ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA HỆ THỐNG THÔNG TIN

BÀI TẬP LỚN 1

THIẾT KẾ CƠ SỞ DỮ LIỆU PHÂN TÁN

MÔN HỌC: CƠ SỞ DỮ LIỆU PHÂN TÁN

LỚP IS211.N11. HTCL – NHÓM 15

GVHD: NGUYỄN MINH NHỰT

THÀNH VIÊN:

Đỗ Đặng Kiến Nam	:	20521627
Phạm Thanh Nhựt	:	20521627
Lê Anh Thư	:	20521985
Lê Phương Tuyết	:	20522135

TP. HỒ CHÍ MINH, 2022

LỜI CẢM ƠN

Lời đầu tiên, nhóm chúng em xin đặc biệt gửi lời cảm ơn đến Thầy Nguyễn Minh Nhựt (giảng viên thực hành môn Cơ sở dữ liệu phân tán), người đã dùng tri thức và tâm huyết của mình để truyền đạt cho chúng em vốn kiến thức vô cùng quý báu trong khoảng thời gian học tập. Những kiến thức mà Thầy truyền đạt là bước đệm quan trọng giúp chúng em có thể hoàn thành đề tài đồ án tốt hơn. Xuất phát từ mục đích học tập, tìm hiểu về kiến thức về cơ sở dữ liệu phân tán trên lớp, nhóm thực hiện thiết kế cơ sở dữ liệu phân tán. Trong quá trình thực hiện đồ án, dựa trên những kiến thức được Thầy chỉ dạy trên lớp cùng với việc học nhóm và tự tìm hiểu những cơng cụ và kiến thức mới, nhóm đã cố gắng thực hiện đồ án một cách tốt nhất. Tuy nhiên, trong quá trình thực hiện, không tránh khỏi những sai sót. Hơn nữa, năng lực và thời gian của cá nhân mỗi thành viên trong nhóm là giới hạn nên sản phẩm vẫn chỉ mang tính chất là một đồ án mơn học, chưa thực sự hoàn thiện. Do đó, rất mong nhận được những sự góp ý từ Thầy nhằm giúp nhóm hoàn thiện những kiến thức đã học tập và cũng là hành trang để nhóm thực hiện tiếp các đề tài khác trong tương lai. Nhóm rất trân trọng và cảm ơn những kiến thức và kỹ năng được truyền đạt đã giúp mỗi thành viên trong nhóm hoàn thiện vốn kiến thức hiện có để nhóm có thể tiếp tục hoàn thành những đồ án và sản phẩm khác cũng như áp dụng vào cơng việc trong tương lai.

Một lần nữa xin gửi đến thầy cô, bạn bè lời cảm ơn chân thành và tốt đẹp nhất!

Thành phố Hồ Chí Minh, tháng 12 năm 2022.

Nhóm sinh viên thực hiện.

MỤC LỤC

CHƯƠNG 1: TỔNG QUAN 1

1. Giới thiệu chung **1**

2. Động lực nghiên cứu **1**

3. Phạm vi báo cáo **1**

4. Thách thức **1**

5. Đóng góp của báo cáo **2**

6. Cấu trúc báo cáo 2

CHƯƠNG 2: THIẾT KẾ CSDL PHÂN TÁN TRÊN MÔI TRƯỜNG MÁY ẢO, RADMIN 3

1. Thiết kế cơ sở dữ liệu 3

2. Thực hiện 10 câu truy vấn 6

2.1. Liệt kê nhân viên (EmpId, EmpName, EmpSalary) ở cả 2 chi nhánh (Hội) 6

2.2. Liệt kê 5 hóa đơn có giá trị cao nhất, sắp xếp giảm dần theo giá trị (PaymentID, Tong, SL) (Gom nhóm) 8

2.3. Cho biết chi nhánh nào bán được đa dạng xe nhất (OficeID, City, SLLoai) (Tính toán) 9

2.4. Tìm khách hàng đã mua xe ở cả hai chi nhánh (CusID, CusName) (Giao) 10

2.5. Tìm khách hàng chỉ mua xe ở chi nhánh Office1 (CusID, CusName) ((Trừ) 11

2.6. Tìm loại xe được mua nhiều nhất (CarID, CarName, SL) (Gom nhóm) 12

2.7. Xem số lượng xe hiện có ở cả hai chi nhánh (CarName, SLOffice1, SLOffice2) 13

2.8. Cho biết tổng số lượng xe đã nhập, đã xuất của từng loại xe trong năm 2020 của chi nhánh Office2 (CarName, SLNhap, SLXuat) 14

