transform_data(revenue_df, 'revenue_branch_data', num_cols, None,
                           datetime_cols, revenue_mapping)

def transform_warehouse_data(warehouse_df):
    warehouse_mapping = {
        'machinhanh': 'ma_chi_nhanh',
        'masanpham': 'ma_san_pham',
    }

```

```

'tensanpham': 'ten_san_pham',
'tinhtrang': 'tinh_trang',
'tongsoluongdanhgia': 'tong_so_luong_danh_gia',
'tongsoluongdaban': 'tong_so_luong_da_ban',
'soluong': 'tong_so_luong_ton_kho'
}

num_cols = ['tong_so_luong_danh_gia', 'tong_so_luong_da_ban',
'tong_so_luong_ton_kho']
str_cols = ['ten_san_pham', 'tinh_trang']

return transform_data(warehouse_df, 'warehouse_data', num_cols, str_cols, None,
warehouse_mapping)

def transform_customer_data(customer_df):
    customer_mapping = {
        'machinhanh': 'ma_chi_nhanh',
        'ngay': 'ngay',
        'soluongkhachhang': 'so_luong_khach_hang'
    }

    datetime_cols = ['ngay']
    num_cols = ['so_luong_khach_hang']

    return transform_data(customer_df, 'customer_data', num_cols, None,
datetime_cols, customer_mapping)

def transform_revenue_sp_quy_cn(doanhthu_sp_df):
    cols_mapping = {

```

```

'machinhanh': 'ma_chi_nhanh',
'masanpham': 'ma_san_pham',
'nam': 'nam',
'quy': 'quy',
'doanhthu': 'tong_doanh_thu'

}

num_cols = ['nam', 'quy', 'ma_chi_nhanh', 'tong_doanh_thu']
str_cols = ['ma_san_pham']

if isinstance(doanhthu_sp_df, pd.DataFrame):
    input_dict = {'doanh_thu_sp_quy_cn': doanhthu_sp_df}
else:
    input_dict = doanhthu_sp_df

return transform_data(input_dict, 'doanh_thu_sp_quy_cn', num_cols, str_cols,
None, cols_mapping)

def transform_revenue_thang_nv_cn(doanhthu_nv_df):
    cols_mapping = {
        'machinhanh': 'ma_chi_nhanh',
        'manhanvien': 'ma_nhan_vien',
        'nam': 'nam',
        'thang': 'thang',
        'doanhthu': 'tong_doanh_thu'
    }

    num_cols = ['nam', 'thang', 'ma_chi_nhanh', 'tong_doanh_thu', 'ma_nhan_vien']

```

```

if isinstance(doanhthu_nv_df, pd.DataFrame):
    input_dict = {'doanh_thu_thang_nv_cn': doanhthu_nv_df}
else:
    input_dict = doanhthu_nv_df

return transform_data(input_dict, 'doanh_thu_thang_nv_cn', num_cols, None,
None, cols_mapping)

```

Phase 3: Load (Tải dữ liệu)

- File thực thi: **load_to_cassandra.py**
- Đặc điểm:
 - Batch processing: Xử lý load dữ liệu theo batch 1000 records.
 - Concurrent execution: Sử dụng execute_concurrent_with_args.
 - Error handling: Xử lý lỗi và retry logic.
 - Prepared statements: Tối ưu hiệu suất.

load_to_cassandra.py

```

from contextlib import contextmanager
import pandas as pd
from cassandra.cluster import Cluster
import traceback
from cassandra.concurrent import execute_concurrent_with_args

BATCH_SIZE=1000

def get_cassandra_cluster():

```

```

try:
    cluster = Cluster(
        ['cassandra'], # Use container name
        port=9042,
        protocol_version=4,
        connect_timeout=20,
        control_connection_timeout=20
    )
    return cluster

except Exception as e:
    print(f"Error when getting Cassandra hook: {e}")
    return None

```

```

@contextmanager
def get_cassandra_session():
    """Context manager for Cassandra session"""
    cluster = None
    session = None
    try:
        cluster = get_cassandra_cluster()
        if cluster:
            session = cluster.connect()
            session.execute("USE BTL2_data")
            session.default_timeout = 60 # Increase timeout for large operations
            yield session
        else:
            yield None
    except Exception as e:

```

```

        print(f"Error with Cassandra session: {e}")
        yield None
    finally:
        if session:
            session.shutdown()
        if cluster:
            cluster.shutdown()

def load_data_in_batches(params_list, session, prepared, concurrency=25):
    try:
        total_batches = len(params_list) // BATCH_SIZE + (1 if len(params_list) % BATCH_SIZE else 0)

        for i in range(0, len(params_list), BATCH_SIZE):
            batch_params = params_list[i:i + BATCH_SIZE]
            print(f"Processing batch {i//BATCH_SIZE + 1}/{total_batches}")

            results = execute_concurrent_with_args(
                session,
                prepared,
                batch_params,
                concurrency,
                raise_on_first_error=False
            )

            errors = []
            for j, (success, result) in enumerate(results):
                if not success:
                    batch_index = i + j

```

```

        error_msg = str(result) if result else "Unknown error"
        print(f"Error at record {batch_index}: {error_msg}")
        if batch_index < len(params_list):
            print(f"Failed parameters:
{params_list[batch_index]}")
            errors.append(result)

    if errors:
        print(f"Errors in batch: {len(errors)}")
        # Print first few errors for debugging
        for idx, error in enumerate(errors[:3]): # Show first 3 errors
            print(f"Error {idx + 1}: {error}")

    print(f"Successfully processed {len(params_list)} categories to Cassandra")
except Exception as e:
    print(f"Error: {e}")

def load_user_data_optimized(user_data):
    try:
        print(f"Input data type: {type(user_data)}")

        if user_data.empty:
            print("Empty user data, nothing to load")
            return

    with get_cassandra_session() as session:
        if not session:
            print("No Cassandra session available")
            return

```

```

insert_user_cql = """
    INSERT INTO BTL2_data.khachhang (
        ma_khach_hang, email, ho_ten, sdt, dia_chi, gioi_tinh,
ngay_sinh
    ) VALUES (?, ?, ?, ?, ?, ?, ?)
"""

prepared = session.prepare(insert_user_cql)

# Prepare data for batch execution
parameters_list = []
for _, row in user_data.iterrows():
    params = [
        row['ma_khach_hang'],
        row['email'],
        row['ho_ten'],
        row['sdt'],
        row['dia_chi'],
        row['gioi_tinh'],
        row['ngay_sinh']
    ]
    parameters_list.append(params)

load_data_in_batches(parameters_list, session, prepared)

print(f"Successfully processed {len(parameters_list)} users to
Cassandra")

except Exception as e:

```

```

        print(f"Error loading users to Cassandra: {e}")
        traceback.print_exc()

def load_product_data_optimized(product_data):
    print(f"product_data of type {type(product_data)}")
    print(f"product_data shape: {product_data.shape if hasattr(product_data, 'shape')\
else 'No shape'}")

    if product_data.empty:
        print("Empty product data, nothing to load")
        return

    try:
        with get_cassandra_session() as session:
            if not session:
                print("No Cassandra session available")
                return

            insert_product_cql = """
                INSERT INTO BTL2_data.sanpham (
                    ma_san_pham, ten_san_pham, the_loai, gia
                ) VALUES (?, ?, ?, ?)
            """
            prepared = session.prepare(insert_product_cql)

            parameters_list = []
            for _, row in product_data.iterrows():
                params = [
                    row['ma_san_pham'],

```

```

        row['ten_san_pham'],
        row['the_loai'],
        row['gia']
    ]
parameters_list.append(params)

load_data_in_batches(parameters_list, session, prepared)

print(f"Successfully processed {len(parameters_list)} products to
Cassandra")

except Exception as e:
    print(f"Error loading products to Cassandra: {e}")
    traceback.print_exc()

def load_attr_product_data_optimized(attr_product_data):
    print(f"attr_product_data of type {type(attr_product_data)}")
    print(f"attr_product_data shape: {attr_product_data.shape if
hasattr(attr_product_data, 'shape') else 'No shape'}")

    if attr_product_data.empty:
        print("Empty attribute data, nothing to load")
        return

try:
    with get_cassandra_session() as session:
        if not session:
            print("No Cassandra session available")
            return

```

```

insert_attr_cql = """
    INSERT INTO BTL2_data.thuoctinh_sanpham (
        ma_san_pham, ten_thuoc_tinh, gia_tri_thuoc_tinh
    ) VALUES (?, ?, ?)
"""

prepared = session.prepare(insert_attr_cql)

parameters_list = []
for _, row in attr_product_data.iterrows():
    params = [
        row['ma_san_pham'],
        row['ten_thuoc_tinh'],
        row['gia_tri_thuoc_tinh']
    ]
    parameters_list.append(params)

load_data_in_batches(parameters_list, session, prepared)

print(f"Successfully processed {len(parameters_list)} attributes to
Cassandra")

except Exception as e:
    print(f"Error loading attributes to Cassandra: {e}")
    traceback.print_exc()

def load_cat_product_data_optimized(cat_product_data):
    print(f"cat_product_data of type {type(cat_product_data)}")
    print(f"cat_product_data shape: {cat_product_data.shape if
hasattr(cat_product_data, 'shape') else 'No shape'}")

```

```

if cat_product_data.empty:
    print("Empty category data, nothing to load")
    return

try:
    with get_cassandra_session() as session:
        if not session:
            print("No Cassandra session available")
            return

        insert_cat_cql = """
            INSERT INTO BTL2_data.danhmuc_sanpham (
                ma_san_pham, ten_danh_muc
            ) VALUES (?, ?)
        """
        prepared = session.prepare(insert_cat_cql)

        parameters_list = []
        for _, row in cat_product_data.iterrows():
            params = [
                row['ma_san_pham'],
                row['ten_danh_muc']
            ]
            parameters_list.append(params)

        load_data_in_batches(parameters_list, session, prepared)

        print(f"Successfully processed {len(parameters_list)} categories to
Cassandra")

```

```

except Exception as e:
    print(f'Error loading categories to Cassandra: {e}')
    traceback.print_exc()

def load_invoice_details_data_optimized(invoice_data):
    print(f'invoice_data of type {type(invoice_data)}')
    if not isinstance(invoice_data, pd.DataFrame):
        print(f'Expected DataFrame, got {type(invoice_data)}')
        return

    if invoice_data.empty:
        print('Empty input data, nothing to load')
        return

    try:
        with get_cassandra_session() as session:
            if not session:
                print("No Cassandra session available")
                return

            prepared_stmt = session.prepare("""
                INSERT INTO BTL2_data.chi_tiet_hoa_don_theo_ma_kh (
                    ma_khach_hang, ma_hoa_don, ma_san_pham,
                    so_luong,
                    thanh_tien, tong_tien, ngay_tao,
                    phuong_thuc_thanh_toan, ma_nhan_vien
                ) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)
            """)

```

```

params_list = []
for _, row in invoice_data.iterrows():
    params = [
        row['ma_khach_hang'],
        row['ma_hoa_don'],
        row['ma_san_pham'],
        row['so_luong'],
        row['thanh_tien'],
        row['tong_tien'],
        pd.to_datetime(row['ngay_tao']).to_pydatetime(),
        row['phuong_thuc_thanh_toan'],
        row['ma_nhan_vien']
    ]
    params_list.append(params)

load_data_in_batches(params_list, session, prepared_stmt)

except Exception as e:
    print(f"Error loading categories to Cassandra: {e}")
    traceback.print_exc()

def load_revenue_data_optimized(revenue_data):
    print(f'revenue_data of type {type(revenue_data)}')
    if not isinstance(revenue_data, pd.DataFrame):
        print(f'Expected DataFrame, got {type(revenue_data)}')
        return

    if revenue_data.empty:
        print('Empty input data, nothing to load')

```

```

        return

    try:
        with get_cassandra_session() as session:
            if not session:
                print("No Cassandra session available")
                return

            prepared_stmt = session.prepare("""
                INSERT INTO
                BTL2_data.doanh_thu_moi_ngay_theo_ma_cn (
                    ma_chi_nhanh,
                    ngay,
                    tong_tien
                ) VALUES (?, ?, ?)
                """
            )

            params_list = []
            for _, row in revenue_data.iterrows():
                params = [
                    row['ma_chi_nhanh'],
                    row['ngay'],
                    row['tong_tien']
                ]
                params_list.append(params)

            load_data_in_batches(params_list, session, prepared_stmt)

    except Exception as e:

```

```

        print(f"Error loading to Cassandra: {e}")
        traceback.print_exc()

def load_wh_data_optimized(wh_data):
    print(f"wh_data of type {type(wh_data)}")
    if not isinstance(wh_data, pd.DataFrame):
        print(f"Expected DataFrame, got {type(wh_data)}")
        return

    if wh_data.empty:
        print('Empty input data, nothing to load')
        return

    try:
        with get_cassandra_session() as session:
            if not session:
                print("No Cassandra session available")
                return

            prepared_stmt = session.prepare("""
                INSERT INTO BTL2_data.kho_sp_theo_ma_cn (
                    ma_chi_nhanh,
                    ma_san_pham,
                    ten_san_pham,
                    tinh_trang,
                    tong_so_luong_danh_gia,
                    tong_so_luong_da_ban,
                    tong_so_luong_ton_kho
                ) VALUES (?, ?, ?, ?, ?, ?, ?, ?)
            """)

```

```

        """)

params_list = []
for _, row in wh_data.iterrows():
    params = [
        row['ma_chi_nhanh'],
        row['ma_san_pham'],
        row['ten_san_pham'],
        row['tinh_trang'],
        row['tong_so_luong_danh_gia'],
        row['tong_so_luong_da_ban'],
        row['tong_so_luong_ton_kho']
    ]
    params_list.append(params)

load_data_in_batches(params_list, session, prepared_stmt)

except Exception as e:
    print(f"Error loading to Cassandra: {e}")
    traceback.print_exc()

def load_cus_data_optimized(cus_data):
    print(f"user_data of type {type(cus_data)}")
    if not isinstance(cus_data, pd.DataFrame):
        print(f"Expected DataFrame, got {type(cus_data)}")
        return

    if cus_data.empty:
        print('Empty input data, nothing to load')

```

```

        return

    try:
        with get_cassandra_session() as session:
            if not session:
                print("No Cassandra session available")
                return

            prepared_stmt = session.prepare("""
                INSERT INTO
                BTL2_data.sl_khach_hang_moi_ngay_theo_ma_cn (
                    ma_chi_nhanh,
                    ngay,
                    so_luong_khach_hang
                ) VALUES (?, ?, ?)
                """
            )

            params_list = []
            for _, row in cus_data.iterrows():
                params = [
                    row['ma_chi_nhanh'],
                    row['ngay'],
                    row['so_luong_khach_hang']
                ]
                params_list.append(params)

            load_data_in_batches(params_list, session, prepared_stmt)

    except Exception as e:

```

```

        print(f"Error loading to Cassandra: {e}")
        traceback.print_exc()

def load_doanhnhanh_sp_data_optimized(dt_sp_data):
    print(f"user_data of type {type(dt_sp_data)}")
    if not isinstance(dt_sp_data, pd.DataFrame):
        print(f"Expected DataFrame, got {type(dt_sp_data)}")
        return

    if dt_sp_data.empty:
        print('Empty input data, nothing to load')
        return

    try:
        with get_cassandra_session() as session:
            if not session:
                print("No Cassandra session available")
                return

            prepared_stmt = session.prepare("""
                INSERT INTO BTL2_data.doanh_thu_sp_quy_cn (
                    ma_chi_nhanh,
                    ma_san_pham,
                    nam,
                    quy,
                    tong_doanh_thu
                ) VALUES (?, ?, ?, ?, ?);
            """)

```

```

        params_list = []
        for _, row in dt_sp_data.iterrows():
            params = [
                row['ma_chi_nhanh'],
                row['ma_san_pham'],
                row['nam'],
                row['quy'],
                row['tong_doanh_thu']
            ]
            params_list.append(params)

    load_data_in_batches(params_list, session, prepared_stmt)

except Exception as e:
    print(f"Error loading to Cassandra: {e}")
    traceback.print_exc()

def load_doanhthu_nv_data_optimized(dt_nv_data):
    print(f"user_data of type {type(dt_nv_data)}")
    if not isinstance(dt_nv_data, pd.DataFrame):
        print(f"Expected DataFrame, got {type(dt_nv_data)}")
        return

    if dt_nv_data.empty:
        print('Empty input data, nothing to load')
        return

    try:
        with get_cassandra_session() as session:

```

```

if not session:
    print("No Cassandra session available")
    return

prepared_stmt = session.prepare("""
    INSERT INTO BTL2_data.doanh_thu_thang_nv_cn (
        ma_chi_nhanh,
        ma_nhan_vien,
        nam,
        thang,
        tong_doanh_thu
    ) VALUES (?, ?, ?, ?, ?)
""")

params_list = []
for _, row in dt_nv_data.iterrows():
    params = [
        row['ma_chi_nhanh'],
        row['ma_nhan_vien'],
        row['nam'],
        row['thang'],
        row['tong_doanh_thu']
    ]
    params_list.append(params)

load_data_in_batches(params_list, session, prepared_stmt)

except Exception as e:
    print(f"Error loading to Cassandra: {e}")

```

```
traceback.print_exc()
```

4. Airflow DAG

- Quản lý các tác vụ trong ETL pipeline sử dụng DAG (Directed Acyclic Graph - Đồ thị có hướng không chu trình).
- File thực thi: **migrate_dag.py**
- Task dependencies:
 - Thiết lập kết nối với cơ sở dữ liệu Oracle.
 - 6 extract tasks chạy song song sau bước setup.
 - Mỗi transform task phụ thuộc vào extract task tương ứng.
 - Mỗi load task phụ thuộc vào transform task tương ứng.
 - Verification task chờ tất cả load tasks hoàn thành.



Hình 4.11 Pipeline toàn bộ quá trình ETL dữ liệu từ Oracle sang Cassandra

```
migrate_dag.py
```

```
from datetime import datetime, timedelta
import pandas as pd
import sys
import os

from airflow import DAG
from airflow.operators.python import PythonOperator
from airflow.exceptions import AirflowSkipException
from airflow.operators.python import BranchPythonOperator

# Add scripts directory to path to import modules
sys.path.append('/opt/airflow/scripts')
sys.path.append(os.path.join(os.path.dirname(__file__), '..', 'scripts'))

from extract_oracle_data import extract_invoice_data, extract_revenue_branch_data,
extract_warehouse_data, extract_customer_data, extract_doanh_thu_sp_quy_cn,
extract_doanh_thu_thang_nv_cn
from transform import transform_invoice_data, transform_revenue_branch_data,
transform_warehouse_data, transform_customer_data, transform_revenue_sp_quy_cn,
transform_revenue_thang_nv_cn
from setup_connections import setup_connections

default_args = {
    'owner': 'hienfaang',
    'email_on_failure': True,
    'email_on_retry': False,
    'email': 'thuyhienphanthi2004@gmail.com',
    'retries': 3,
    'retry_delay': timedelta(minutes=5),
```

```

        'start_date': datetime(2025, 5, 28)
    }

dag = DAG(
    dag_id='oracle_to_cassandra_migration',
    default_args=default_args,
    description='ETL pipeline from Oracle to Cassandra',
    schedule_interval=None,
    catchup=False,
    tags=['migration', 'oracle', 'cassandra']
)

setup_connections_op = PythonOperator(
    task_id='setup_db_connections',
    python_callable=setup_connections,
    dag=dag
)

# Task 1: Extract data from Oracle
extract_invoice_data_op = PythonOperator(
    task_id='extract_invoice_data',
    python_callable=extract_invoice_data,
    dag=dag
)

extract_revenue_branch_data_op = PythonOperator(
    task_id='extract_revenue_branch_data',
    python_callable=extract_revenue_branch_data,
    dag=dag
)

```

```
)  
  
extract_warehouse_data_op = PythonOperator(  
    task_id='extract_warehouse_data',  
    python_callable=extract_warehouse_data,  
    dag=dag  
)  
  
extract_customer_data_op = PythonOperator(  
    task_id='extract_customer_data',  
    python_callable=extract_customer_data,  
    dag=dag  
)  
  
extract_revenue_product_data_op = PythonOperator(  
    task_id='extract_revenue_sp_data',  
    python_callable=extract_doanh_thu_sp_quy_cn,  
    dag=dag  
)  
  
extract_revenue_employee_data_op = PythonOperator(  
    task_id='extract_revenue_nv_data',  
    python_callable=extract_doanh_thu_thang_nv_cn,  
    dag=dag  
)  
  
# Task 2: Transform the extracted data  
def transformed_invoice_data(**kwargs):  
    ti = kwargs['ti']
```

```

input_data = ti.xcom_pull(task_ids='extract_invoice_data')
print(f"Extracted data type: {type(input_data)}")
print("Transforming data...")

return {
    'invoice_data': transform_invoice_data(input_data)
}

transform_invoice_data_op = PythonOperator(
    task_id='transform_invoice_data',
    python_callable=transformed_invoice_data,
    provide_context=True,
    dag=dag
)

def transformed_revenue_data(**kwargs):
    ti = kwargs['ti']

    input_data = ti.xcom_pull(task_ids='extract_revenue_branch_data')
    print(f"Extracted data type: {type(input_data)}")
    print("Transforming data...")

    return {
        'revenue_branch_data': transform_revenue_branch_data(input_data)
    }

transform_revenue_branch_data_op = PythonOperator(
    task_id='transform_revenue_branch_data',

```

```

    python_callable=transformed_revenue_data,
    provide_context=True,
    dag=dag
)

def transformed_warehouse_data(**kwargs):
    ti = kwargs['ti']

    input_data = ti.xcom_pull(task_ids='extract_warehouse_data')
    print(f"Extracted data type: {type(input_data)}")
    print("Transforming data...")

    return {
        'warehouse_data': transform_warehouse_data(input_data)
    }

transform_warehouse_data_op = PythonOperator(
    task_id='transform_warehouse_data',
    python_callable=transformed_warehouse_data,
    provide_context=True,
    dag=dag
)

def transformed_customer_data(**kwargs):
    ti = kwargs['ti']

    input_data = ti.xcom_pull(task_ids='extract_customer_data')
    print(f"Extracted data type: {type(input_data)}")
    print("Transforming data...")

```

```

        return {
            'customer_data': transform_customer_data(input_data)
        }

transform_customer_data_op = PythonOperator(
    task_id='transform_customer_data',
    python_callable=transformed_customer_data,
    provide_context=True,
    dag=dag
)

def transformed_revenue_product_data(**kwargs):
    ti = kwargs['ti']

    input_data = ti.xcom_pull(task_ids='extract_revenue_sp_data')
    print(f"Extracted data type: {type(input_data)}")
    print(f"Extracted data keys: {input_data.keys() if input_data else 'None'}")
    print("Transforming data...")

    if input_data and 'doanh_thu_sp_quy_cn_data' in input_data:
        df = input_data['doanh_thu_sp_quy_cn_data']
        print(f'DataFrame shape: {df.shape if hasattr(df, "shape") else "Not a DataFrame"}')

        if hasattr(df, 'empty') and not df.empty:
            transformed_df = transform_revenue_sp_quy_cn(df)
            return {
                'doanh_thu_sp_quy_cn_data': transformed_df
            }

```

```

else:
    print("DataFrame is empty")
    return {'doanh_thu_sp_quy_cn_data': {}}

else:
    print("No input data found for revenue product transformation")
    print(f"Available keys: {list(input_data.keys())} if input_data else 'None'")
    return {'doanh_thu_sp_quy_cn_data': {}}

transform_revenue_product_data_op = PythonOperator(
    task_id='transform_revenue_sp_data',
    python_callable=transformed_revenue_product_data,
    provide_context=True,
    dag=dag
)

def transformed_revenue_employee_data(**kwargs):
    ti = kwargs['ti']

    input_data = ti.xcom_pull(task_ids='extract_revenue_nv_data')
    print(f"Extracted data type: {type(input_data)}")
    print(f"Extracted data keys: {input_data.keys()} if input_data else 'None'")
    print("Transforming data...")

    # Fixed: Check for the correct key that extraction returns
    if input_data and 'doanh_thu_thang_nv_cn_data' in input_data:
        df = input_data['doanh_thu_thang_nv_cn_data']
        print(f"DataFrame shape: {df.shape} if hasattr(df, 'shape') else 'Not a DataFrame'")

        if hasattr(df, 'empty') and not df.empty:

```

```

        transformed_df = transform_revenue_thang_nv_cn(df)

        return {
            'doanh_thu_thang_nv_cn_data': transformed_df
        }

    else:
        print("DataFrame is empty")
        return {'doanh_thu_thang_nv_cn_data': {}}

else:
    print("No input data found for revenue employee transformation")
    print(f"Available keys: {list(input_data.keys())} if input_data else 'None'")

    return {'doanh_thu_thang_nv_cn_data': {}}

transform_revenue_employee_data_op = PythonOperator(
    task_id='transform_revenue_nv_data',
    python_callable=transformed_revenue_employee_data,
    provide_context=True,
    dag=dag
)

# Task 3: Load the transformed data to Cassandra
def load_invoice_data(**kwargs):
    ti = kwargs['ti']
    invoice_data = ti.xcom_pull(task_ids='transform_invoice_data')

    print("Loading chi_tiet_hoa_don_theo_ma_kh data ...")
    if invoice_data and 'invoice_data' in invoice_data:
        from load_to_cassandra import load_invoice_details_data_optimized

        load_invoice_details_data_optimized(invoice_data['invoice_data'].get('invoice_data'))

```

```

        print("Invoice data loaded successfully to Cassandra")
    else:
        print("No invoice data to load")

load_invoice_data_op = PythonOperator(
    task_id='load_invoice_data',
    python_callable=load_invoice_data,
    provide_context=True,
    dag=dag
)

def load_revenue_data(**kwargs):
    ti = kwargs['ti']
    revenue_data = ti.xcom_pull(task_ids='transform_revenue_branch_data')

    print("Loading doanh_thu_moi_ngay_theo_ma_cn data...")
    if revenue_data and 'revenue_branch_data' in revenue_data:
        from load_to_cassandra import load_revenue_data_optimized

        load_revenue_data_optimized(revenue_data['revenue_branch_data'].get('revenue_branch_data'))
        print("Revenue data loaded successfully to Cassandra")
    else:
        print("No revenue data to load")

load_revenue_branch_data_op = PythonOperator(
    task_id='load_revenue_branch_data',
    python_callable=load_revenue_data,
    provide_context=True,

```

```

dag=dag
)

def load_warehouse_data(**kwargs):
    ti = kwargs['ti']
    warehouse_data = ti.xcom_pull(task_ids='transform_warehouse_data')

    print("Loading kho_sp_theo_ma_cn data...")
    if warehouse_data and 'warehouse_data' in warehouse_data:
        from load_to_cassandra import load_wh_data_optimized
        load_wh_data_optimized(warehouse_data['warehouse_data'].get('warehouse_data'))
        print("Warehouse data loaded successfully to Cassandra")
    else:
        print("No warehouse data to load")

load_warehouse_data_op = PythonOperator(
    task_id='load_warehouse_data',
    python_callable=load_warehouse_data,
    provide_context=True,
    dag=dag
)

def load_customer_data(**kwargs):
    ti = kwargs['ti']
    customer_data = ti.xcom_pull(task_ids='transform_customer_data')

    print("Loading sl_khach_hang_moi_ngay_theo_ma_cn data...")
    if customer_data and 'customer_data' in customer_data:
        from load_to_cassandra import load_cus_data_optimized
        load_cus_data_optimized(customer_data['customer_data'].get('customer_data'))

```

```

        print("Customer data loaded successfully to Cassandra")
    else:
        print("No customer data to load")

load_customer_data_op = PythonOperator(
    task_id='load_customer_data',
    python_callable=load_customer_data,
    provide_context=True,
    dag=dag
)

def load_revenue_employee_data(**kwargs):
    ti = kwargs['ti']
    revenue_nv_data = ti.xcom_pull(task_ids='transform_revenue_nv_data')

    print("Loading 'doanh_thu_thang_nv_cn_data' ...")
    print(f'Retrieved data: {revenue_nv_data}') # Debug print

    if revenue_nv_data and 'doanh_thu_thang_nv_cn_data' in revenue_nv_data:
        from load_to_cassandra import load_doanhthu_nv_data_optimized
        transformed_data = revenue_nv_data['doanh_thu_thang_nv_cn_data']
        print(f'Transformed data: {transformed_data}') # Debug print

        if 'doanh_thu_thang_nv_cn' in transformed_data:
            load_doanhthu_nv_data_optimized(transformed_data['doanh_thu_thang_nv_cn'])
            print("doanh_thu_thang_nv_cn data loaded successfully to Cassandra")
        else:
            print("No 'doanh_thu_thang_nv_cn' data in transformed result")
    else:

```

```

print("No 'doanh_thu_thang_nv_cn_data' data to load")

load_revenue_nv_op = PythonOperator(
    task_id='load_revenue_nv_data',
    python_callable=load_revenue_employee_data,
    provide_context=True,
    dag=dag
)

def load_revenue_product_data(**kwargs):
    ti = kwargs['ti']
    revenue_sp_data = ti.xcom_pull(task_ids='transform_revenue_sp_data')

    print("Loading 'doanh_thu_sp_quy_cn' data...")
    print(f'Retrieved data: {revenue_sp_data}') # Debug print

    if revenue_sp_data and 'doanh_thu_sp_quy_cn_data' in revenue_sp_data:
        from load_to_cassandra import load_doanhthu_sp_data_optimized
        transformed_data = revenue_sp_data['doanh_thu_sp_quy_cn_data']
        print(f'Transformed data: {transformed_data}') # Debug print

        if 'doanh_thu_sp_quy_cn' in transformed_data:
            load_doanhthu_sp_data_optimized(transformed_data['doanh_thu_sp_quy_cn'])
            print('"doanh_thu_sp_quy_cn" data loaded successfully to Cassandra')
        else:
            print("No 'doanh_thu_sp_quy_cn' data in transformed result")
    else:
        print("No 'doanh_thu_sp_quy_cn' data to load")

```

```

load_revenue_product_op = PythonOperator(
    task_id='load_revenue_sp_data',
    python_callable=load_revenue_product_data,
    provide_context=True,
    dag=dag
)

def verify_completion(**kwargs):
    print("All data has been successfully loaded to Cassandra")
    return True

verification_op = PythonOperator(
    task_id='verification',
    python_callable=verify_completion,
    dag=dag
)

# Update task dependencies
setup_connections_op >> [extract_invoice_data_op, extract_revenue_branch_data_op,
extract_warehouse_data_op, extract_customer_data_op,
extract_revenue_product_data_op, extract_revenue_employee_data_op]

extract_invoice_data_op >> transform_invoice_data_op >> load_invoice_data_op
extract_revenue_branch_data_op >> transform_revenue_branch_data_op >>
load_revenue_branch_data_op
extract_warehouse_data_op >> transform_warehouse_data_op >>
load_warehouse_data_op
extract_customer_data_op >> transform_customer_data_op >> load_customer_data_op
extract_revenue_product_data_op >> transform_revenue_product_data_op >>

```

```

load_revenue_product_op
extract_revenue_employee_data_op >> transform_revenue_employee_data_op >>
load_revenue_nv_op

[load_invoice_data_op, load_revenue_branch_data_op, load_warehouse_data_op,
load_customer_data_op, load_revenue_product_op, load_revenue_nv_op] >>
verification_op

```

2. Ưu điểm và nhược điểm của việc di chuyển

2.1. Ưu điểm

- Workflow Orchestration và Task Management:
 - Airflow quản lý workflow phức tạp một cách rõ ràng.