2.9. Tìm nhân viên bán được nhiều xe nhất trong năm 2016 của cả hai chi nhánh. 16

2.10. Tìm hóa đơn mua tất cả loại xe hiện có trong công ty có nhà cung cấp là Toyota. (Phép chia) 17

CHƯƠNG 3: VIẾT HÀM, THỦ TỤC, RÀNG BUỘC TOÀN VẸN TRUY VẤN TRÊN MÔI TRƯỜNG PHÂN TÁN 18

1. Trigger 18

2. Procedure 20

3. Function 21

CHƯƠNG 4: DEMO CÁC MỨC CÔ LẬP (ISOLATION LEVEL) TRONG MÔI TRƯỜNG PHÂN TÁN 23

1. LOST UPDATE 23

2. NON-REPEATEABLE 24

3. DEADLOCK 26

4. DIRTY READ 28

5. PHANTOM READ 28

CHƯƠNG 5: THỰC HIỆN TỐI ƯU HÓA TRUY VẤN TRÊN MÔI TRƯỜNG PHÂN TÁN 1 CÂU TRUY VẤN ĐƠN GIẢN 30

1. Lược đồ phân mảnh: 30

2. Tối ưu hóa câu truy vấn: 32

CHƯƠNG 6: CƠ CHẾ NHÂN BẢN TRONG ORACLE 40

1. Nhân bản Oracle là gì? 40

2. .Nhân bản Oracle để làm gì? 40

3. Nhân bản Oracle hoạt động như thế nào? 40

4. Các phương pháp nhân bản cơ sở dữ liệu Oracle 42

4.1. Kết xuất đầy đủ và tải (Full Dump and Load) 42

4.2. Incremental Approach (Table Differencing) 43

4.3. Trigger-Based Approach 43

4.4. Change Data Capture 44

5. DEMO 45

CHƯƠNG 1: TỔNG QUAN

Trong chương này, chúng tôi giới thiệu chung về báo cáo, động lực nghiên cứu, xác định mục tiêu và phạm vi tương ứng, từ đó nêu lên các đóng góp chính của báo cáo. Phần tóm tắt từng chương trong báo cáo được trình bày ở cuối chương này.

1. Giới thiệu chung

Tài liệu này là báo cáo cho bài tập lớn 1 của môn học "Cơ sở dữ liệu phân tán" với đề tài "Thiết kế cơ sở dữ liệu phân tán". Tài liệu bao gồm các nội dung chính:

2. Động lực nghiên cứu

- Đáp ứng yêu cầu của môn học "Cơ sở dữ liệu phân tán" .
- Áp dụng các kiến thức đã được học qua quá trình được giảng dạy trên lớp, thực hiện các yêu cầu, để có thể hiểu rõ rang hơn.

3. Phạm vi báo cáo

Trong phạm vi báo cáo này, chúng tôi tập trung tìm hiểu về CSDL phân tán trên môi trường máy ảo, các hàm, thủ tục, ràng buộc toàn vẹn truy vấn, các mức cô lập (ISOLATION LEVEL), tối ưu hóa truy vấn và cơ chế nhân bản.

4. Thách thức

- Kiến thức mới, nhiều khái niệm chuyên sâu về hệ quản trị cơ sở dữ liệu mà chúng tôi lần đầu tiên tiếp cận.
- Lượng thông tin, tài liệu rất lớn từ cộng đồng phát triển cũng như là các nhà cung cấp hoặc xây dựng hệ quản trị cơ sở dữ liệu. Chúng tôi cần phải chọn lọc thông tin phù hợp.

5. Đóng góp của báo cáo

- Cài đặt trên 2 máy trở lên và thực hiện truy vấn giữa hai máy, các mức cô lập trong môi trường phân tán và đề xuất giải quyết.
- Tìm hiểu về cơ chế nhân bản, cách thức hoạt động trong Oracle.

6. Cấu trúc báo cáo

Báo cáo này này được trình bày trong 6 chương, nội dung chính được tóm tắt như dưới đây:

Chương 1: Tổng quan.

Chương 2: Thiết kế CSDL phân tán trên môi trường máy ảo, Radmin.

Chương 3: Viết hàm, thủ tục, ràng buộc toàn vẹn truy vấn trên môi trường phân tán.

Chương 4: Demo các mức cô lập (ISOLATION LEVEL) trong môi trường phân tán.

Chương 5: Thực hiện tối ưu hóa truy vấn trên môi trường phân tán 1 câu truy vấn đơn giản.

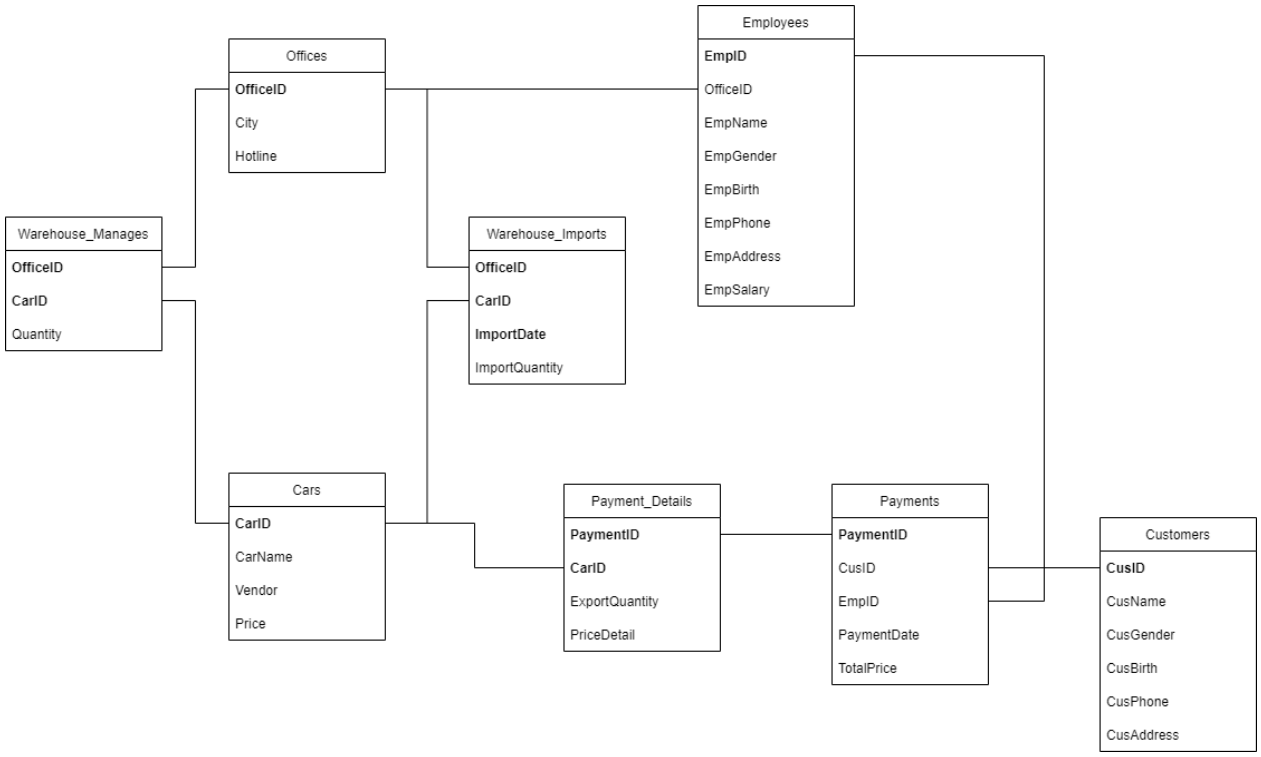
Chương 6: Cơ chế nhân bản trong Oracle.