```

# Task dependencies được định nghĩa rõ ràng
setup_connections_op >> [extract_invoice_data_op, extract_revenue_branch_data_op, extract_warehouse_data_op, extract_customer_data_op, extract_revenue_product_data_op, extract_revenue_nv_op]

# ETL pipeline cho từng data stream
extract_invoice_data_op >> transform_invoice_data_op >> load_invoice_data_op
extract_revenue_branch_data_op >> transform_revenue_branch_data_op >> load_revenue_branch_data_op
extract_warehouse_data_op >> transform_warehouse_data_op >> load_warehouse_data_op
extract_customer_data_op >> transform_customer_data_op >> load_customer_data_op
extract_revenue_product_data_op >> transform_revenue_product_data_op >> load_revenue_product_op
extract_revenue_employee_data_op >> transform_revenue_employee_data_op >> load_revenue_nv_op

[load_invoice_data_op, load_revenue_branch_data_op, load_warehouse_data_op, load_customer_data_op, load_revenue_product_op, load_revenue_nv_op] >> verification_op

```

Hình 4.12 Workflow Orchestration và Task Management

Lợi ích:

- Visual representation: DAG graph hiển thị workflow trực quan.
- Parallel execution: 6 data streams chạy song song sau setup.
- Clear dependencies: Mỗi transform phụ thuộc vào extract tương ứng.
- Error Handling và Retry Logic:

```

default_args = {
    'owner': 'hienfaang',
    'email_on_failure': True,           # Email khi task fail
    'email_on_retry': False,            # Không email khi retry
    'email': 'thuyhienphanthi2004@gmail.com',
    'retries': 3,                      # Retry 3 lần
    'retry_delay': timedelta(minutes=5), # Delay 5 phút giữa các retry
}

```

Hình 4.13 Error Handling và Retry Logic

Lợi ích:

- Automatic recovery: Tự động retry khi có lỗi tạm thời.
- Notification system: Thông báo email khi có lỗi nghiêm trọng.
- Granular control: Cấu hình retry khác nhau cho từng task.
- Data Flow Management với XCom: inter-task communication

```

def transformed_invoice_data(**kwargs):
    ti = kwargs['ti']
    # Pull data từ extract task
    input_data = ti.xcom_pull(task_ids='extract_invoice_data')
    print(f"Extracted data type: {type(input_data)}")

    # Transform và return
    return {
        'invoice_data': transform_invoice_data(input_data)
    }

def load_invoice_data(**kwargs):
    ti = kwargs['ti']
    # Pull data từ transform task
    invoice_data = ti.xcom_pull(task_ids='transform_invoice_data')

    if invoice_data and 'invoice_data' in invoice_data:
        load_invoice_details_data_optimized(invoice_data['invoice_data']).get('invoice_data')

```

Hình 4.14 Data Flow Management với XCom

Lợi ích:

- Seamless data passing: Dữ liệu truyền tự động giữa các tasks.
- Data validation: Kiểm tra data tại mỗi bước.
- Debugging friendly: Log chi tiết về data type và structure.
- Scheduling và Monitoring:

```

dag = DAG(
    dag_id='oracle_to_cassandra_migration',
    schedule_interval=None, # Manual trigger cho migration
    catchup=False,          # Không chạy lại historical runs
    tags=['migration', 'oracle', 'cassandra']
)

```

Hình 4.15 Scheduling và Monitoring

Lợi ích:

- Manual control: Migration chạy khi cần thiết.
- No historical baggage: catchup=False tránh chạy quá khứ.
- Organized tagging: Dễ tìm kiếm và phân loại.
- Modular và Reusable Code:

```

# Imports từ các modules riêng biệt
from extract_oracle_data import extract_invoice_data, extract_revenue_branch_data
from transform import transform_invoice_data, transform_revenue_branch_data
from setup_connections import setup_connections

```

Hình 4.16 Modular và Reusable Code

Lợi ích:

- Separation of concerns: Logic tách biệt khỏi orchestration.
- Testability: Có thể test từng function độc lập.
- Reusability: Functions có thể dùng lại cho DAGs khác.
- Tính năng monitoring:
 - Task status tracking: Success/Failed/Running states.
 - Execution logs: Chi tiết logs cho debugging.
 - Gantt charts: Thời gian execution của từng task.
 - Graph view: Visual representation của workflow.

2.2. Nhược điểm

- Hạn chế về truyền tải và lưu trữ dữ liệu nội bộ (XCom Storage Limitations):

```

# Có thể thấy data được pass qua nhiều layers
input_data = ti.xcom_pull(task_ids='extract_invoice_data')
# → Transform
transformed_data = transform_invoice_data(input_data)
# → Load
load_invoice_details_data_optimized(transformed_data)

```

Hình 4.17 XCom Storage Limitation

- Giới hạn kích thước: Cơ chế XCom của Airflow (thường dùng để truyền dữ liệu nhỏ giữa các tác vụ) có giới hạn kích thước nghiêm ngặt (thường là 1MB). Điều này làm cho việc sử dụng XCom để truyền tải dữ liệu thực tế (payload) từ Oracle sang Cassandra là không khả thi do khối lượng dữ liệu lớn trong di chuyển.
- Chi phí tuân tự hóa (Serialization overhead): Việc chuyển đổi các cấu trúc dữ liệu phức tạp như Pandas DataFrame thành định dạng có thể truyền qua XCom (và ngược lại) rất chậm, làm giảm hiệu suất tổng thể của quá trình di chuyển.
- Tiêu thụ bộ nhớ: Nếu cố gắng tải toàn bộ các tập dữ liệu lớn vào bộ nhớ trong quá trình xử lý (dù là để truyền hay chỉ để xử lý trong một tác vụ), có nguy cơ gây ra tình trạng "bộ nhớ tăng đột biến" (memory spikes), dẫn đến lỗi Out-of-Memory hoặc làm chậm hệ thống do phải sử dụng bộ nhớ ảo.
- Vấn đề quản lý tài nguyên (Resource Management Issues):

```

# Setup connections chạy trước tất cả tasks
setup_connections_op >> [all_extract_tasks]

```

Hình 4.18 Resource Management Issues

- Không tự động quản lý kết nối: Airflow không tự động quản lý kết nối cơ sở dữ liệu (ví dụ: pooling kết nối). Điều này đòi hỏi người phát triển phải tự quản lý việc mở và đóng kết nối tới cả Oracle và Cassandra một cách cẩn thận, dễ dẫn đến lỗi.

- Rò rỉ tài nguyên: Nguy cơ các kết nối tới Oracle hoặc Cassandra không được đóng đúng cách, gây rò rỉ tài nguyên và có thể làm cạn kiệt số lượng kết nối cho phép trên các hệ thống cơ sở dữ liệu, ảnh hưởng đến hoạt động của các ứng dụng khác.
 - Quá tải cơ sở dữ liệu: Khi nhiều tác vụ chạy song song để di chuyển dữ liệu, chúng có thể đồng thời truy vấn Oracle và ghi vào Cassandra với cường độ cao, dễ gây quá tải cho cả hai hệ thống, ảnh hưởng đến hiệu suất và tính ổn định của cơ sở dữ liệu sản xuất.
- Độ phức tạp trong phát triển và gỡ lỗi (Development và Debugging Complexity):

```
# Nhiều wrapper functions chỉ để handle XCom
def transformed_invoice_data(**kwargs):
    ti = kwargs['ti']
    input_data = ti.xcom_pull(task_ids='extract_invoice_data')
    return {
        'invoice_data': transform_invoice_data(input_data) # Actual logic
    }
```

Hình 4.19 Development và Debugging Complexity

- Mã rập khuôn (Boilerplate code): Yêu cầu viết nhiều hàm bao bọc (wrapper functions) hoặc đoạn mã lặp lại để xử lý việc truyền dữ liệu giữa các tác vụ (thay thế cho XCom hoặc để thao tác với XCom cho các thông điệp điều khiển), làm tăng số lượng mã không liên quan đến nghiệp vụ.
 - Khó khăn trong kiểm thử: Việc giả lập (mock) hành vi của XCom và các tương tác Airflow-specific khác để kiểm thử các tác vụ di chuyển dữ liệu trở nên phức tạp, làm chậm quá trình phát triển và tăng nguy cơ lỗi khi triển khai thực tế.
 - Hỗ trợ IDE hạn chế: Các môi trường phát triển tích hợp (IDE) thường có hỗ trợ Intellisense (gợi ý mã) hạn chế cho các mã đặc trưng của Airflow, làm giảm năng suất code và tăng khả năng mắc lỗi cú pháp.
- Nút thắt cổ chai về hiệu suất (Performance Bottlenecks):

```

# Mỗi task xử lý tuần tự
for _, row in invoice_data.iterrows(): # Row-by-row processing
    params = [row['ma_khach_hang'], row['ma_hoa_don'], ...]
    params_list.append(params)

```

Hình 4.20 Performance Bottlenecks

- Tác vụ đơn luồng (Single-threaded tasks): Các tác vụ trong Airflow thường chạy trên một luồng đơn. Đối với việc di chuyển dữ liệu lớn, điều này hạn chế khả năng tận dụng tối đa tài nguyên CPU và làm chậm tổng thời gian hoàn thành quá trình di chuyển, trừ khi pipeline được thiết kế đặc biệt để chạy song song.
- Tắc nghẽn I/O: Các hoạt động đọc từ Oracle và ghi vào Cassandra là các hoạt động I/O nặng. Nếu các tác vụ bị chặn (blocking) trong khi chờ kết quả từ cơ sở dữ liệu, hiệu suất tổng thể của pipeline sẽ bị giảm đáng kể.
- Độ phức tạp vận hành (Operational Complexity):
Yêu cầu nhiều thành phần: Để duy trì một pipeline di chuyển dữ liệu bằng Airflow, cần phải quản lý và vận hành nhiều thành phần riêng biệt như Airflow webserver, Airflow scheduler và một cơ sở dữ liệu để lưu trữ metadata. Điều này làm tăng độ phức tạp của hạ tầng và yêu cầu chi phí bảo trì.

3. Thực hiện truy vấn để so sánh hiệu suất truy vấn giữa cơ sở dữ liệu quan hệ và Cassandra

3.1. Kết quả đối với RDBMS

❖ Kết quả chạy câu truy vấn ở Máy 1 (Chi nhánh 1)

```

SELECT /*+ GATHER_PLAN_STATISTICS */
hd."MaKhachHang",
cthd."MaHoaDon",
cthd."MaSanPham",

```

```
cthds."SoLuong",
cthds."ThanhTien",
hd."TongTien",
hd."NgayTao",
hd."PhuongThucThanhToan",
hd."MaNhanVien"

FROM BTL1."ChiTietHoaDon"@NhanVien12Link cthd
JOIN BTL1."HoaDon"@NhanVien12Link hd ON cthd."MaHoaDon" =
hd."MaHoaDon"
WHERE hd."MaKhachHang" IN (34, 38)
ORDER BY hd."MaHoaDon", hd."NgayTao" DESC;
```

-- Tìm SQL_ID của câu truy vấn tối ưu vừa thực thi

```
SELECT sql_id, sql_text
FROM v$sql
WHERE sql_text LIKE '%hd."MaKhachHang%"' AND sql_text NOT LIKE
'%v$transaction%'
ORDER BY last_load_time DESC;
```

-- Chạy EXPLAIN câu truy vấn có SQL_ID vừa tìm được

```
SELECT * FROM TABLE(DBMS_XPLAN.display_cursor('6atn9j106td21', NULL,
'ALLSTATS LAST'));
```

PLAN_TABLE_OUTPUT VARCHAR2(4000)
SQL_ID 6atn9j106td21, child number 0
<pre>SELECT /*+ GATHER_PLAN_STATISTICS */ hd."MaKhachHang", cthd."MaHoaDon", cthd."MaSanPham", cthd."SoLuong", cthd."ThanhTien", hd."TongTien", hd."NgayTao", hd."PhuongThucThanhToan", hd."MaNhanVien" FROM BTL1."ChiTietHoaDon"@NhanVien12Link cthd JOIN BTL1."HoaDon"@NhanVien12Link hd ON cthd."MaHoaDon" = hd."MaHoaDon" WHERE hd."MaKhachHang" IN (34, 38) ORDER BY hd."MaHoaDon", hd."NgayTao" DESC</pre>
NOTE: cannot fetch plan for SQL_ID: 6atn9j106td21, CHILD_NUMBER: 0
Please verify value of SQL_ID and CHILD_NUMBER;
It could also be that the plan is no longer in cursor cache (check v\$sql_plan)

Hình 4.21 Kết quả EXPLAIN câu truy vấn ở RDBMS (CNI)

```

DECLARE
  v_sql      CLOB;
  v_start_time TIMESTAMP;
  v_end_time  TIMESTAMP;
  v_dummy     NUMBER;

BEGIN
  v_sql := '
    SELECT
      hd."MaKhachHang",
      cthd."MaHoaDon",
      cthd."MaSanPham",
      cthd."SoLuong",
      cthd."ThanhTien",
      hd."TongTien",
      hd."NgayTao",

```

```

hd."PhuongThucThanhToan",
hd."MaNhanVien"
FROM BTL1."ChiTietHoaDon"@NhanVien12Link cthd
JOIN BTL1."HoaDon"@NhanVien12Link hd ON cthd."MaHoaDon" =
hd."MaHoaDon"
WHERE hd."MaKhachHang" IN (34, 38)
ORDER BY hd."MaHoaDon", hd."NgayTao" DESC
';

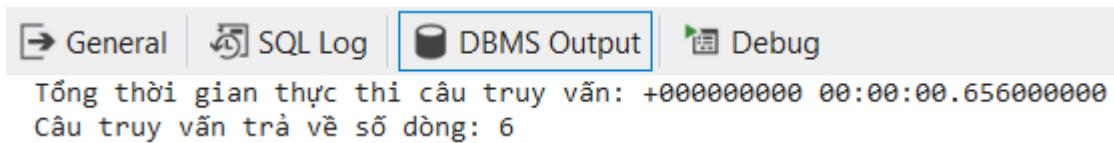
v_start_time := SYSTIMESTAMP;

EXECUTE IMMEDIATE 'SELECT COUNT(DISTINCT "MaHoaDon") FROM (' ||
v_sql || ') t' INTO v_dummy;

v_end_time := SYSTIMESTAMP;

DBMS_OUTPUT.PUT_LINE('Tổng thời gian thực thi câu truy vấn: ' ||
TO_CHAR(v_end_time - v_start_time));
DBMS_OUTPUT.PUT_LINE('Câu truy vấn trả về số dòng: ' || v_dummy);
END;

```



Hình 4.22 Kết quả thời gian thực thi câu truy vấn ở RDBMS (CN1)

❖ Kết quả chạy câu truy vấn ở Máy 2 (Chi nhánh 2)

```

SELECT /*+ GATHER_PLAN_STATISTICS */
hd."MaKhachHang",

```

```
cthd."MaHoaDon",
cthd."MaSanPham",
cthd."SoLuong",
cthd."ThanhTien",
hd."TongTien",
hd."NgayTao",
hd."PhuongThucThanhToan",
hd."MaNhanVien"

FROM BTL1."ChiTietHoaDon"@NhanVien21Link cthd
JOIN BTL1."HoaDon"@NhanVien21Link hd ON cthd."MaHoaDon" =
hd."MaHoaDon"
WHERE hd."MaKhachHang" IN (34, 38)
ORDER BY hd."MaHoaDon", hd."NgayTao" DESC;
```

-- Tìm SQL_ID của câu truy vấn tối ưu vừa thực thi

```
SELECT sql_id, sql_text
FROM v$sql
WHERE sql_text LIKE '%hd."MaKhachHang%"' AND sql_text NOT LIKE
'%v$transaction%'
ORDER BY last_load_time DESC;
```

-- Chạy EXPLAIN câu truy vấn có SQL_ID vừa tìm được

```
SELECT * FROM TABLE(DBMS_XPLAN.display_cursor('chgm2yka7c2wc', NULL,
'ALLSTATS LAST'));
```

PLAN_TABLE_OUTPUT
VARCHAR2(4000)
SQL_ID chgm2yka7c2wc, child number 0
SELECT /*+ GATHER_PLAN_STATISTICS */ hd."MaKhachHang", cthd."MaHoaDon", cthd."MaSanPham", cthd."SoLuong", cthd."ThanhTien", hd."TongTien", hd."NgayTao", hd."PhuongThucThanhToan", hd."MaNhanVien" FROM BTL1."ChiTietHoaDon"@NhanVien21Link cthd JOIN BTL1."HoaDon"@NhanVien21Link hd ON cthd."MaHoaDon" = hd."MaHoaDon" WHERE hd."MaKhachHang" IN (34, 38) ORDER BY hd."MaHoaDon", hd."NgayTao" DESC
NOTE: cannot fetch plan for SQL_ID: chgm2yka7c2wc, CHILD_NUMBER: 0
Please verify value of SQL_ID and CHILD_NUMBER;
It could also be that the plan is no longer in cursor cache (check v\$sql_plan)

Hình 4.23 Kết quả EXPLAIN câu truy vấn ở RDBMS (CN2)

```

DECLARE
v_sql      CLOB;
v_start_time TIMESTAMP;
v_end_time  TIMESTAMP;
v_dummy     NUMBER;
BEGIN
v_sql := '
SELECT
hd."MaKhachHang",
cthhd."MaHoaDon",
cthhd."MaSanPham",
cthhd."SoLuong",
cthhd."ThanhTien",
hd."TongTien",
hd."NgayTao",

```

```

        hd."PhuongThucThanhToan",
        hd."MaNhanVien"
FROM BTL1."ChiTietHoaDon"@NhanVien21Link cthd
JOIN BTL1."HoaDon"@NhanVien21Link hd ON cthd."MaHoaDon" =
hd."MaHoaDon"
WHERE hd."MaKhachHang" IN (34, 38)
ORDER BY hd."MaHoaDon", hd."NgayTao" DESC
';

v_start_time := SYSTIMESTAMP;

EXECUTE IMMEDIATE 'SELECT COUNT(DISTINCT "MaHoaDon") FROM (' ||
v_sql || ') t' INTO v_dummy;

v_end_time := SYSTIMESTAMP;