CHƯƠNG 2: THIẾT KẾ CSDL PHÂN TÁN TRÊN MÔI TRƯỜNG MÁY ẢO, RADMIN

1. Thiết kế cơ sở dữ liệu

CSDL quản lý các chi nhánh bán ô tô ở thành phố New York và Paris.

Lược đồ CSDL của mỗi chi nhánh như sau:



CARS (CARID, CARNAME, VENDOR, PRICE)

Tân từ: Mỗi loại xe có mã xe (CARID) là duy nhất. tên xe (CARNAME), nhà cung cấp (VENDOR), giá bán (PRICE).

CUSTOMER (CUSID, CUSNAME, CUSGENDER, CUSBIRTH, CUSPHONE, CUSADDRESS)

Tân từ: Mỗi khách hàng có mã khách hàng (CUSID) là duy nhất, tên khách hàng (CUSNAME), giới tính của khách hàng (CUSGENDER), ngày sinh của khách hàng (CUSBIRTH), số điện thoại của khách hàng (CUSPHONE), địa chỉ của khách hàng (CUSADDRESS).

OFFICES (OFFICEID, CITY, HOTLINE)

Tân từ: Mỗi chi nhánh (văn phòng) có mã văn phòng (OFFICEID) là duy nhất, để phân biệt với văn phòng khác, tên thành phố chứa văn phòng (CITY), đường dây nóng để liên hệ với chi nhánh (văn phòng) (HOTLINE).

EMPLOYEES (EMPID, EMPNAME, EMPGENDER, EMPBIRTH, EMPADDRESS, EMPPHONE, EMPSALARY, OFFICEID)

Tân từ: Mỗi nhân viên có mã nhân viên dung để phân biệt với nhân viên khác (EMPID) là duy nhất, tên nhân viên (EMPNAME, giới tính nhân viên (EMPGENDER), ngày sinh nhân viên (EMPBIRTH), địa chỉ của nhân viên (EMPADDRESS), số điện thoại của nhân viên (EMPPHONE), mức lương của nhân viên (EMPSALARY).

PAYMENTS (PAYMENTID, PAYMENTDATE, TOTALPRICE, CUSID, EMPID)

Tân từ: Mỗi hóa đơn thanh toán có mã thanh toán (PAYMENTID) là duy nhất để phân biệt với các đơn thanh toán khác, mỗi khách hàng khi thực hiện thanh toán sẽ lưu lại mã khách hàng (CUSID), mã nhân viên (EMPID) tại hóa đơn thanh toán, ngày thanh toán (PAYMENTDATE), tổng số tiền thực hiện thanh toán (TOTALPRICE).

PAYMENT_DETAILS (PAYMENTID, CARID, EXPORTQUANTITY, PRICEDETAIL)

Tân từ: Mỗi khách hàng khi thực hiện thanh toán xe sẽ lưu lại trong chi tiết hóa đơn thanh toán bao gồm mã thanh toán (PAYMENTID), mã xe (CARID), số lượng xe bán (EXPORTQUANTITY), giá ứng với mã xe*số lượng xe bán ra (PRICEDETAIL).

WAREHOUSE_IMPORTS (OFFICEID, CARID, IMPORTDATE, IMPORTQUANTITY)

Tân từ: Mỗi loại xe được nhập vào một chi nhánh được mô tả thông qua các thuộc tính như mã chi nhánh (văn phòng) (OFFICEID), mã xe (CARID), ngày nhập kho (IMPORTDATE), số lượng nhập kho (IMPORTQUANTITY).

WAREHOUSE_MANAGES (OFFICEID, CARID, QUANTITY)

Tân từ: Mỗi xe được quản lý ở mỗi chi nhánh được mô tả thông qua các thuộc tính như mã chi nhánh (văn phòng) (OFFICEID), mã xe (CARID), số lượng hàng trong kho (QUANTITY).