DBMS_OUTPUT.PUT_LINE('Tổng thời gian thực thi câu truy vấn: ' ||
TO_CHAR(v_end_time - v_start_time));
DBMS_OUTPUT.PUT_LINE('Câu truy vấn trả về số dòng: ' || v_dummy);
END;

```



Hình 4.24 Kết quả thời gian thực thi câu truy vấn ở RDBMS (CN2)

3.2. Kết quả đối với Cassandra

- ❖ Kết quả chạy câu truy vấn ở Máy 1 (Chi nhánh 1)

File cluster_queries.py

```
from connect_2_clusters import connect_to_cluster
from cassandra.query import SimpleStatement
from tabulate import tabulate

cluster_ip = '26.93.36.133'
keyspace_name = 'bt12_data'

try:
    cluster, session = connect_to_cluster(cluster_ip, keyspace_name)

    print("\n=====Kiểm tra kết nối 2 máy thuộc 2 cluster=====")

    query = "SELECT * FROM chi_tiet_hoa_don_theo_ma_kh WHERE ma_khach_hang IN (34, 38);"
    print(f"\nMáy 1 đang thực hiện câu truy vấn: {query}\n")

    statement = SimpleStatement(query)
    rows = session.execute(statement, trace=True)
    row_list = list(rows) # <-- Chuyển sang list để đếm và xử lý

    print(f"Số dòng của kết quả trả về: {len(row_list)}")

    if not row_list:
        print("No results found")
    else:
        headers = rows.column_names
        print(tabulate(row_list, headers=headers, tablefmt='fancy_grid'))

        trace = rows.get_query_trace()
        print("\n--- Query tracing ---")
        for event in trace.events:
            print(f"{event.source} - {event.description} - {event.source_elapsed}μs")

except Exception as e:
    print(f"Error when doing queries: {e}")

finally:
    if 'cluster' in locals():
        cluster.shutdown()
```

Hình 4.25 Chạy câu truy vấn ở Cassandra dùng code Python (CN1)

```

D:\Nam3_HK2\CoSoDuLieuPhanTan\DoAn\BTL2\migrate_oracle_cassandra>python cluster_queries.py
True
Connected to cluster_2 having IP address: 26.93.36.133 successfully!
Attempting to connect to BTL2_data keyspace...
Connected to btl2_data successfully!
=====Kiểm tra kết nối 2 máy thuộc 2 cluster=====
Máy 2 đang thực hiện câu truy vấn: SELECT * FROM chi_tiet_hoa_don_theo_ma_kh WHERE ma_khachhang IN (34, 38);
Số dòng của kết quả trả về: 6


| ma_khachhang | ngay_tao            | ma_hoa_don | ma_nhan_vien | ma_san_pham | phuong_thuc_thanh_toan | so_luong | thanh_tien | tong_tien |
|--------------|---------------------|------------|--------------|-------------|------------------------|----------|------------|-----------|
| 34           | 2023-07-19 06:59:11 | 471        | 897          | CCNPLT0008  | Ngân Hàng              | 9        | 3060000    | 4500000   |
| 34           | 2022-11-24 13:25:18 | 477        | 695          | CCNPOT0141  | Ngân Hàng              | 5        | 2450000    | 3530000   |
| 38           | 2025-08-03 04:46:15 | 539        | 745          | CCNPOT0139  | Ngân Hàng              | 1        | 400000     | 14930000  |
| 38           | 2025-03-31 23:53:22 | 531        | 653          | CCNPOT0170  | Ngân Hàng              | 1        | 3500000    | 7620000   |
| 38           | 2023-06-26 09:18:38 | 540        | 947          | CCNPLT0005  | Ngân Hàng              | 7        | 8400000    | 16840000  |
| 38           | 2023-03-20 11:42:28 | 547        | 750          | CCNPOT0019  | Tiền Mặt               | 10       | 2000000    | 2000000   |


--- Query tracing ---
172.21.0.5 - Parsing SELECT * FROM chi_tiet_hoa_don_theo_ma_kh WHERE ma_khachhang IN (34, 38); - 0:00:00.001223µs
172.21.0.5 - Preparing statement - 0:00:00.001979µs
172.21.0.5 - Executing single-partition query on chi_tiet_hoa_don_theo_ma_kh - 0:00:00.004468µs
172.21.0.5 - Acquiring sstable references - 0:00:00.004605µs
172.21.0.5 - Skipped @/1 non-slice-intersecting stables, included 0 due to tombstones - 0:00:00.004812µs
172.21.0.5 - Key cache hit for sstable 33 - 0:00:00.005084µs
172.21.0.5 - Merged data from memtables and 1 stables - 0:00:00.005591µs
172.21.0.5 - Read 2 live rows and 0 tombstone cells - 0:00:00.005826µs
172.21.0.5 - Executing single-partition query on chi_tiet_hoa_don_theo_ma_kh - 0:00:00.006681µs
172.21.0.5 - Acquiring sstable references - 0:00:00.006775µs
172.21.0.5 - Skipped @/1 non-slice-intersecting stables, included 0 due to tombstones - 0:00:00.006867µs
172.21.0.5 - Key cache hit for sstable 33 - 0:00:00.007035µs
172.21.0.5 - Merged data from memtables and 1 stables - 0:00:00.007390µs
172.21.0.5 - Read 4 live rows and 0 tombstone cells - 0:00:00.007503µs

```

Hình 4.26 Kết quả chạy câu truy vấn ở Cassandra dùng code Python (CN1)

- ❖ Kết quả chạy câu truy vấn ở Máy 2 (Chi nhánh 2)

```

from connect_2_clusters import connect_to_cluster
from cassandra.query import SimpleStatement      Import "cassandra.query" could not be resolved
from tabulate import tabulate      Import "tabulate" could not be resolved from source

cluster_ip = '26.103.246.194'
keyspace_name = 'bt12_data'

try:
    cluster, session = connect_to_cluster(cluster_ip, keyspace_name)

    print("\n=====Kiểm tra kết nối 2 máy thuộc 2 cluster=====")

    query = "SELECT * FROM chi_tiet_hoa_don_theo_ma_kh WHERE ma_khach_hang IN (34, 38);"
    print(f"\nMáy 2 đang thực hiện câu truy vấn: {query}\n")

    statement = SimpleStatement(query)
    rows = session.execute(statement, trace=True)
    row_list = list(rows) # <-- Chuyển sang list để đếm và xử lý

    print(f"Số dòng của kết quả trả về: {len(row_list)}")

    if not row_list:
        print("No results found")
    else:
        headers = rows.column_names
        print(tabulate(row_list, headers=headers, tablefmt="fancy_grid"))

        trace = rows.get_query_trace()
        print("\n--- Query tracing ---")
        for event in trace.events:
            print(f"{event.source} - {event.description} - {event.source_elapsed}μs")

except Exception as e:
    print(f"Error when doing queries: {e}")

finally:
    if 'cluster' in locals():
        cluster.shutdown()

```

File cluster_queries.py

Hình 4.27 Chạy câu truy vấn ở Cassandra dùng code Python (CN2)

```

Connected to cluster_1 having IP address: 26.103.246.194 successfully!
Attempting to connect to BTL2_data keyspace...
Connected to bt12_data successfully!
=====Kiểm tra kết nối 2 máy thuộc 2 cluster=====
Máy 2 đang thực hiện câu truy vấn: SELECT * FROM chi_tiet_hoa_don_theo_ma_kh WHERE ma_khachhang IN (34, 38);
Số dòng của kết quả trả về: 11


| ma_khachhang | ngay_tao            | ma_hoa_don | ma_nhan_vien | ma_san_pham | phuong_thuc_thanh_toan | so_luong | thanh_tien | tong_tien |
|--------------|---------------------|------------|--------------|-------------|------------------------|----------|------------|-----------|
| 34           | 2025-01-05 19:07:59 | 473        | 319          | CNPLT0177   | Tiền Mặt               | 10       | 5500000    | 13530000  |
| 34           | 2024-04-27 23:21:00 | 472        | 400          | CNPLT0223   | Tiền Mặt               | 9        | 3060000    | 3060000   |
| 34           | 2023-08-03 23:39:27 | 474        | 223          | CNPLT0214   | Ngân Hàng              | 9        | 2520000    | 3660000   |
| 38           | 2025-12-07 04:05:27 | 546        | 198          | CNPLT0224   | Tiền Mặt               | 1        | 990000     | 19167200  |
| 38           | 2025-11-26 17:51:46 | 536        | 156          | CNPLT0183   | Tiền Mặt               | 8        | 12400000   | 12400000  |
| 38           | 2025-05-21 03:26:14 | 535        | 10           | CNPLT0434   | Ngân Hàng              | 6        | 3000000    | 18975000  |
| 38           | 2024-11-04 21:27:08 | 541        | 17           | CNPLT0417   | Ngân Hàng              | 4        | 480000     | 7262882   |
| 38           | 2024-09-02 03:05:47 | 537        | 259          | CNPLT0576   | Tiền Mặt               | 1        | 55000      | 1435000   |
| 38           | 2023-06-22 01:04:08 | 545        | 161          | CNPOT0168   | Tiền Mặt               | 4        | 480000     | 5550000   |
| 38           | 2022-11-02 05:38:46 | 551        | 386          | CNPOT0178   | Tiền Mặt               | 1        | 450000     | 3190000   |
| 38           | 2022-09-13 11:49:17 | 542        | 407          | CNPOT0116   | Tiền Mặt               | 5        | 9250000    | 13431334  |



— Query tracing —



```

172.18.0.3 - Parsing SELECT * FROM chi_tiet_hoa_don_theo_ma_kh WHERE ma_khachhang IN (34, 38); - 0:00:00.006707μs
172.18.0.3 - Preparing statement - 0:00:00.007569μs
172.18.0.3 - Executing single-partition query on chi_tiet_hoa_don_theo_ma_kh - 0:00:00.009000μs
172.18.0.3 - Acquiring sstable references - 0:00:00.009164μs
172.18.0.3 - Skipped 0/3 non-slice-intersecting stables, included 0 due to tombstones - 0:00:00.009777μs
172.18.0.3 - Bloom filter allows skipping sstable 2 - 0:00:00.010529μs
172.18.0.3 - Bloom filter allows skipping sstable 3 - 0:00:00.010912μs
172.18.0.3 - Key cache hit for sstable 1 - 0:00:00.011400μs
172.18.0.3 - Merged data from memtables and 1 sstables - 0:00:00.013567μs

```


```

Hình 4.28 Kết quả chạy câu truy vấn ở Cassandra dùng code Python (CN2)

3.3. So sánh và nhận xét

❖ Máy 1 (Chi nhánh 1)

- Oracle:

- + Thời gian thực thi: khoảng 0.65 giây (656ms) - đây là thời gian khá nhanh đối với một truy vấn có join và order by, qua database link.
- + Số dòng trả về: 6

- Cassandra:

- + Số dòng trả về là 6, đúng với số dòng đã lấy từ Oracle.
- + Thời gian thực thi cho mỗi bước là rất nhỏ (micro giây đến vài mili giây), thể hiện truy vấn rất nhanh và hiệu quả.
- + Các dòng log truy vấn ở dưới thể hiện quá trình Cassandra xử lý truy vấn:
 - Parsing SELECT: Cassandra phân tích cú pháp truy vấn.

- Preparing statement: Chuẩn bị câu lệnh SQL (prepared statement).
- Executing single-partition query: Truy vấn được thực thi trên single partition (một phân vùng dữ liệu). Điều này nghĩa là bảng được thiết kế với partition key phù hợp.
- Acquiring sstable references: Cassandra đọc dữ liệu từ các sstable (file lưu dữ liệu vật lý).
- Key cache hit: Bộ nhớ đệm khóa được truy cập thành công, giúp tăng tốc đọc.
- Merging data from memtables and sstables: Cassandra kết hợp dữ liệu trong bộ nhớ đệm (memtable) và trên đĩa (sstable).
- Read live rows and tombstone cells: Đọc các bản ghi còn tồn tại (live rows), tombstone cells (bản ghi bị xóa) là 0, nghĩa là không có dữ liệu bị xóa ảnh hưởng.

- **Kết luận:**

- + Câu truy vấn chạy trên Oracle cũng trả về 6 dòng, tương đương Cassandra.
- + Thời gian Oracle thực thi khoảng 0.65 giây, trong khi Cassandra ghi lại truy vấn chi tiết rất nhanh (chỉ micro giây cho mỗi bước truy xuất).
- + Điều này phản ánh điểm mạnh của Cassandra trong việc xử lý truy vấn theo partition key với độ trễ thấp.

❖ **Máy 2 (Chi nhánh 2)**

- **Oracle:**

- + Thời gian thực thi: khoảng 0.75 giây (750 ms) - đây là thời gian khá nhanh đối với một truy vấn có join và order by, qua database link.
- + Số dòng trả về: 11

- **Cassandra:**

- + Số dòng trả về là 11, đúng với số dòng đã lấy từ Oracle.
- + Thời gian thực thi cho mỗi bước là rất nhỏ (micro giây đến vài mili giây), thể hiện truy vấn rất nhanh và hiệu quả.
- + Các dòng log truy vấn ở dưới thể hiện quá trình Cassandra xử lý truy vấn:
 - Parsing SELECT: Cassandra phân tích cú pháp truy vấn.
 - Preparing statement: Chuẩn bị câu lệnh SQL (prepared statement).
 - Executing single-partition query: Truy vấn được thực thi trên single partition (một phân vùng dữ liệu). Điều này nghĩa là bảng được thiết kế với partition key phù hợp.
 - Acquiring sstable references: Cassandra đọc dữ liệu từ các sstable (file lưu dữ liệu vật lý).
 - Key cache hit: Bộ nhớ đệm khóa được truy cập thành công, giúp tăng tốc đọc.
 - Merging data from memtables and sstables: Cassandra kết hợp dữ liệu trong bộ nhớ đệm (memtable) và trên đĩa (sstable).
 - Read live rows and tombstone cells: Đọc các bản ghi còn tồn tại (live rows), tombstone cells (bản ghi bị xóa) là 0, nghĩa là không có dữ liệu bị xóa ảnh hưởng.

- **Kết luận:**

- + Câu truy vấn chạy trên Oracle cũng trả về 11 dòng, tương đương Cassandra.
- + Thời gian Oracle thực thi khoảng 0.75 giây, trong khi Cassandra ghi lại truy vấn chi tiết rất nhanh (chỉ micro giây cho mỗi bước truy xuất).
- + Oracle có thể chậm hơn do join, sắp xếp, và đặc điểm của database link.

- + Cassandra tối ưu truy vấn theo partition key rất tốt, giảm đáng kể thời gian truy vấn khi dữ liệu được phân vùng đúng.

CHƯƠNG 5. THAO TÁC DỮ LIỆU GIỮA CÁC MÁY

1. Thêm dữ liệu

- *Trường hợp demo thêm dữ liệu vào Máy 1 (Chi nhánh 1):*

- + Kiểm tra doanh thu của ngày 2/1/2021 ở Chi nhánh 1 (CN1) là bao nhiêu. Kết quả cho thấy chưa có dữ liệu doanh thu ngày 2/1/2021 ở CN1. Máy 2 tiến hành thao tác thêm doanh thu ngày 2/1/2021 vào cơ sở dữ liệu ở CN1.

```
cqlsh:btl2_data> select * from doanh_thu_moi_ngay_theo_ma_cn where ma_chi_nhanh= 1 and ngay='2021-01-02';

ma_chi_nhanh | ngay | tong_tien
-----+-----+
(0 rows)
```

Hình 5.1 Terminal CN1 kiểm tra sự tồn tại của doanh thu ngày 2/1/2021

```
cqlsh:btl2_data> select * from doanh_thu_moi_ngay_theo_ma_cn where ma_chi_nhanh= 2 and ngay='2021-01-02';

ma_chi_nhanh | ngay | tong_tien
-----+-----+
(0 rows)
```

Hình 5.2 Terminal CN2 kiểm tra sự tồn tại của doanh thu ngày 2/1/2021

- + Code Python.