Có 2 chi nhánh: Office1 và Office2

User	**Role**
GiamDoc	CONNECT, CREATE DATABASELINK
QuanLy	CONNECT
NhanVien	CONNECT

Phân quyền:

- **Chi nhánh 1:**

- **GiamDoc**: Xem được toàn bộ thông tin của cả hai chi nhánh.
- **QuanLy:**

Xem được thông tin CUSTOMERS, CARS , PAYMENTS, PAYMENT_DETAILS của cả hai chi nhánh.

Xem được thông tin WAREHOUSE_Manages, WAREHOUSE_imports, EMPLOYEES của chi nhánh mình quản lí.

- **NhanVien:** Xem được thông tin WAREHOUSE_Manages, CARS của chi nhánh mình làm việc.

❖ **Chi nhánh 2:**

- **QuanLy:**

Xem được thông tin CUSTOMERS, CARS, PAYMENTS, PAYMENT_DETAILS của cả hai chi nhánh.

Xem được thông tin WAREHOUSE_Manages, WAREHOUSE_imports, EMPLOYEES của chi nhánh mình quản lí.

- **NhanVien**: Xem được thông tin WAREHOUSE_Manages, CARS của chi nhánh mình làm việc.

2. Thực hiện 10 câu truy vấn

2.1. Liệt kê nhân viên (EmpId, EmpName, EmpSalary) ở cả 2 chi nhánh (Hội)

Query:

```
(SELECT EmpId, EmpName, EmpSalary
FROM EMPLOYEES)
UNION
(SELECT EmpId, EmpName, EmpSalary
FROM EMPLOYEES@off2)
```

Result:

	EMPID	EMPNAME	EMPSALARY
1	1	Anita Jacomb	1730
2	10	Clemens Petrasch	1430
3	11	Florinda Zambon	890
4	12	Marcelo Toon	930
5	13	Dyna Pyffe	810
6	14	Cooper Bowart	840
7	15	Barthel Sellens	1490
8	16	Krishna Ashwell	1870
9	17	Staci Labrom	530
10	18	Kevina Commander	940
11	19	Shir Berge	1180
12	2	Julita Culter	1400
13	20	Petey Ginner	1580
14	21	Gerick Figure	790
15	22	Cthrine Dimbleby	1560
16	23	Ludovika Mars	1690
17	24	Arleyne Mullis	1050
18	25	Wylie Hindmore	1440
19	26	Elisha Brignall	1270
20	27	Ammamaria Koche	650
21	28	Crysta Roussell	960
22	29	Hamlin Siddeley	920
23	3	Helenka Dightham	1680
24	30	Iain Allen	1280
25	31	Anatollo Stear	1080
26	32	Wendie Beveridge	1460
27	33	Lesya Waryk	690
28	34	Dew Cleeves	1720
29	35	Garth Gatenby	610
30	36	Debi Aynscombe	550
31	37	Jarid Wace	820
32	38	Maisey Breed	540
33	39	Red Domotor	1500
34	4	Taddeo MacKenny	1190
35	40	Gardner Brundell	940
36	5	Suellen Romke	1890
37	6	Selinda Screen	1490
38	7	Baird Asker	1680
39	8	Trixie Brittles	1480
40	9	Herby Ruffle	1650

2.2. Liệt kê 5 hóa đơn có giá trị cao nhất, sắp xếp giảm dần theo giá trị (PaymentID, Tong, SL) (Gom nhóm)

Query:

```
(SELECT pm1.PaymentID, TotalPrice, SUM(ExportQuantity) AS SL
FROM Payment_Details pmd1, Payments pm1
WHERE pmd1.PaymentID=pm1.PaymentID
GROUP BY pm1.PaymentID, TotalPrice)
UNION
(SELECT pm2.PaymentID, TotalPrice, SUM(ExportQuantity) AS SL
FROM Payment_Details@off2 pmd2, Payments@off2 pm2
WHERE pmd2.PaymentID=pm2.PaymentID
GROUP BY pm2.PaymentID, TotalPrice)
ORDER BY TOTALPRICE DESC, SL DESC
FETCH FIRST 5 ROWS ONLY;
```

Result:

	PAYMENTID	TOTALPRICE	SUM(EXPORTQUANTITY)
1	54	516100	12
2	26	454900	16
3	97	452000	10
4	62	410500	12
5	2	409500	16

2.3. Cho biết chi nhánh nào bán được đa dạng xe nhất (OficeID, City, SLLoai) (Tính toán)

Query:

```
(SELECT o.OfficeID, City, COUNT(Distinct pd.CarID) AS SLXE
FROM Offices o, Warehouse_Manages wm ,Cars c, payment_details pd
WHERE
   o.OfficeID=wm.OfficeID
```

```
    AND c.CarID=wm.CarID
    AND pd.CarID=c.CarID
    AND ExportQuantity >0
GROUP BY o.OfficeID, City)
UNION
(SELECT o2.OfficeID, City, COUNT(Distinct pd2.CarID) AS SLXE
FROM Offices@off2 o2, Warehouse_Manages@off2 wm2 ,Cars@off2 c2,
payment_details@off2 pd2
WHERE
    o2.OfficeID=wm2.OfficeID
    AND c2.CarID=wm2.CarID
    AND pd2.CarID=c2.CarID
    AND ExportQuantity >0
GROUP BY o2.OfficeID, City)
ORDER BY SLXE
FETCH FIRST 1 ROW WITH TIES;
```