```
import os
import sys
from connect_2_clusters import connect_to_cluster
from cassandra.query import SimpleStatement
from tabulate import tabulate
from cassandra.cluster import Cluster

sys.path.append('/opt/airflow/scripts')
sys.path.append(os.path.join(os.path.dirname(__file__), '..', 'scripts'))
```

```

from load_to_cassandra import get_cassandra_session

cluster_ip = '26.103.246.194'
keyspace_name = 'btl2_data'

def crud_ops(query, cluster_sessions, table_name, condition, insert):
    stm = SimpleStatement(query)
    check_query = f"""
        SELECT COUNT(*) FROM {table_name} WHERE {condition};
    """

    if insert:
        cluster_sessions[0].execute(stm)
        print("Inserted successfully!")
        return True

    success = False
    for i in range(len(cluster_sessions)):
        try:
            tmp_check = cluster_sessions[i].execute(check_query)
            if tmp_check.one()[0] > 0:
                cluster_sessions[i].execute(stm)
                success = True
                print('Operation is executed successfully!')
            else:
                print(f"No matching records found in session {i}")
        except Exception as e:
            print(f"Error executing on remote session: {e}")

```

```

        continue

    if not success:
        print("No matching data found in any session")
        return False

    try:
        remote_cluster, remote_session = connect_to_cluster(cluster_ip, keyspace_name)
        my_cluster, my_session = connect_to_cluster('127.0.0.1', keyspace_name)

        cluster_sessions = [
            my_session,
            remote_session
        ]

        print("\n=====Kiểm tra kết nối 2 máy thuộc 2 cluster=====")

        # insert
        insert_query = f"""
        INSERT INTO doanh_thu_moi_ngay_theo_ma_cn (
            ma_chi_nhanh,
            ngay,
            tong_tien
        ) VALUES (
            1,
            '2021-01-02',
            123000456
        )
        """

```

```

insert_rs = crud_ops(
    query=insert_query,
    cluster_sessions=[remote_session],
    table_name='doanh_thu_moi_ngay_theo_ma_cn',
    condition=None,
    insert=True
)
if not insert_rs:
    print("Executed failed!")

except Exception as e:
    print(f"Error when doing queries: {e}")

finally:
    if 'remote_cluster' in locals():
        remote_cluster.shutdown()
    if 'my_cluster' in locals():
        my_cluster.shutdown()

```

```

pwsh python3 .\crud.py
True
Connected to cluster_1 having IP address: 26.103.246.194 successfully!

Attempting to connect to BTL2_data keyspace...
Connected to btl2_data successfully!
True
Connected to cluster_1 having IP address: 127.0.0.1 successfully!

Attempting to connect to BTL2_data keyspace...
Connected to btl2_data successfully!

=====Kiểm tra kết nối 2 máy thuộc 2 cluster=====
Inserted successfully!

```

Hình 5.3 Terminal ở CN2 chạy file code Python thực hiện insert

- + Máy 1 (CN1) kiểm tra xem Máy 2 có thực hiện thêm doanh thu ngày 2/6/2025 vào CN1 thành công hay chưa.

```
cqlsh:btl2_data> select * from doanh_thu_moi_ngay_theo_ma_cn where ma_chi_nhanh= 1 and ngay='2021-01-02';

ma_chi_nhanh | ngay      | tong_tien
-----+-----+-----
1 | 2021-01-02 | 123000456

(1 rows)
```

Hình 5.4 Terminal ở CN1 kiểm tra kết quả insert của CN2

- + Để kiểm tra tính đúng đắn của thao tác thêm là chỉ thêm record vào CN1 thì CN2 kiểm tra lại thông tin.

```
cqlsh:btl2_data> select * from doanh_thu_moi_ngay_theo_ma_cn where ma_chi_nhanh= 2 and ngay='2021-01-02';

ma_chi_nhanh | ngay | tong_tien
-----+-----+-----
(0 rows)

cqlsh:btl2_data>
cqlsh:btl2_data> select * from doanh_thu_moi_ngay_theo_ma_cn where ma_chi_nhanh= 1 and ngay='2021-01-02';

ma_chi_nhanh | ngay | tong_tien
-----+-----+-----
(0 rows)
```

Hình 5.5 Terminal CN2 kiểm tra tính đúng đắn của thao tác insert

- **Trường hợp demo thêm dữ liệu vào Máy 2 (Chi nhánh 2):**

- + Kiểm tra doanh thu của ngày 3/1/2021 ở Chi nhánh 2 (CN2) là bao nhiêu. Kết quả cho thấy chưa có dữ liệu doanh thu ngày 3/1/2021 ở CN2. Máy 1 tiến hành thao tác thêm doanh thu ngày 3/1/2021 vào cơ sở dữ liệu ở CN2.

```
cqlsh:btl2_data> select * from doanh_thu_moi_ngay_theo_ma_cn where ma_chi_nhanh= 2 and ngay='2021-01-03';

ma_chi_nhanh | ngay | tong_tien
-----+-----+-----
(0 rows)
```

Hình 5.6 Terminal CN2 kiểm tra sự tồn tại của doanh thu ngày 3/1/2021

```
cqlsh:btl2_data> select * from doanh_thu_moi_ngay_theo_ma_cn where ma_chi_nhanh= 1 and ngay='2021-01-03';

ma_chi_nhanh | ngay | tong_tien
-----+-----+
(0 rows)
```

Hình 5.7 Terminal CSDL kiểm tra sự tồn tại của doanh thu ngày 3/1/2021

+ Code Python.

```
import os
import sys
from connect_2_clusters import connect_to_cluster
from cassandra.query import SimpleStatement
from tabulate import tabulate
from cassandra.cluster import Cluster

sys.path.append('/opt/airflow/scripts')
sys.path.append(os.path.join(os.path.dirname(__file__), '..', 'scripts'))

from load_to_cassandra import get_cassandra_session

cluster_ip = '26.93.36.133'
keyspace_name = 'btl2_data'

def crud_ops(query, cluster_sessions, table_name, condition, insert):
    stm = SimpleStatement(query)
    check_query = f"""
        SELECT COUNT(*) FROM {table_name} WHERE {condition};
    """
    if insert:
        cluster_sessions[0].execute(stm)
```

```

        print("Inserted successfully!")
        return True

success = False
for i in range(len(cluster_sessions)):
    try:
        tmp_check = cluster_sessions[i].execute(check_query)
        if tmp_check.one()[0] > 0:
            cluster_sessions[i].execute(stm)
            success = True
            print('Operation is executed successfully!')
        else:
            print(f'No matching records found in session {i}')
    except Exception as e:
        print(f'Error executing on remote session: {e}')
        continue

if not success:
    print("No matching data found in any session")
    return False

try:
    remote_cluster, remote_session = connect_to_cluster(cluster_ip, keyspace_name)
    my_cluster, my_session = connect_to_cluster('127.0.0.1', keyspace_name)

    cluster_sessions = [
        my_session,
        remote_session

```

```
]
```

```
print("\n=====Kiểm tra kết nối 2 máy thuộc 2 cluster====")
```

```
# insert
```

```
insert_query = f"""
```

```
INSERT INTO doanh_thu_moi_ngay_theo_ma_cn (
```

```
    ma_chi_nhanh,
```

```
    ngay,
```

```
    tong_tien
```

```
) VALUES (
```

```
    2,
```

```
    '2021-01-03',
```

```
    123000456
```

```
)
```

```
"""
```

```
insert_rs = crud_ops(
```

```
    query=insert_query,
```

```
    cluster_sessions=[remote_session],
```

```
    table_name='doanh_thu_moi_ngay_theo_ma_cn',
```

```
    condition=None,
```

```
    insert=True
```

```
)
```

```
if not insert_rs:
```

```
    print("Executed failed!")
```

```
except Exception as e:
```

```
    print(f"Error when doing queries: {e}")
```

```

finally:
    if 'remote_cluster' in locals():
        remote_cluster.shutdown()
    if 'my_cluster' in locals():
        my_cluster.shutdown()

```

```

True
Connected to cluster_2 having IP address: 127.0.0.1 successfully!

Attempting to connect to BTL2_data keyspace...
Connected to btl2_data successfully!

=====Kiểm tra kết nối 2 máy thuộc 2 cluster=====
Inserted successfully!

D:\Nam3_HK2\CoSoDuLieuPhanTan\DoAn\BTL2\migrate_oracle_cassandra\cassandra>

```

Hình 5.8 Terminal ở CN1 chạy file code Python thực hiện insert

- + Máy 2 (CN2) kiểm tra xem Máy 1 có thực hiện thêm doanh thu ngày 3/1/2021 vào CN2 thành công hay chưa.

```

cqlsh:btl2_data> select * from doanh_thu_moi_ngay_theo_ma_cn where ma_chi_nhanh= 2 and ngay='2021-01-03';

ma_chi_nhanh | ngay      | tong_tien
-----+-----+-----
2 | 2021-01-03 | 123000456

(1 rows)

```

Hình 5.9 Terminal ở CN2 kiểm tra kết quả insert của CN1

- + Để kiểm tra tính đúng đắn của thao tác thêm là chỉ thêm record vào CN2 thì CN1 kiểm tra lại thông tin.

```
cqlsh:btl2_data> select * from doanh_thu_moi_ngay_theo_ma_cn where ma_chi_nhanh= 1 and ngay='2021-01-03';

ma_chi_nhanh | ngay | tong_tien
-----+-----+
(0 rows)

cqlsh:btl2_data> select * from doanh_thu_moi_ngay_theo_ma_cn where ma_chi_nhanh= 2 and ngay='2021-01-03';

ma_chi_nhanh | ngay | tong_tien
-----+-----+
(0 rows)
```

Hình 5.10 Terminal CN1 kiểm tra tính đúng đắn của thao tác insert

2. Cập nhật dữ liệu (Trong code có áp dụng tính trong suốt phân tán)

- *Trường hợp demo update với dòng dữ liệu chỉ thuộc Máy 1 (Chi nhánh 1):*
 - + Code Python.

```
import os
import sys
from connect_2_clusters import connect_to_cluster
from cassandra.query import SimpleStatement
from tabulate import tabulate
from cassandra.cluster import Cluster

sys.path.append('/opt/airflow/scripts')
sys.path.append(os.path.join(os.path.dirname(__file__), '..', 'scripts'))

from load_to_cassandra import get_cassandra_session

cluster_ip = '26.103.246.194'
keyspace_name = 'btl2_data'

def crud_ops(query, cluster_sessions, table_name, condition, insert):
    stm = SimpleStatement(query)
```

```

check_query = f"""
    SELECT COUNT(*) FROM {table_name} WHERE {condition};
"""

if insert:
    cluster_sessions[0].execute(stm)
    print("Inserted successfully!")
    return True

success = False
for i in range(len(cluster_sessions)):
    try:
        tmp_check = cluster_sessions[i].execute(check_query)
        if tmp_check.one()[0] > 0:
            cluster_sessions[i].execute(stm)
            success = True
            print('Operation is executed successfully!')
        else:
            print(f"No matching records found in session {i}")
    except Exception as e:
        print(f"Error executing on remote session: {e}")
        continue

if not success:
    print("No matching data found in any session")
    return False

try:

```

```

remote_cluster, remote_session = connect_to_cluster(cluster_ip, keyspace_name)
my_cluster, my_session = connect_to_cluster('127.0.0.1', keyspace_name)

cluster_sessions = [
    my_session,
    remote_session
]

print("\n=====Kiểm tra kết nối 2 máy thuộc 2 cluster=====")

# update
update_query = f"""
    UPDATE {keyspace_name}.chi_tiet_hoa_don_theo_ma_kh
    SET so_luong = 15
    WHERE ma_khachhang = 4317 AND ngay_tao = '2023-07-17
13:01:23.000000+0000'
        AND ma_hoa_don = 58495
;"""

update_rs = crud_ops(
    update_query,
    cluster_sessions,
    'chi_tiet_hoa_don_theo_ma_kh',
    "ma_khachhang = 4317 AND ngay_tao = '2023-07-17
13:01:23.000000+0000' AND ma_hoa_don = 58495",
    insert=False
)
if not update_rs:
    print("Executed failed!")

```

```
except Exception as e:
```

```
    print(f"Error when doing queries: {e}")
```

```
finally:
```

```
    if 'remote_cluster' in locals():
        remote_cluster.shutdown()
    if 'my_cluster' in locals():
        my_cluster.shutdown()
```

- + Demo đối với dòng dữ liệu chi tiết hóa đơn có hóa đơn có mã là 58495, ngày tạo là “2023-07-17 13:01:23.000000+0000” của khách hàng có mã là 4317. Dòng dữ liệu này chỉ có trong Chi nhánh 1 (CN1).

```
cqlsh:btl2_data> select * from chi_tiet_hoa_don_theo_ma_kh where ma_khach_hang=4317 and ngay_tao='2023-07-17 13:01:23.000000+0000' and ma_hoa_don=58495;
ma_khach_hang | ngay_tao | ma_hoa_don | ma_nhan_vien | ma_san_pham | phuong_thuc_thanh_toan | so_luong | thanh_tien | tong_tien
-----+-----+-----+-----+-----+-----+-----+-----+-----+
4317 | 2023-07-17 13:01:23.000000+0000 | 58495 | 285 | CCNPOT0196 | Tiền Mát | 4 | 5000000 | 16950000
(1 rows)
```

Hình 5.11 Terminal CN1 thực hiện kiểm tra sự tồn tại của dòng dữ liệu

```
cqlsh:btl2_data> select * from chi_tiet_hoa_don_theo_ma_kh where ma_khach_hang=4317 and ngay_tao='2023-07-17 13:01:23.000000+0000' and ma_hoa_don=58495;
ma_khach_hang | ngay_tao | ma_hoa_don | ma_nhan_vien | ma_san_pham | phuong_thuc_thanh_toan | so_luong | thanh_tien | tong_tien
-----+-----+-----+-----+-----+-----+-----+-----+-----+
(0 rows)
cqlsh:btl2_data> |
```

Hình 5.12 Terminal CN2 thực hiện kiểm tra sự tồn tại của dòng dữ liệu

- + Tính trong suốt phân tán thể hiện trong trường hợp này như sau: Khi Máy 2 thực hiện cập nhật số lượng của chi tiết hóa đơn đó thì logic xử lý sẽ là kiểm tra xem chi tiết hóa đơn đó được tạo bởi nhân viên nào, từ đó sẽ xác định được hóa đơn thuộc chi nhánh nào (vì mỗi nhân viên chỉ thuộc một chi nhánh) và thực hiện thao tác cập nhật tương ứng trên chi nhánh đó.

```

pwsh python3 .\crud.py
True
Connected to cluster_1 having IP address: 26.103.246.194 successfully!

Attempting to connect to BTL2_data keyspace...
Connected to BTL2_DATA successfully!
True
Connected to cluster_1 having IP address: 127.0.0.1 successfully!

Attempting to connect to BTL2_data keyspace...
Connected to BTL2_DATA successfully!

=====Kiểm tra kết nối 2 máy thuộc 2 cluster=====
No matching records found in session 0
Operation is executed successfully!

```

Hình 5.13 Terminal CN2 chạy file code Python thực hiện update

- + Máy 1 (Chi nhánh 1) kiểm tra xem Máy 2 có thực hiện cập nhật số lượng chi tiết hóa đơn ở CN1 thành công hay chưa.

```

cqlsh:btl2_data> select * from chi_tiet_hoa_don_theo_ma_kh where ma_khach_hang=4317 and ngay_tao='2023-07-17 13:01:23.000000+0000' and ma_hoa_don=58495;

ma_khach_hang | ngay_tao | ma_hoa_don | ma_nhan_vien | ma_san_pham | phuong_thuc_thanh_toan | so_luong | thanh_tien | tong_tien
-----+-----+-----+-----+-----+-----+-----+-----+-----+
4317 | 2023-07-17 13:01:23.000000+0000 | 58495 | 285 | CNPOT0196 | Tiền Mật | 15 | 5000000 | 16950000
(1 rows)

```

Hình 5.14 Terminal ở CN1 kiểm tra kết quả cập nhật của CN2

- + Để kiểm tra tính đúng đắn của thao tác cập nhật là chỉ cập nhật record ở CN1 thì CN2 kiểm tra lại thông tin.

```

cqlsh:btl2_data> select * from chi_tiet_hoa_don_theo_ma_kh where ma_khach_hang=4317 and ngay_tao='2023-07-17 13:01:23.000000+0000' and ma_hoa_don=58495;

ma_khach_hang | ngay_tao | ma_hoa_don | ma_nhan_vien | ma_san_pham | phuong_thuc_thanh_toan | so_luong | thanh_tien | tong_tien
-----+-----+-----+-----+-----+-----+-----+-----+-----+
(0 rows)

```

Hình 5.15 Terminal CN2 kiểm tra tính đúng đắn của thao tác update

- **Trường hợp demo update với dòng dữ liệu chỉ thuộc Máy 2 (Chi nhánh 2):**
 - + Code Python.