Result:

	OFFICEID	CITY	SLXE
1	1	New York	27
2	2	Paris	27

2.4. Tìm khách hàng đã mua xe ở cả hai chi nhánh (CusID, CusName) (Giao)

Query:

```
SELECT C1.CUSID, CUSNAME
FROM CUSTOMERS C1, PAYMENTS P1
WHERE C1.CUSID = P1.CUSID
INTERSECT
```

```
SELECT C2.CUSID, CUSNAME
FROM CUSTOMERS@off2 C2, PAYMENTS@off2 P2
WHERE C2.CUSID = P2.CUSID
```

Result:

	CUSID	CUSNAME
1	14	Fanny Cadalleder
2	32	Baird Hakonsson
3	44	Penrod Coursor
4	65	Valeda Boshere
5	68	Faber Assel
6	7	Adena Prater
7	71	Matthew MacColl
8	79	Jazmin Scarff
9	81	Damian Sokell
10	88	Nealon Parchment
11	89	Domenico Gobeaux
12	9	Brady Rudyard
13	91	Herbie Goucher

2.5. Tìm khách hàng chỉ mua xe ở chi nhánh Office1 (CusID, CusName) ((Trừ)

Query:

```
SELECT DISTINCT C.CUSID, CUSNAME
FROM CUSTOMERS C, PAYMENTS P1
WHERE C.CUSID=P1.CUSID
MINUS
SELECT DISTINCT C.CUSID, CUSNAME
FROM CUSTOMERS C, PAYMENTS@off2 P2
WHERE C.CUSID=P2.CUSID;
```

Result:

	CUSID	CUSNAME
1	31	Sandye Putt
2	60	Reginauld Benoist
3	77	Bird Cardenas
4	82	Domingo Keese
5	90	Gaven Halwill
6	6	Rora Shildrick
7	87	Winn Corless
8	25	Lizzie Bolin
9	41	Wileen Carnall
10	28	Nerty Engel
11	29	Morten Basnett
12	40	Jonathan Galvan
13	64	Nickolaus Gozzard
14	53	Benjamin Caine
15	21	Alisha Hiskey
16	26	Maryanne Swendell
17	70	Chalmers O'Murtagh

2.6. Tìm loại xe được mua nhiều nhất (CarID, CarName, SL) (Gom nhóm)

Query:

```
(SELECT pd.CarID,CarName, SUM(ExportQuantity) AS SL
FROM Payment_details pd, Cars c
WHERE
   c.CarID=pd.CarID
GROUP BY  pd.CarID, CarName)
UNION
(SELECT pd2.CarID,CarName, SUM(ExportQuantity) AS SL
FROM Payment_details@off2 pd2, Cars@off2 c2
WHERE
   c2.CarID=pd2.CarID
```

```
GROUP BY  pd2.CarID, CarName)
ORDER BY SL DESC
FETCH FIRST 1 ROW WITH TIES;
```

Result:

	CARID	CARNAME	SL
1	23	Edge	20

2.7. Xem số lượng xe hiện có ở cả hai chi nhánh (TOTAL)

Query:

```
SELECT SUM(SLOFFICE) AS TOTAL
FROM
((SELECT SUM(QUANTITY) AS SLOFFICE
FROM WAREHOUSE_MANAGES W1 )
UNION
(SELECT SUM(QUANTITY) AS SLOFFICE
FROM  WAREHOUSE_MANAGES@off2 W2))
```

Result:

	TOTAL
1	1499

2.8. Cho biết tổng số lượng xe đã nhập, đã xuất của từng loại xe trong năm 2020 của chi nhánh Office2 (CarName, SLNhap, SLXuat)

Query:

```
SELECT C.CARID, CARNAME, NVL(SUM(IMPORTQUANTITY), 0) AS SLNhap,
NVL(SUM(EXPORTQUANTITY), 0) AS SLXuat
FROM CARS C
LEFT JOIN WAREHOUSE_IMPORTS@off2 WM2 ON C.CARID=WM2.CARID
LEFT JOIN PAYMENT_DETAILS@off2 PD2 ON C.CARID=PD2.CARID
```

```
GROUP BY C.CARID, CARNAME;
```

Result:

16	25	Diamante	0	3
17	14	CLK-Class	32	6
18	7	Panamera	17	0
19	22	Space	104	12
20	26	Bronco II	22	3
21	9	Sidekick	225	20
22	21	Express 1500	86	3
23	16	Ranger	0	16
24	27	SC	0	14
25	18	Thunderbird	0	5
26	20	LHS	106	6
27	23	Edge	264	20
28	15	Ram 1500	32	2
29	4	4Runner	54	4
30	6	Jimmy	44	13

2.9. Tìm nhân viên bán được nhiều xe nhất trong năm 2016 của cả hai chi nhánh.

Query:

```
(SELECT e.EmpID, EmpName, SUM(ExportQuantity) AS DoanhSo
FROM Employees e, Payments p, Payment_details pd
WHERE
  e.EmpID=p.EmpID
  AND p.PaymentID=pd.PaymentID
  AND EXTRACT(YEAR FROM PaymentDate) = 2016
GROUP BY e.EmpID, EmpName)
UNION
(SELECT e.EmpID, EmpName, SUM(ExportQuantity) AS DoanhSo
```

```
FROM Employees@off2 e, Payments@off2 p, Payment_details@off2 pd
WHERE
 e.EmpID=p.EmpID
 AND p.PaymentID=pd.PaymentID
 AND EXTRACT(YEAR FROM PaymentDate) = 2016
GROUP BY e.EmpID, EmpName)
ORDER BY DOANHSO DESC
FETCH FIRST 1 ROWS ONLY;
```