```

import os
import sys
from connect_2_clusters import connect_to_cluster

```

```

from cassandra.query import SimpleStatement
from tabulate import tabulate
from cassandra.cluster import Cluster

sys.path.append('/opt/airflow/scripts')
sys.path.append(os.path.join(os.path.dirname(__file__), '..', 'scripts'))

from load_to_cassandra import get_cassandra_session

cluster_ip = '26.93.36.133'
keyspace_name = 'btl2_data'

def crud_ops(query, cluster_sessions, table_name, condition, insert):
    stm = SimpleStatement(query)
    check_query = f"""
        SELECT COUNT(*) FROM {table_name} WHERE {condition};
    """

    if insert:
        cluster_sessions[0].execute(stm)
        print("Inserted successfully!")
        return True

    success = False
    for i in range(len(cluster_sessions)):
        try:
            tmp_check = cluster_sessions[i].execute(check_query)
            if tmp_check.one()[0] > 0:
                cluster_sessions[i].execute(stm)
                success = True
        except Exception as e:
            print(f"Error executing query on session {i}: {e}")
    return success

```

```

        success = True
        print('Operation is executed successfully!')
    else:
        print(f'No matching records found in session {i}')

except Exception as e:
    print(f'Error executing on remote session: {e}')
    continue

if not success:
    print("No matching data found in any session")
    return False

return True

try:
    remote_cluster, remote_session = connect_to_cluster(cluster_ip, keyspace_name)
    my_cluster, my_session = connect_to_cluster('127.0.0.1', keyspace_name)

    cluster_sessions = [
        my_session,
        remote_session
    ]

    print("\n=====Kiểm tra kết nối 2 máy thuộc 2 cluster=====")

    # update
    update_query = f"""
        UPDATE {keyspace_name}.chi_tiet_hoa_don_theo_ma_kh
        SET so_luong = 8
    """

```

```

WHERE ma_khach_hang = 4317 AND ngay_tao = '2025-02-18
13:16:05.000000+0000'
    AND ma_hoa_don = 58486
    ;
"""

update_rs = crud_ops(
    update_query,
    cluster_sessions,
    'chi_tiet_hoa_don_theo_ma_kh',
    "ma_khach_hang = 4317 AND ngay_tao = '2025-02-18
13:16:05.000000+0000' AND ma_hoa_don = 58486",
    insert=False
)
if not update_rs:
    print("Executed failed!")

except Exception as e:
    print(f"Error when doing queries: {e}")

finally:
    if 'remote_cluster' in locals():
        remote_cluster.shutdown()
    if 'my_cluster' in locals():
        my_cluster.shutdown()

```

- + Demo đối với dòng dữ liệu chi tiết hóa đơn có hóa đơn có mã là 58486, ngày tạo là “2025-02-18 13:16:05.000000+0000” của khách hàng có mã là 4317. Dòng dữ liệu này chỉ có trong Chi nhánh 2 (CN2).

cqlsh:btl2_data> select * from chi_tiet_hoa_don_theo_ma_kh where ma_khachhang=4317 and ngaytao='2025-02-18 13:16:05.000000+0000' and ma_hoadon=58486;								
ma_khachhang	ngaytao	ma_hoadon	ma_nhanvien	ma_sanpham	phuong_thuc_thanh_toan	soluong	thanh_tien	tong_tien
4317	2025-02-18 13:16:05.000000+0000	58486	803	CCNPLT0414	Tiền Mát	6	840000	840000

Hình 5.16 Terminal CN2 thực hiện kiểm tra sự tồn tại của dòng dữ liệu

cqlsh:btl2_data> select * from chi_tiet_hoa_don_theo_ma_kh where ma_khachhang=4317 and ngaytao='2025-02-18 13:16:05.000000+0000' and ma_hoadon=58486;								
ma_khachhang	ngaytao	ma_hoadon	ma_nhanvien	ma_sanpham	phuong_thuc_thanh_toan	soluong	thanh_tien	tong_tien
(0 rows)								

Hình 5.17 Terminal CN1 thực hiện kiểm tra sự tồn tại của dòng dữ liệu

- + Tính trong suốt phân tán thể hiện trong trường hợp này như sau: Khi Máy 1 thực hiện cập nhật số lượng của chi tiết hóa đơn đó thì logic xử lý sẽ là kiểm tra xem chi tiết hóa đơn đó được tạo bởi nhân viên nào, từ đó sẽ xác định được hóa đơn thuộc chi nhánh nào (vì mỗi nhân viên chỉ thuộc một chi nhánh) và thực hiện thao tác cập nhật tương ứng trên chi nhánh đó.

```
True
Connected to cluster_2 having IP address: 127.0.0.1 successfully!

Attempting to connect to BTL2_data keyspace...
Connected to btl2_data successfully!

=====Kiểm tra kết nối 2 máy thuộc 2 cluster=====
No matching records found in session 0
Operation is executed successfully!
```

Hình 5.18 Terminal CN1 chạy file code Python thực hiện update

- + Máy 2 (Chi nhánh 2) kiểm tra xem Máy 1 có thực hiện cập nhật số lượng chi tiết hóa đơn ở CN2 thành công hay chưa.

cqlsh:btl2_data> select * from chi_tiet_hoa_don_theo_ma_kh where ma_khachhang=4317 and ngaytao='2025-02-18 13:16:05.000000+0000' and ma_hoadon=58486;								
ma_khachhang	ngaytao	ma_hoadon	ma_nhanvien	ma_sanpham	phuong_thuc_thanh_toan	soluong	thanh_tien	tong_tien
4317	2025-02-18 13:16:05.000000+0000	58486	803	CCNPLT0414	Tiền Mát	8	840000	840000

Hình 5.19 Terminal ở CN2 kiểm tra kết quả update của CN1

- + Để kiểm tra tính đúng đắn của thao tác cập nhật là chỉ cập nhật record ở CN2 thì CN1 kiểm tra thông tin.

```
cqlsh:btl2_data> select * from chi_tiet_hoa_don_theo_ma_kh where ma_khachhang=4317 and ngay_tao='2025-02-18 13:16:05.000000+0000' and ma_hoa_don=58486;
ma_khachhang | ngay_tao | ma_hoa_don | ma_nhan_vien | ma_san_pham | phuong_thuc_thanh_toan | so_luong | thanh_tien | tong_tien
-----+-----+-----+-----+-----+-----+-----+-----+-----+
(0 rows)
```

Hình 5.20 Terminal CN1 kiểm tra tính đúng đắn của thao tác update

3. Xoá dữ liệu (Trong code có áp dụng tính trong suốt phân tán)

- *Trường hợp demo xoá với dòng dữ liệu chỉ thuộc Máy 1 (Chi nhánh 1):*

- + Code Python.

```
import os
import sys
from connect_2_clusters import connect_to_cluster
from cassandra.query import SimpleStatement
from tabulate import tabulate
from cassandra.cluster import Cluster

sys.path.append('/opt/airflow/scripts')
sys.path.append(os.path.join(os.path.dirname(__file__), '..', 'scripts'))

from load_to_cassandra import get_cassandra_session

cluster_ip = '26.103.246.194'
keyspace_name = 'btl2_data'

def crud_ops(query, cluster_sessions, table_name, condition, insert):
    stm = SimpleStatement(query)
    check_query = f"""
        SELECT COUNT(*) FROM {table_name} WHERE {condition};
```

```

"""
if insert:
    cluster_sessions[0].execute(stm)
    print("Inserted successfully!")
    return True

success = False
for i in range(len(cluster_sessions)):
    try:
        tmp_check = cluster_sessions[i].execute(check_query)
        if tmp_check.one()[0] > 0:
            cluster_sessions[i].execute(stm)
            success = True
            print(f'Operation is executed successfully inn session {i}!')
        else:
            print(f'No matching records found in session {i}')
    except Exception as e:
        print(f'Error executing on remote session: {e}')
        continue

if not success:
    print("No matching data found in any session")
    return False

return True

try:

```

```

remote_cluster, remote_session = connect_to_cluster(cluster_ip, keyspace_name)
my_cluster, my_session = connect_to_cluster('127.0.0.1', keyspace_name)

cluster_sessions = [
    my_session,
    remote_session
]

print("\n=====Kiểm tra kết nối 2 máy thuộc 2 cluster=====")

# delete
del_query = f"""
DELETE
FROM {keyspace_name}.chi_tiet_hoa_don_theo_ma_kh
WHERE ma_khachhang = 4317 AND ngay_tao = '2023-07-17
13:01:23.000000+0000' AND ma_hoa_don = 58495
""""

del_rs = crud_ops(
    del_query,
    cluster_sessions,
    'chi_tiet_hoa_don_theo_ma_kh',
    "ma_khachhang = 4317 AND ngay_tao = '2023-07-17
13:01:23.000000+0000' AND ma_hoa_don = 58495",
    insert=False
)
if not del_rs:
    print("Executed failed!")

except Exception as e:

```

```
print(f"Error when doing queries: {e}")
```

finally:

```
if 'remote_cluster' in locals():
    remote_cluster.shutdown()
if 'my_cluster' in locals():
    my_cluster.shutdown()
```

- + Demo đối với dòng dữ liệu chi tiết hóa đơn có hóa đơn có mã là 58495, ngày tạo là “2023-07-17 13:01:23.000000+0000” của khách hàng có mã là 4317. Dòng dữ liệu này chỉ có trong Chi nhánh 1 (CN1).

```
cqlsh:btl2_data> select * from chi_tiet_hoa_don_theo_ma_kh where ma_khach_hang=4317 and ngay_tao='2023-07-17 13:01:23.000000+0000' and ma_hoa_don=58495;
ma_khach_hang | ngay_tao | ma_hoa_don | ma_nhan_vien | ma_san_pham | phuong_thuc_thanh_toan | so_luong | thanh_tien | tong_tien
-----+-----+-----+-----+-----+-----+-----+-----+-----+
4317 | 2023-07-17 13:01:23.000000+0000 | 58495 | 285 | CNPOT0196 | Tiền Mật | 4 | 5000000 | 16950000
(1 rows)
```

Hình 5.21 Terminal CN1 thực hiện kiểm tra sự tồn tại của dòng dữ liệu

```
cqlsh:btl2_data> select * from chi_tiet_hoa_don_theo_ma_kh where ma_khach_hang=4317 and ngay_tao='2023-07-17 13:01:23.000000+0000' and ma_hoa_don=58495;
ma_khach_hang | ngay_tao | ma_hoa_don | ma_nhan_vien | ma_san_pham | phuong_thuc_thanh_toan | so_luong | thanh_tien | tong_tien
-----+-----+-----+-----+-----+-----+-----+-----+-----+
(0 rows)
```

Hình 5.22 Terminal CN2 thực hiện kiểm tra sự tồn tại của dòng dữ liệu

- + Tính trong suốt phân tán thể hiện trong trường hợp này như sau: Khi Máy 2 thực hiện xóa chi tiết hóa đơn đó thì logic xử lý sẽ là kiểm tra xem chi tiết hóa đơn đó được tạo bởi nhân viên nào, từ đó sẽ xác định được hóa đơn thuộc chi nhánh nào (vì mỗi nhân viên chỉ thuộc một chi nhánh) và thực hiện thao tác xoá tương ứng trên chi nhánh đó.

```

pwsh python3 .\crud.py
True
Connected to cluster_1 having IP address: 26.103.246.194 successfully!

Attempting to connect to BTL2_data keyspace...
Connected to BTL2_DATA successfully!
True
Connected to cluster_1 having IP address: 127.0.0.1 successfully!

Attempting to connect to BTL2_data keyspace...
Connected to BTL2_DATA successfully!

=====Kiểm tra kết nối 2 máy thuộc 2 cluster=====
No matching records found in session 0
Operation is executed successfully inn session 1!

```

Hình 5.23 Terminal CN2 chạy file Python thực hiện delete

- + Máy 1 (Chi nhánh 1) kiểm tra xem Máy 2 có thực hiện xóa chi tiết hóa đơn ở CN1 thành công hay chưa.

```

cqlsh:btl2_data> select * from chi_tiet_hoa_don_theo_ma_kh where ma_khach_hang=4317 and ngay_tao='2023-07-17 13:01:23.000000+0000' and ma_hoa_don=58495;
ma_khach_hang | ngay_tao | ma_hoa_don | ma_nhan_vien | ma_san_pham | phuong_thuc_thanh_toan | so_luong | thanh_tien | tong_tien
-----+-----+-----+-----+-----+-----+-----+-----+-----+
(0 rows)

```

Hình 5.24 Terminal ở CN1 kiểm tra kết quả delete của CN2

- **Trường hợp demo xoá với dòng dữ liệu chỉ thuộc Máy 2 (Chi nhánh 2):**
 - + Code Python.

```

import os
import sys
from connect_2_clusters import connect_to_cluster
from cassandra.query import SimpleStatement
from tabulate import tabulate
from cassandra.cluster import Cluster

sys.path.append('/opt/airflow/scripts')
sys.path.append(os.path.join(os.path.dirname(__file__), '..', 'scripts'))

```

```

from load_to_cassandra import get_cassandra_session

cluster_ip = '26.93.36.133'
keyspace_name = 'btl2_data'

def crud_ops(query, cluster_sessions, table_name, condition, insert):
    stm = SimpleStatement(query)
    check_query = f"""
        SELECT COUNT(*) FROM {table_name} WHERE {condition};
    """

    if insert:
        cluster_sessions[0].execute(stm)
        print("Inserted successfully!")
        return True

    success = False
    for i in range(len(cluster_sessions)):
        try:
            tmp_check = cluster_sessions[i].execute(check_query)
            if tmp_check.one()[0] > 0:
                cluster_sessions[i].execute(stm)
                success = True
                print(f'Operation is executed successfully inn session {i}!')
            else:
                print(f'No matching records found in session {i}')
        except Exception as e:
            print(f'Error executing on remote session: {e}')

```

```

        continue

    if not success:
        print("No matching data found in any session")
        return False

    return True

try:
    remote_cluster, remote_session = connect_to_cluster(cluster_ip, keyspace_name)
    my_cluster, my_session = connect_to_cluster('127.0.0.1', keyspace_name)

    cluster_sessions = [
        my_session,
        remote_session
    ]

    print("\n=====Kiểm tra kết nối 2 máy thuộc 2 cluster=====")

    # delete
    del_query = f"""
    DELETE
    FROM {keyspace_name}.chi_tiet_hoa_don_theo_ma_kh
    WHERE ma_khachhang = 4317 AND ngay_tao = '2025-02-18
13:16:05.000000+0000' AND ma_hoa_don = 58486
"""

    del_rs = crud_ops(
        del_query,
        cluster_sessions,

```

```

        'chi_tiet_hoa_don_theo_ma_kh',
        "ma_khachhang = 4317 AND ngay_tao = '2025-02-18
13:16:05.000000+0000' AND ma_hoa_don = 58486",
        insert=False
    )
    if not del_rs:
        print("Executed failed!")

except Exception as e:
    print(f"Error when doing queries: {e}")

finally:
    if 'remote_cluster' in locals():
        remote_cluster.shutdown()
    if 'my_cluster' in locals():
        my_cluster.shutdown()

```

- + Demo đối với dòng dữ liệu chi tiết hóa đơn có hóa đơn có mã là 58486, ngày tạo là “2025-02-18 13:16:05.000000+0000” của khách hàng có mã là 4317. Dòng dữ liệu này chỉ có trong Chi nhánh 2 (CN2).

```
cqlsh:btl2_data> select * from chi_tiet_hoa_don_theo_ma_kh where ma_khachhang=4317 and ngay_tao='2025-02-18 13:16:05.000000+0000' and ma_hoa_don=58486;
ma_khachhang | ngay_tao | ma_hoa_don | ma_nhan_vien | ma_san_pham | phuong_thuc_thanh_toan | so_luong | thanh_tien | tong_tien
4317 | 2025-02-18 13:16:05.000000+0000 | 58486 | 803 | CCNPLT0414 | Tiền Mật | 8 | 840000 | 840000
(1 rows)
```

Hình 5.25 Terminal CN2 thực hiện kiểm tra sự tồn tại của dòng dữ liệu

```
cqlsh:btl2_data> select * from chi_tiet_hoa_don_theo_ma_kh where ma_khachhang=4317 and ngay_tao='2025-02-18 13:16:05.000000+0000' and ma_hoa_don=58486;
ma_khachhang | ngay_tao | ma_hoa_don | ma_nhan_vien | ma_san_pham | phuong_thuc_thanh_toan | so_luong | thanh_tien | tong_tien
-----+-----+-----+-----+-----+-----+-----+-----+
(0 rows)
```

Hình 5.26 Terminal CN1 thực hiện kiểm tra sự tồn tại của dòng dữ liệu

- + Tính trong suốt phân tán thể hiện trong trường hợp này như sau: Khi Máy 1 thực hiện xóa chi tiết hóa đơn đó thì logic xử lý sẽ là kiểm tra xem chi tiết hóa đơn đó được tạo bởi nhân viên nào, từ đó sẽ xác định được hóa đơn thuộc chi nhánh nào (vì mỗi nhân viên chỉ thuộc một chi nhánh) và thực hiện thao tác xoá tương ứng trên chi nhánh đó.

```
True
Connected to cluster_2 having IP address: 127.0.0.1 successfully!

Attempting to connect to BTL2_data keyspace...
Connected to btl2_data successfully!

=====Kiểm tra kết nối 2 máy thuộc 2 cluster=====
No matching records found in session 0
Operation is executed successfully inn session 1!

D:\Nam3_HK2\CoSoDuLieuPhanTan\DoAn\BTL2\migrate_oracle_cassandra>
```

Hình 5.27 Terminal CN1 chạy file Python thực hiện delete

- + Máy 2 (Chi nhánh 2) kiểm tra xem Máy 1 có thực hiện xóa chi tiết hóa đơn ở CN2 thành công hay chưa.

```
cqlsh:btl2_data> select * from chi_tiet_hoa_don_theo_ma_kh where ma_khach_hang=4317 and ngay_tao='2025-02-18 13:16:05.000000+0000' and ma_hoa_don=58486;
ma_khach_hang | ngay_tao | ma_hoa_don | ma_nhan_vien | ma_san_pham | phuong_thuc_thanh_toan | so_luong | thanh_tien | tong_tien
-----+-----+-----+-----+-----+-----+-----+-----+-----+
(0 rows)
```

Hình 5.28 Terminal ở CN2 kiểm tra kết quả delete của CN1

4. Thực hiện một số câu truy vấn diễn hình giữa hai máy trong Cassandra

- Trong Cassandra, 5 loại truy vấn diễn hình (phổ biến và thực tế hay gặp nhất), dựa trên đặc điểm của kiến trúc phân tán theo partition key.
- Partition Key:
 - + Là phần đầu tiên của khóa chính (PRIMARY KEY)
 - + Dùng để xác định dữ liệu sẽ được lưu vào node nào trong cụm Cassandra.
 - + Tất cả các bản ghi có cùng Partition Key sẽ nằm trong cùng một partition vật lý → tối ưu cho truy vấn nhanh.
- Clustering Column:

- + Là phần còn lại trong khóa chính sau Partition Key.
- + Dùng để xếp thứ tự dữ liệu bên trong một partition (dữ liệu được lưu có thứ tự theo Clustering Column).
- + Giúp truy vấn theo phạm vi (range) hoặc lọc theo điều kiện cụ thể.

❖ Truy vấn theo partition key (tối ưu nhất)

- Cú pháp: WHERE partition_key = ...
- Cassandra lưu dữ liệu theo partition key, nên truy vấn này là nhanh nhất và truy cập trực tiếp vào phân vùng.
- Kết quả chạy ở Máy 1 (Chi nhánh 1).

```
cqlsh:btl2_data> SELECT * FROM chi_tiet_hoa_don_theo_ma_kh WHERE ma_khach_hang = 1584;

ma_khach_hang | ngay_tao          | ma_hoa_don | ma_nhan_vien | ma_san_pham | phuong_thuc_thanh_toan | so_luong | thanh_tien | tong_tien
-----+-----+-----+-----+-----+-----+-----+-----+-----+
1584 | 2024-09-04 02:54:18.000000+0000 | 21710 | 257 | CNPPLT0359 | Ngân Hàng | 4 | 720000 | 1080000
1584 | 2024-02-13 01:13:01.000000+0000 | 21716 | 254 | CNPOT0138 | Tiền Mát | 7 | 840000 | 840000
1584 | 2022-06-12 04:28:32.000000+0000 | 21712 | 283 | CNPPLT0599 | Ngân Hàng | 4 | 320000 | 5830000
(3 rows)
```

Hình 5.29 Thực thi câu truy vấn 1 (CN1)

- Kết quả chạy ở Máy 2 (Chi nhánh 2).

```
cqlsh:btl2_data> SELECT * FROM chi_tiet_hoa_don_theo_ma_kh WHERE ma_khach_hang = 1584;

ma_khach_hang | ngay_tao          | ma_hoa_don | ma_nhan_vien | ma_san_pham | phuong_thuc_thanh_toan | so_luong | thanh_tien | tong_tien
-----+-----+-----+-----+-----+-----+-----+-----+-----+
1584 | 2025-05-03 00:32:34.000000+0000 | 21717 | 875 | CNPPLT0116 | Tiền Mát | 1 | 420000 | 4060000
1584 | 2023-08-21 00:28:03.000000+0000 | 21713 | 789 | CNPPLT0644 | Tiền Mát | 3 | 285000 | 3380000
1584 | 2022-08-08 14:52:18.000000+0000 | 21718 | 646 | CNPPLT0169 | Tiền Mát | 6 | 2160000 | 2160000
1584 | 2022-07-27 15:35:00.000000+0000 | 21715 | 810 | CNPPLT0493 | Ngân Hàng | 4 | 200000 | 200000
(4 rows)
```

Hình 5.30 Thực thi câu truy vấn 1 (CN2)

- Code Python thực hiện UNION kết quả ở Chi nhánh 1 và Chi nhánh 2.

```
import os
import sys
from connect_2_clusters import connect_to_cluster
from cassandra.query import SimpleStatement
from tabulate import tabulate
from cassandra.cluster import Cluster
```

```

sys.path.append('/opt/airflow/scripts')
sys.path.append(os.path.join(os.path.dirname(__file__), '..', 'scripts'))

from load_to_cassandra import get_cassandra_session

cluster_ip = '26.103.246.194'
keyspace_name = 'btl2_data'

def select_ops(query, cluster_sessions, table_name, condition):
    stm = SimpleStatement(query)
    success = False
    check_query = f"""
        SELECT COUNT(*) FROM {table_name} WHERE {condition};
    """
    headers = None
    all_rows = []
    for i in range(len(cluster_sessions)):
        try:
            tmp_check = cluster_sessions[i].execute(check_query)
            if tmp_check.one()[0] > 0:
                rows = cluster_sessions[i].execute(stm)
                if headers is None:
                    headers = list(rows.column_names)
                for row in rows:
                    row_data = list(row)
                    all_rows.append(row_data)
        except Exception as e:
            print(f"Error executing query for session {i}: {e}")
    return all_rows

```

```

        success = True
        print(f'Operation is executed successfully in session {i}!')
    else:
        print(f'No matching records found in session {i}')

except Exception as e:
    print(f'Error executing on remote session: {e}')
    continue

if not success:
    print("No matching data found in any session")
    return False

if all_rows and headers:
    print(tabulate(all_rows, headers=headers, tablefmt='fancy_grid'))

return True

try:
    remote_cluster, remote_session = connect_to_cluster(cluster_ip, keyspace_name)
    my_cluster, my_session = connect_to_cluster('127.0.0.1', keyspace_name)

    cluster_sessions = [
        my_session,
        remote_session
    ]

    print("\n=====Kiểm tra kết nối 2 máy thuộc 2 cluster=====")

```

```

# select: partition key
sel_part_query = f"""
SELECT * FROM chi_tiet_hoa_don_theo_ma_kh
WHERE ma_khach_hang = 1584;
"""

sel_part_rs = select_ops(
    sel_part_query,
    cluster_sessions,
    'chi_tiet_hoa_don_theo_ma_kh',
    "ma_khach_hang = 1584"
)
if not sel_part_rs:
    print("Executed failed!")

except Exception as e:
    print(f"Error when doing queries: {e}")

finally:
    if 'remote_cluster' in locals():
        remote_cluster.shutdown()
    if 'my_cluster' in locals():
        my_cluster.shutdown()

```

```

pwsh python3 .\crud.py
True
Connected to cluster_1 having IP address: 26.103.246.194 successfully!

Attempting to connect to BTL2_DATA keyspace...
Connected to BTL2_DATA successfully!
True
Connected to cluster_1 having IP address: 127.0.0.1 successfully!

Attempting to connect to BTL2_DATA keyspace...
Connected to BTL2_DATA successfully!

=====Kiểm tra kết nối 2 máy thuộc 2 cluster=====
Operation is executed successfully in session 0!
Operation is executed successfully in session 1!



| ma_khach_hang | ngay_tao            | ma_hoa_don | ma_nhan_vien | ma_san_pham | phuong_thuc_thanh_toan | so_luong | thanh_tien | tong_tien |
|---------------|---------------------|------------|--------------|-------------|------------------------|----------|------------|-----------|
| 1584          | 2025-05-03 00:32:34 | 21717      | 875          | CCNPLT0116  | Tiền Mặt               | 1        | 420000     | 4060000   |
| 1584          | 2023-08-21 00:28:03 | 21713      | 789          | CCNPLT0644  | Tiền Mặt               | 3        | 285000     | 3380000   |
| 1584          | 2022-08-08 14:52:18 | 21718      | 646          | CCNPLT0169  | Tiền Mặt               | 6        | 2160000    | 2160000   |
| 1584          | 2022-07-27 15:35:00 | 21715      | 810          | CCNPLT0493  | Ngân Hàng              | 4        | 200000     | 200000    |
| 1584          | 2024-09-04 02:54:18 | 21710      | 257          | CCNPLT0359  | Ngân Hàng              | 4        | 720000     | 1080000   |
| 1584          | 2024-02-13 01:13:01 | 21716      | 254          | CCNPOT0138  | Tiền Mặt               | 7        | 840000     | 840000    |
| 1584          | 2022-06-12 04:28:32 | 21712      | 283          | CCNPLT0599  | Ngân Hàng              | 4        | 320000     | 5830000   |


```

Hình 5.31 Union kết quả câu truy vấn 1 của CNI và CN2

❖ Truy vấn theo partition key + clustering key

- Cho phép truy xuất một hàng cụ thể hoặc nhiều hàng trong một phân vùng, dùng clustering key để lọc hoặc sắp xếp.
- Kết quả chạy ở Máy 1 (Chi nhánh 1).

```

cqlsh:btl2_data> SELECT * FROM sl_khach_hang_moi_ngay_theo_ma_cn WHERE ma_chi_nhanh = 1 AND ngay = '2024-05-01';

ma_chi_nhanh | ngay      | so_luong_khach_hang
-----+-----+-----
1 | 2024-05-01 |          314

(1 rows)

```

Hình 5.32 Thực thi câu truy vấn 2 (CNI)

- Kết quả chạy ở Máy 2 (Chi nhánh 2).

```

cqlsh:btl2_data> SELECT * FROM sl_khach_hang_moi_ngay_theo_ma_cn WHERE ma_chi_nhanh = 2 AND ngay = '2024-05-01';

ma_chi_nhanh | ngay      | so_luong_khach_hang
-----+-----+-----
2 | 2024-05-01 |          299

(1 rows)

```

Hình 5.33 Thực thi câu truy vấn 2 (CN2)

- Code Python thực hiện UNION kết quả ở Chi nhánh 1 và Chi nhánh 2.

```

import os
import sys
from connect_2_clusters import connect_to_cluster
from cassandra.query import SimpleStatement
from tabulate import tabulate
from cassandra.cluster import Cluster

sys.path.append('/opt/airflow/scripts')
sys.path.append(os.path.join(os.path.dirname(__file__), '..', 'scripts'))

from load_to_cassandra import get_cassandra_session

cluster_ip = '26.103.246.194'
keyspace_name = 'btl2_data'

def select_ops(query, cluster_sessions, table_name, condition):
    stm = SimpleStatement(query)
    success = False
    if condition:
        check_query = f"""
            SELECT COUNT(*) FROM {table_name} WHERE {condition};
        """
        headers = None
        all_rows = []
        for i in range(len(cluster_sessions)):
            try:
                tmp_check = cluster_sessions[i].execute(check_query)
                if tmp_check.one()[0] > 0:

```

```

        rows = cluster_sessions[i].execute(stm)
        if headers is None:
            headers = list(rows.column_names)

        for row in rows:
            row_data = list(row)
            all_rows.append(row_data)

        success = True
        print(f'Operation is executed successfully in session {i}!')
    else:
        print(f'No matching records found in session {i}!')

    except Exception as e:
        print(f'Error executing on remote session: {e}')
        continue

if not success:
    print("No matching data found in any session")
    return False

if all_rows and headers:
    print(tabulate(all_rows, headers=headers, tablefmt='fancy_grid'))

return True

try:
    remote_cluster, remote_session = connect_to_cluster(cluster_ip, keyspace_name)
    my_cluster, my_session = connect_to_cluster('127.0.0.1', keyspace_name)

```

```

cluster_sessions = [
    my_session,
    remote_session
]

print("\n=====Kiểm tra kết nối 2 máy thuộc 2 cluster=====")

sel_part_clus_query = f"""
SELECT * FROM chi_tiet_hoa_don_theo_ma_cn
WHERE ma_chi_nhanh IN (1, 2) AND ngay = '2024-05-01';
"""

sel_part_clus_rs = select_ops(
    sel_part_clus_query,
    cluster_sessions,
    'sl_khach_hang_moi_ngay_theo_ma_cn',
    "ma_chi_nhanh IN (1, 2) AND ngay = '2024-05-01"
)
if not sel_part_clus_rs:
    print("Executed failed!")

except Exception as e:
    print(f"Error when doing queries: {e}")

finally:
    if 'remote_cluster' in locals():
        remote_cluster.shutdown()
    if 'my_cluster' in locals():
        my_cluster.shutdown()

```

```

pwshpython3 .\crud.py
True
Connected to cluster_1 having IP address: 26.103.246.194 successfully!

Attempting to connect to BTL2_DATA keyspace...
Connected to BTL2_DATA successfully!
True
Connected to cluster_1 having IP address: 127.0.0.1 successfully!

Attempting to connect to BTL2_DATA keyspace...
Connected to BTL2_DATA successfully!

=====Kiểm tra kết nối 2 máy thuộc 2 cluster=====
Operation is executed successfully in session 0!
Operation is executed successfully in session 1!



| ma_chi_nhanh | ngay       | so_luong_khach_hang |
|--------------|------------|---------------------|
| 2            | 2024-05-01 | 299                 |
| 1            | 2024-05-01 | 314                 |


```

Hình 5.34 Union kết quả câu truy vấn 2 của CN1 và CN2

❖ Truy vấn theo partition key với range trên clustering key

- Dùng để lấy một tập con các bản ghi trong cùng một partition, ví dụ dùng >, <, =, >=, <= trên clustering key.
- Kết quả chạy ở Máy 1 (Chi nhánh 1).

```

cqlsh:btl2_data> SELECT * FROM kho_sp_theo_ma_cn WHERE ma_chi_nhanh = 1 AND ma_san_pham = 'CCNPLT0021' AND tong_so_luong_ton_kho >= 23 AND tong_so_luong_ton_kho <= 64;

ma_chi_nhanh | ma_san_pham | tong_so_luong_ton_kho | ten_san_pham
-----+-----+-----+-----+
1 | CCNPLT0021 | 50 | Băng Singapore / Sung tỳ bà (thân thấp) | Côn hàng | 297 | 99
-----+-----+-----+-----+
(1 rows)

```

Hình 5.35 Thực thi câu truy vấn 3 (CNI)

- Kết quả chạy ở Máy 2 (Chi nhánh 2).

```

cqlsh:btl2_data> SELECT * FROM kho_sp_theo_ma_cn WHERE ma_chi_nhanh = 2 AND ma_san_pham = 'CCNPLT0021' AND tong_so_luong_ton_kho >= 23 AND tong_so_luong_ton_kho <= 64;

ma_chi_nhanh | ma_san_pham | tong_so_luong_ton_kho | ten_san_pham | tinh_trang | tong_so_luong_da_ban | tong_so_luong_danh_gia
-----+-----+-----+-----+-----+-----+-----+
(0 rows)

```

Hình 5.36 Thực thi câu truy vấn 3 (CN2)

- Code Python thực hiện UNION kết quả ở Chi nhánh 1 và Chi nhánh 2.