Result:

	EMPID	EMPNAME	DOANHSO
1	27	Ammamaria Koche	22

2.10. Tìm hóa đơn mua tất cả loại xe hiện có trong công ty có nhà cung cấp là Toyota. (Phép chia)

Query:

```
SELECT *
FROM PAYMENTS P
WHERE NOT EXISTS
  (
  SELECT *
  FROM CARS C
  WHERE VENDOR='Toyota'
  AND NOT EXISTS
    (
    SELECT *
    FROM PAYMENT_DETAILS PD
    WHERE PD.CARID=C.CARID
    AND PD.PAYMENTID=P.PAYMENTID))
```

```
UNION
SELECT *
FROM PAYMENTS@off2 P2
WHERE NOT EXISTS
  (
  SELECT *
  FROM CARS C
  WHERE VENDOR='Toyota'
  AND NOT EXISTS
    (
    SELECT *
    FROM PAYMENT_DETAILS@off2 PD2
    WHERE PD2.CARID=C.CARID
    AND PD2.PAYMENTID=P2.PAYMENTID));
```

Result:

	PAYMENTID	CUSID	EMPID	PAYMENTDATE	TOTALPRICE
1	100	49	40	14-06-2018	228600
2	27	83	4	18-09-2013	234000
3	76	79	13	13-02-2017	187200
4	79	90	38	19-09-2017	170400

CHƯƠNG 3: VIẾT HÀM, THỦ TỤC, RÀNG BUỘC TOÀN VẸN TRUY VẤN TRÊN MÔI TRƯỜNG PHÂN TÁN

1. Trigger

Trigger kiểm tra thông tin xe nhập vào phải có giá trị lớn hơn 0.

```
CREATE OR REPLACE TRIGGER TRG_VALID_CARS_INSERT
BEFORE INSERT OR UPDATE ON CARS FOR EACH ROW
BEGIN
```

```
   IF(:NEW.Price<=0)
   THEN
   RAISE_APPLICATION_ERROR(-20100, 'Gia xe khong hop le, xin vui long nhap lai');
   END IF;
END;
```

Trigger kiểm tra nhân viên phải đạt tối thiểu 18 tuổi khi vào làm việc

```
CREATE OR REPLACE TRIGGER trg_EMP_insert_update
AFTER INSERT OR UPDATE ON EMPLOYEES
FOR EACH ROW
DECLARE
   today DATE;
BEGIN
      SELECT SYSDATE INTO today FROM DUAL;
      IF EXTRACT(YEAR FROM today) - EXTRACT(YEAR FROM :NEW.EMPBIRTH) < 18
THEN
         RAISE_APPLICATION_ERROR(-20100, 'Nhan vien phai dat toi thieu 18 tuoi khi vao
lam viec');
      END IF;
END;
```

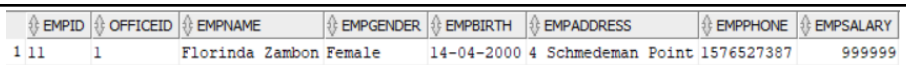
2. Procedure

Procedure	Tìm nhân viên được nhập từ máy và thay đổi mức lương của nhân viên đó theo mức lương được nhập vào	
Procedure Name	changeEmployeeSalary	
Arguments	empID sal	Mã nhân viên Số lương theo yêu cầu

Side-effect	Tìm nhân viên có EmpID trong bảng EMPLOYEE nhập vào từ người quản lí và thay đổi mức lương (EMPSALARY) của nhân viên đó thành sal

Query	CREATE OR REPLACE PROCEDURE changeEmployeeSalary (empID VARCHAR2 ,sal NUMBER)
	AS
	dem NUMBER;
	BEGIN
	SELECT COUNT(Emp1.EmpID) INTO dem
	FROM EMPLOYEES Emp1
	WHERE Emp1.EmpID = empID;
	IF (dem>0) THEN
	UPDATE EMPLOYEES
	SET EMPSALARY = sal
	WHERE EMPID = empID;
	ELSE
	SELECT COUNT(Emp2.EmpID) INTO dem
	FROM EMPLOYEES@off2 Emp2
	WHERE Emp2.EmpID = empID;
	IF (dem > 0) THEN
	UPDATE EMPLOYEES@off2
	SET EMPSALARY = sal
	WHERE EMPID = empID;
	END IF;
	END IF;
	COMMIT;
	END;

Execute Procedure	BEGIN

	changeEmployeeSalary('11', 999999); END;
Kết quả	

	EMPID	OFFICEID	EMPNAME	EMPGENDER	EMPBIRTH	EMPADDRESS	EMPPHONE	EMPSALARY
1	11	1	Florinda Zambon	Female	14-04-2000	4 Schmedeman Point	1576527387	999999

3. Function

Function	**Tính tổng tiền tất cả các hóa đơn khách hàng chi trả**	
Function Name	sumPrice	
Arguments	cusId	Mã khách hàng
Output	Tổng tiền các hóa đơn của khách hàng có mã khách hàng là cusId	