```

import os
import sys
from connect_2_clusters import connect_to_cluster
from cassandra.query import SimpleStatement
from tabulate import tabulate
from cassandra.cluster import Cluster

sys.path.append('/opt/airflow/scripts')
sys.path.append(os.path.join(os.path.dirname(__file__), '..', 'scripts'))

from load_to_cassandra import get_cassandra_session

cluster_ip = '26.103.246.194'
keyspace_name = 'btl2_data'

def select_union_ops(queries, cluster_sessions):
    stm = []
    if isinstance(queries, list):
        for query in queries:
            prepared_stm = SimpleStatement(query)
            stm.append(prepared_stm)

    success = False

    headers = None
    all_rows = []
    for i in range(len(cluster_sessions)):
        try:

```

```

rows = cluster_sessions[i].execute(stm[i])
if headers is None:
    headers = list(rows.column_names)

for row in rows:
    row_data = list(row)
    all_rows.append(row_data)

success = True
print(f'Operation is executed successfully in session {i}!')

except Exception as e:
    print(f'Error executing on remote session: {e}')
    continue

if not success:
    print("No matching data found in any session")
    return False

if all_rows and headers:
    print(tabulate(all_rows, headers=headers, tablefmt='fancy_grid'))

return True

try:
    remote_cluster, remote_session = connect_to_cluster(cluster_ip, keyspace_name)
    my_cluster, my_session = connect_to_cluster('127.0.0.1', keyspace_name)

```

```

cluster_sessions = [
    my_session,
    remote_session
]

print("\n=====Kiểm tra kết nối 2 máy thuộc 2 cluster=====")

sel_part_clus_query_1 = f"""
SELECT * FROM kho_sp_theo_ma_cn
WHERE ma_chi_nhanh = 2 AND ma_san_pham = 'CCNPLT0021' AND
tong_so_luong_ton_kho >= 23 AND tong_so_luong_ton_kho <= 64;
"""

sel_part_clus_query_2 = f"""
SELECT * FROM kho_sp_theo_ma_cn
WHERE ma_chi_nhanh = 1 AND ma_san_pham = 'CCNPLT0021' AND
tong_so_luong_ton_kho >= 23 AND tong_so_luong_ton_kho <= 64;
"""

sel_part_clus_union_rs = select_union_ops(
    [sel_part_clus_query_1, sel_part_clus_query_2],
    cluster_sessions
)

if not sel_part_clus_union_rs:
    print("Executed failed!")

except Exception as e:
    print(f"Error when doing queries: {e}")

finally:
    if 'remote_cluster' in locals():

```

```

remote_cluster.shutdown()

if 'my_cluster' in locals():

    my_cluster.shutdown()

```

```

pwsh python3 ..\crud.py
True
Connected to cluster_1 having IP address: 26.103.246.194 successfully!

Attempting to connect to BTL2_DATA keyspace...
Connected to BTL2_DATA successfully!
True
Connected to cluster_1 having IP address: 127.0.0.1 successfully!

Attempting to connect to BTL2_DATA keyspace...
Connected to BTL2_DATA successfully!

=====Kiểm tra kết nối 2 máy thuộc 2 cluster=====
Operation is executed successfully in session 0!
Operation is executed successfully in session 1!

```

ma_chi_nhanh	ma_san_pham	tong_so_luong_ton_kho	ten_san_pham	tinh_trang	tong_so_luong_da_ban	tong_so_luong_danh_gia
1	CNPLT0021	50	Bàng Singapore / Sung tỳ bà (thân thấp)	Còn hàng	297	99

Hình 5.37 Union kết quả câu truy vấn 3 của CN1 và CN2

❖ Truy vấn bằng ALLOW FILTERING (*không khuyến khích*)

- Khi truy vấn không theo khóa chính, Cassandra yêu cầu thêm ALLOW FILTERING, rất chậm và có thể gây lỗi hiệu năng. Chỉ dùng trong trường hợp bắt đắc dĩ, tốt nhất là thiết kế lại bảng để phục vụ truy vấn.
- Kết quả chạy ở Máy 1 (Chi nhánh 1).

```

cqlsh:btl2_data> SELECT * FROM chi_tiet_hoa_don_theo_ma_kh WHERE phuong_thuc_thanh_toan= 'Tiền Mặt' LIMIT 5 ALLOW FILTERING;

ma_khach_hang | ngay_tao | ma_hoa_don | ma_nhan_vien | ma_san_pham | phuong_thuc_thanh_toan | so_luong | thanh_tien | tong_tien
-----+-----+-----+-----+-----+-----+-----+-----+-----+
4317 | 2024-12-17 08:16:22.000000+0000 | 58493 | 149 | CCNPLT0108 | Tiền Mặt | 15 | 1266000 | 442582
62693 | 2023-01-03 03:52:59.000000+0000 | 851138 | 483 | CCNPLT0790 | Tiền Mặt | 10 | 600000 | 11400000
62693 | 2022-02-15 19:44:28.000000+0000 | 851147 | 287 | CCNPLT0632 | Tiền Mặt | 6 | 360000 | 635000
51678 | 2023-10-02 23:41:00.000000+0000 | 701863 | 214 | CCNPLT0212 | Tiền Mặt | 2 | 480000 | 3280000
51678 | 2023-10-02 18:29:07.000000+0000 | 701854 | 89 | CCNPLT0319 | Tiền Mặt | 4 | 700000 | 5940000

```

(5 rows)

Hình 5.38 Thực thi câu truy vấn 4 (CN1)

- Kết quả chạy ở Máy 2 (Chi nhánh 2).

```

cqlsh:btl2_data> SELECT * FROM chi_tiet_hoa_don_theo_ma_kh WHERE phuong_thuc_thanh_toan= 'Tiền Mặt' LIMIT 5 ALLOW FILTERING;

ma_khach_hang | ngay_tao | ma_hoa_don | ma_nhan_vien | ma_san_pham | phuong_thuc_thanh_toan | so_luong | thanh_tien | tong_tien
-----+-----+-----+-----+-----+-----+-----+-----+-----+
4317 | 2023-07-28 16:56:01.000000+0000 | 58488 | 874 | CCNPLT0758 | Tiền Mặt | 4 | 728524 | 728524
4317 | 2023-07-28 13:35:20.000000+0000 | 58498 | 727 | CCNPLT0396 | Tiền Mặt | 1 | 75000 | 15780000
4317 | 2023-07-26 23:51:43.000000+0000 | 58491 | 682 | CCNPLT0746 | Tiền Mặt | 9 | 1356300 | 1356300
4317 | 2022-01-16 10:49:03.000000+0000 | 58485 | 860 | CCNPLT0684 | Tiền Mặt | 6 | 2700000 | 54225000
62693 | 2025-10-12 23:42:21.000000+0000 | 851137 | 646 | CCNPLT0276 | Tiền Mặt | 4 | 880000 | 13480000

```

(5 rows)

Hình 5.39 Thực thi câu truy vấn 4 (CN2)

- Code Python thực hiện UNION kết quả ở Chi nhánh 1 và Chi nhánh 2.

```

import os
import sys
from connect_2_clusters import connect_to_cluster
from cassandra.query import SimpleStatement
from tabulate import tabulate
from cassandra.cluster import Cluster

sys.path.append('/opt/airflow/scripts')
sys.path.append(os.path.join(os.path.dirname(__file__), '..', 'scripts'))

from load_to_cassandra import get_cassandra_session

cluster_ip = '26.103.246.194'
keyspace_name = 'btl2_data'

def select_ops(query, cluster_sessions, table_name, condition):
    stm = SimpleStatement(query)
    success = False

    # Check if the main query uses ALLOW FILTERING
    uses_allow_filtering = 'ALLOW FILTERING' in query.upper()
    if condition:
        check_query = f"""
            SELECT COUNT(*) FROM {table_name} WHERE {condition}
        """
        # Add ALLOW FILTERING to check query if main query uses it
        if uses_allow_filtering:
            check_query += " ALLOW FILTERING"

```

```

check_query += ";"

headers = None
all_rows = []
for i in range(len(cluster_sessions)):
    try:
        tmp_check = cluster_sessions[i].execute(check_query)
        if tmp_check.one()[0] > 0:
            rows = cluster_sessions[i].execute(stm)
            if headers is None:
                headers = list(rows.column_names)

            for row in rows:
                row_data = list(row)
                all_rows.append(row_data)

            success = True
            print(f'Operation is executed successfully in session {i}!')
        else:
            print(f'No matching records found in session {i}')
    except Exception as e:
        print(f'Error executing on remote session: {e}')
        continue

if not success:
    print("No matching data found in any session")
    return False

```

```

if all_rows and headers:
    print(tabulate(all_rows, headers=headers, tablefmt='fancy_grid'))

return True

try:
    remote_cluster, remote_session = connect_to_cluster(cluster_ip, keyspace_name)
    my_cluster, my_session = connect_to_cluster('127.0.0.1', keyspace_name)

    cluster_sessions = [
        my_session,
        remote_session
    ]

print("\n=====Kiểm tra kết nối 2 máy thuộc 2 cluster=====")

sel_allow_filter_query = f"""
SELECT * FROM chi_tiet_hoa_don_theo_ma_kh
WHERE phuong_thuc_thanh_toan= 'Tiền Mặt'
LIMIT 5
ALLOW FILTERING;
"""

sel_allow_filter_rs = select_ops(
    sel_allow_filter_query,
    cluster_sessions,
    'chi_tiet_hoa_don_theo_ma_kh',
    "phuong_thuc_thanh_toan= 'Tiền Mặt"
)

if not sel_allow_filter_rs:

```

```

        print("Executed failed!")

except Exception as e:
    print(f"Error when doing queries: {e}")

finally:
    if 'remote_cluster' in locals():
        remote_cluster.shutdown()
    if 'my_cluster' in locals():
        my_cluster.shutdown()

```

ma_khach_hang	ngay_tao	ma_hoa_don	ma_nhan_vien	ma_san_pham	phuong_thuc_thanh_toan	so_luong	thanh_tien	tong_tien
4317	2023-07-28 16:56:01	58488	874	CCNPLT0758	Tiền Mặt	4	728524	728524
4317	2023-07-28 13:35:20	58498	727	CCNPLT0396	Tiền Mặt	1	75000	15780000
4317	2023-07-26 23:51:43	58491	682	CCNPLT0746	Tiền Mặt	9	1356300	1356300
4317	2022-01-16 10:49:03	58485	860	CCNPLT0684	Tiền Mặt	6	2700000	54225000
62693	2025-10-12 23:42:21	851137	646	CCNPLT0276	Tiền Mặt	4	880000	13480000
4317	2024-12-17 08:16:22	58493	149	CCNPLT0108	Tiền Mặt	15	1260000	4425852
62693	2023-01-03 03:52:59	851138	483	CCNPLT0700	Tiền Mặt	10	600000	11400000
62693	2022-02-15 19:44:28	851147	207	CCNPLT0632	Tiền Mặt	6	360000	635000
51678	2023-10-02 23:41:00	701863	214	CCNPLT0212	Tiền Mặt	2	480000	3280000
51678	2023-10-02 10:29:07	701854	89	CCNPLT0319	Tiền Mặt	4	700000	5940000

Hình 5.40 Union kết quả câu truy vấn 4 của CN1 và CN2

❖ Truy vấn bằng Secondary Index (*hạn chế dùng*)

- Tạo index trên cột không phải khóa chính để phục vụ truy vấn. Không hiệu quả với dữ liệu lớn hoặc phân mảnh cao.
- Dùng tốt cho cột có ít giá trị phân biệt, nhưng vẫn kém hiệu quả hơn so với thiết kế lược đồ phù hợp.

- Kết quả chạy ở Máy 1 (Chi nhánh 1).

```
cqlsh:btl2_data> CREATE INDEX ON doanh_thu_thang_nv_cn (tong_doanh_thu);
cqlsh:btl2_data> SELECT * FROM doanh_thu_thang_nv_cn WHERE tong_doanh_thu > 365000000 ALLOW FILTERING;

ma_chi_nhanh | ma_nhan_vien | nam | thang | tong_doanh_thu
-----+-----+-----+-----+
 1 |      181 | 2024 |     8 | 379403984
 1 |      355 | 2023 |    10 | 365335394
 1 |      308 | 2022 |    12 | 431037311

(3 rows)
```

Engine disk utilization. This includes

Hình 5.41 Thực thi câu truy vấn 5 (CN1)

- Kết quả chạy ở Máy 2 (Chi nhánh 2).

```
cqlsh:btl2_data> CREATE INDEX IF NOT EXISTS ON doanh_thu_thang_nv_cn (tong_doanh_thu);
cqlsh:btl2_data> SELECT * FROM doanh_thu_thang_nv_cn WHERE tong_doanh_thu > 365000000 ALLOW FILTERING;

ma_chi_nhanh | ma_nhan_vien | nam | thang | tong_doanh_thu
-----+-----+-----+-----+
 2 |      685 | 2025 |     4 | 386889754
 2 |      571 | 2025 |     8 | 376913601
 2 |      814 | 2023 |     1 | 367361804
 2 |      676 | 2023 |     1 | 380225819

(4 rows)
```

Hình 5.42 Thực thi câu truy vấn 5 (CN2)

- Code Python thực hiện UNION kết quả ở Chi nhánh 1 và Chi nhánh 2.

```
import os
import sys
from connect_2_clusters import connect_to_cluster
from cassandra.query import SimpleStatement
from tabulate import tabulate
from cassandra.cluster import Cluster

sys.path.append('/opt/airflow/scripts')
sys.path.append(os.path.join(os.path.dirname(__file__), '..', 'scripts'))

from load_to_cassandra import get_cassandra_session
```

```

cluster_ip = '26.103.246.194'
keyspace_name = 'btl2_data'

def select_ops(query, cluster_sessions, table_name, condition):
    stm = SimpleStatement(query)
    success = False

    # Check if the main query uses ALLOW FILTERING
    uses_allow_filtering = 'ALLOW FILTERING' in query.upper()
    if condition:
        check_query = f"""
            SELECT COUNT(*) FROM {table_name} WHERE {condition}
        """
        # Add ALLOW FILTERING to check query if main query uses it
        if uses_allow_filtering:
            check_query += " ALLOW FILTERING"
        check_query += ";"

    headers = None
    all_rows = []
    for i in range(len(cluster_sessions)):
        try:
            tmp_check = cluster_sessions[i].execute(check_query)
            if tmp_check.one()[0] > 0:
                rows = cluster_sessions[i].execute(stm)
                if headers is None:
                    headers = list(rows.column_names)

```

```

        for row in rows:
            row_data = list(row)
            all_rows.append(row_data)

            success = True
            print(f'Operation is executed successfully in session {i}!')

        else:
            print(f'No matching records found in session {i}!')

    except Exception as e:
        print(f'Error executing on remote session: {e}')
        continue

    if not success:
        print("No matching data found in any session")
        return False

    if all_rows and headers:
        print(tabulate(all_rows, headers=headers, tablefmt='fancy_grid'))

    return True

try:
    remote_cluster, remote_session = connect_to_cluster(cluster_ip, keyspace_name)
    my_cluster, my_session = connect_to_cluster('127.0.0.1', keyspace_name)

    cluster_sessions = [
        my_session,
        remote_session

```

```
]
```

```
print("\n=====Kiểm tra kết nối 2 máy thuộc 2 cluster=====")

create_index_query = """
CREATE INDEX IF NOT EXISTS ON doanh_thu_thang_nv_cn
(tong_doanh_thu);
"""

try:
    for session in cluster_sessions:
        session.execute(create_index_query)
        print("Index created successfully!")
except Exception as e:
    print(f"Index creation error (might already exist): {e}")

index_query = """
SELECT * FROM doanh_thu_thang_nv_cn
WHERE tong_doanh_thu > 365000000
ALLOW FILTERING;
"""

sel_index_rs = select_ops(
    index_query,
    cluster_sessions,
    'doanh_thu_thang_nv_cn',
    'tong_doanh_thu > 365000000'
)
if not sel_index_rs:
    print("Executed failed!")
```

```

except Exception as e:
    print(f"Error when doing queries: {e}")

finally:
    if 'remote_cluster' in locals():
        remote_cluster.shutdown()
    if 'my_cluster' in locals():
        my_cluster.shutdown()

```

```

pwsh python3 .\crud.py
True
Connected to cluster_1 having IP address: 26.103.246.194 successfully!

Attempting to connect to BTL2_DATA keyspace...
Connected to BTL2_DATA successfully!
True
Connected to cluster_1 having IP address: 127.0.0.1 successfully!

Attempting to connect to BTL2_DATA keyspace...
Connected to BTL2_DATA successfully!

=====Kiểm tra kết nối 2 máy thuộc 2 cluster=====
Index created successfully!
Operation is executed successfully in session 0!
Operation is executed successfully in session 1!

```

ma_chi_nhanh	ma_nhan_vien	nam	thang	tong_doanh_thu
2	685	2025	4	386889754
2	571	2025	8	376913601
2	814	2023	1	367361804
2	676	2023	1	380225819
1	181	2024	8	379403984
1	355	2023	10	365335394
1	308	2022	12	431037311

Hình 5.43 Union kết quả câu truy vấn 5 của CN1 và CN2

DANH MỤC TÀI LIỆU THAM KHẢO

- [1] Cassandra 4.0 Documentation – CQL Data Manipulation Language (DML). [Online]. Available: <https://cassandra.apache.org/doc/4.0/cassandra/cql/dml.html>
- [2] “Cluster, Data Centers, Racks and Nodes in Cassandra,” Baeldung, [Online]. Available: <https://www.baeldung.com/cassandra-cluster-datacenters-racks-nodes>