Query	CREATE OR REPLACE FUNCTION sumPrice (customerId CUSTOMERS.CUSID%TYPE) RETURN NUMBER
	AS
	totalPrice NUMBER;
	BEGIN
	SELECT SUM(TOTAL) INTO totalPrice
	FROM (
	SELECT SUM(TOTALPRICE) AS TOTAL FROM PAYMENTS WHERE CUSID=customerId
	UNION
	SELECT SUM(TOTALPRICE) AS TOTAL FROM PAYMENTS@off2 WHERE CUSID=customerId);
	RETURN totalPrice;
	END;	

Execute Procedure	SET SERVEROUTPUT ON; DECLARE t NUMBER;	

	BEGIN
	t := sumPrice('31');
	DBMS_OUTPUT.PUT_LINE('Tong tien: ' || t);
	END;

Kết quả	Tong tien: 160000

CHƯƠNG 4: DEMO CÁC MỨC CÔ LẬP (ISOLATION LEVEL) TRONG MÔI TRƯỜNG PHÂN TÁN

1. LOST UPDATE

Mô tả tình huống:

Nhân viên 1 đang thay đổi thông tin của khách hàng thì có nhân viên 2 đến thay đổi thông tin cũng của chính khách hàng đó nhưng với dữ liệu khác. Từ đó, dẫn đến việc thông tin của nhân viên 2 ghi đè lên trên thông tin của nhân viên 1.

Thực thi:

Time	Office1	Office2
T0	ALTER SESSION SET ISOLATION_LEVEL=READ COMMITTED;	ALTER SESSION SET ISOLATION_LEVEL=READ COMMITTED;
T1	UPDATE Customers SET Cusphone =0123456789 WHERE CusID=7;	
T2		UPDATE Customers@off1 SET Cusphone =1111111111 WHERE Cusid=7;
T3	COMMIT;	
T4		COMMIT;
T5	SELECT Cusphone FROM Customers WHERE CusID=7;	
Result	1111111111	

Cách ngăn chặn:

Time	Office1	Office2
T0	ALTER SESSION SET ISOLATION_LEVEL = SERIALIZABLE;	ALTER SESSION SET ISOLATION_LEVEL=SERIALIZABLE;

T1	UPDATE Customers SET Cusphone =1023456789 WHERE CusID=7;	
T2		UPDATE Customers@off1 SET Cusphone =1111111111 WHERE Cusid=7;
T3	COMMIT;	
T4		COMMIT;
T5	SELECT Cusphone FROM Customers WHERE CusID=7;	
Result	1023456789	Error report - ORA-08177: can't serialize access for this transaction ORA-02063: preceding line from OFF1

2. NON-REPEATEABLE

Mô tả tình huống:

Nhân viên 1 xem thông tin khách hàng lần 1 hoàn tất thì nhân viên 2 truy cập vào hệ thống để thay đổi thông tin khách hàng. Sau đó, nhân viên 1 quay lại để kiểm tra thông tin thì nhận thấy có sự thay đổi so với lần xem đầu tiên.

Thực thi:

Time	Office1	Office2
T0	ALTER SESSION SET ISOLATION_LEVEL=READ COMMITTED;	ALTER SESSION SET ISOLATION_LEVEL=READ COMMITTED;

T1	SELECT Cusphone FROM Customers WHERE CusID=7;	
T2		UPDATE Customers@off1 SET Cusphone = 2222222222 WHERE CusID = 7;
T3		COMMIT;
T4	SELECT Cusphone FROM Customers WHERE CusID=7	SELECT * FROM Customers@off1 WHERE CusID=7
Result	2222222222	2222222222

Cách ngăn chặn:

Time	**Office1**	**Office2**
T0	ALTER SESSION SET ISOLATION_LEVEL = SERIALIZABLE;	ALTER SESSION SET ISOLATION_LEVEL = SERIALIZABLE;
T1	SELECT Cusphone FROM Customers WHERE CusID=7; ---------- 123456789	
T2		UPDATE Customers@off1 SET Cusphone = 2222222222 WHERE CusID = 7;
T3		COMMIT;
T4	SELECT Cusphone FROM Customers WHERE CusID=7;	SELECT Cusphone FROM Customers@off1 WHERE CusID=7;
Result	123456789	2222222222

3. DEADLOCK

Mô tả tình huống:

Nhân viên 1 thay đổi thông tin trên xe A trong lúc đó nhân viên 2 thay đổi thông tin trên xe B. Sau đó, nhân viên 1 chuyển sang thay đổi thông tin trên xe B còn nhân viên 2 cũng làm ngược lại.

Thực thi:

Time	Office1	Office2
T0	ALTER SESSION SET ISOLATION_LEVEL=READ COMMITTED;	ALTER SESSION SET ISOLATION_LEVEL=READ COMMITTED;
T1	UPDATE Cars@off2 SET Price = 20000 WHERE CarID=20;	
T2		UPDATE Cars SET Price = 40000 WHERE CarID = 16;
T3	UPDATE Cars@off2 SET Price = 30000 WHERE CarID=16;	
T4		UPDATE Cars SET Price=50000 WHERE CarID= 20;
T5	Deadlock	
T6	COMMIT;	
T7		COMMIT;
T8		SELECT Price FROM Cars WHERE CarID=20 or CarID=16;

Result	Error report - ORA-00060: deadlock detected while waiting for resource ORA-02063: preceding line from OFF2	Error report - ORA-02049: timeout: distributed transaction waiting for lock ----- 16 40000 20 20000

Cách ngăn chặn:

Time	Office1	Office2
T0	ALTER SESSION SET ISOLATION_LEVEL = SERIALIZABLE;	ALTER SESSION SET ISOLATION_LEVEL = SERIALIZABLE;
T1	UPDATE Cars@off2 SET Price = 20000 WHERE CarID=20;	
T2		UPDATE Cars SET Price = 40000 WHERE CarID = 16;
T3	UPDATE Cars@off2 SET Price = 30000 WHERE CarID=16;	
T4		UPDATE Cars SET Price=50000 WHERE CarID= 20;
T5	WAITING	
	ORA-00060: deadlock detected while waiting for resource ORA-02063: preceding line from OFF2	Error report - ORA-02049: timeout: distributed transaction waiting for lock

T6	COMMIT;	
T7		COMMIT;
T8		SELECT CarID, Price FROM Cars WHERE CarID=20 or CarID=16;
Result		16 40000 20 20000

4. DIRTY READ

Trong hệ quản trị cơ sở dữ liệu Oracle, mức cô lập mặc định là read committed vì vậy không bao giờ có trường hợp Dirty Read.

5. PHANTOM READ

Mô tả tình huống:

Nhân viên 1 kiểm tra thông tin số lượng loại xe của chi nhánh. Sau đó nhân viên 2 xóa thông tin của một loại xe. Khi nhân viên 1 kiểm lại thì thấy thông tin các loại xe bị mất một loại xe.

Thực thi:

Time	Office1	Office2
T0	ALTER SESSION SET ISOLATION_LEVEL=READ COMMITTED;	ALTER SESSION SET ISOLATION_LEVEL=READ COMMITTED;
T1	SELECT COUNT (DISTINCT CarID) AS SoLoaiXe FROM Cars ---------- 31	
T2		DELETE FROM Cars@off1

		WHERE CarID=31; ---------- XÓA 31 Vision Honda 30000
T3		COMMIT;
T4	SELECT COUNT (DISTINCT CarID) AS SoLoaiXe FROM Cars	
Result	30	

Cách ngăn chặn:

Time	Office1	Office2
T0	ALTER SESSION SET ISOLATION_LEVEL = SERIALIZABLE;	ALTER SESSION SET ISOLATION_LEVEL = SERIALIZABLE;
T1	SELECT COUNT(DISTINCT CarID) AS SoLoaiXe FROM Cars ---------- 31	
T2		DELETE FROM Cars@off1 WHERE CarID=31;
T3		COMMIT;
T4	SELECT COUNT(DISTINCT CarID) AS SoLoaiXe FROM Cars	
T5	31	

CHƯƠNG 5: THỰC HIỆN TỐI ƯU HÓA TRUY VẤN TRÊN MÔI TRƯỜNG PHÂN TÁN 1 CÂU TRUY VẤN ĐƠN GIẢN

1. Lược đồ phân mảnh:

- Quan hệ OFFICES phân mảnh nganh chính theo “City”:

OFFICE1= σCity= ‘New York’ (OFFICES)

OFFICE2= σCity= ‘Paris’ (OFFICES)

- Quan hệ EMPLOYEES, PAYMENTS, PAYMENT_DETAILS, WAREHOUSE_MANAGES, WAREHOUSE_IMPORTS phân mảnh ngang dẫn xuất như sau:

EMPLOYEES1=EMPLOYEES⋉OfficeID (OFFICES1)

EMPLOYEES2= EMPLOYEES⋉OfficeID (OFFICES2)

PAYMENTS1=PAYMENT ⋉EmpID (EMPLOYEES1)

PAYMENTS2= PAYMENT ⋉EmpID (EMPLOYEES2)

PAYMENT_DETAILS1=PAYMENT_DETAILS ⋉EmpID (PAYMENT1)

PAYMENT_DETAILS2= PAYMENT_DETAILS ⋉EmpID(PAYMENT2)

WAREHOUSE_MANAGES1=WAREHOUSE_MANAGES⋉OfficeID(OFFICES1)

WAREHOUSE_MANAGES2=WAREHOUSE_MANAGES⋉OfficeID(OFFICES2)

WAREHOUSE_IMPORTS1= WAREHOUSE_IMPORTS⋉OfficeID (OFFICES1)

WAREHOUSE_IMPORTS2= WAREHOUSE_IMPORTS⋉OfficeID (OFFICES2)

Quan hệ CARS, CUSTOMERS được nhân bản ở cả hai chi nhánh.

Câu truy vấn cần tối ưu:

Thống kê số lượng xe đã bán được của từng nhân viên ở Office1, giảm dần theo số lượng.

```
SELECT E.EMPID, EMPNAME, SUM(ExportQuantity) AS DOANHSO
FROM OFFICES O, EMPLOYEES E, PAYMENTS P, PAYMENT_DETAILS PD
WHERE O.OFFICEID=E.OFFICEID
AND E.EMPID=P.EMPID
AND P.PAYMENTID=PD.PAYMENTID
AND EXTRACT(YEAR FROM PAYMENTDATE)=2016
AND O.OFFICEID='1'
GROUP BY E.EMPID, EMPNAME
ORDER BY DOANHSO DESC;
```

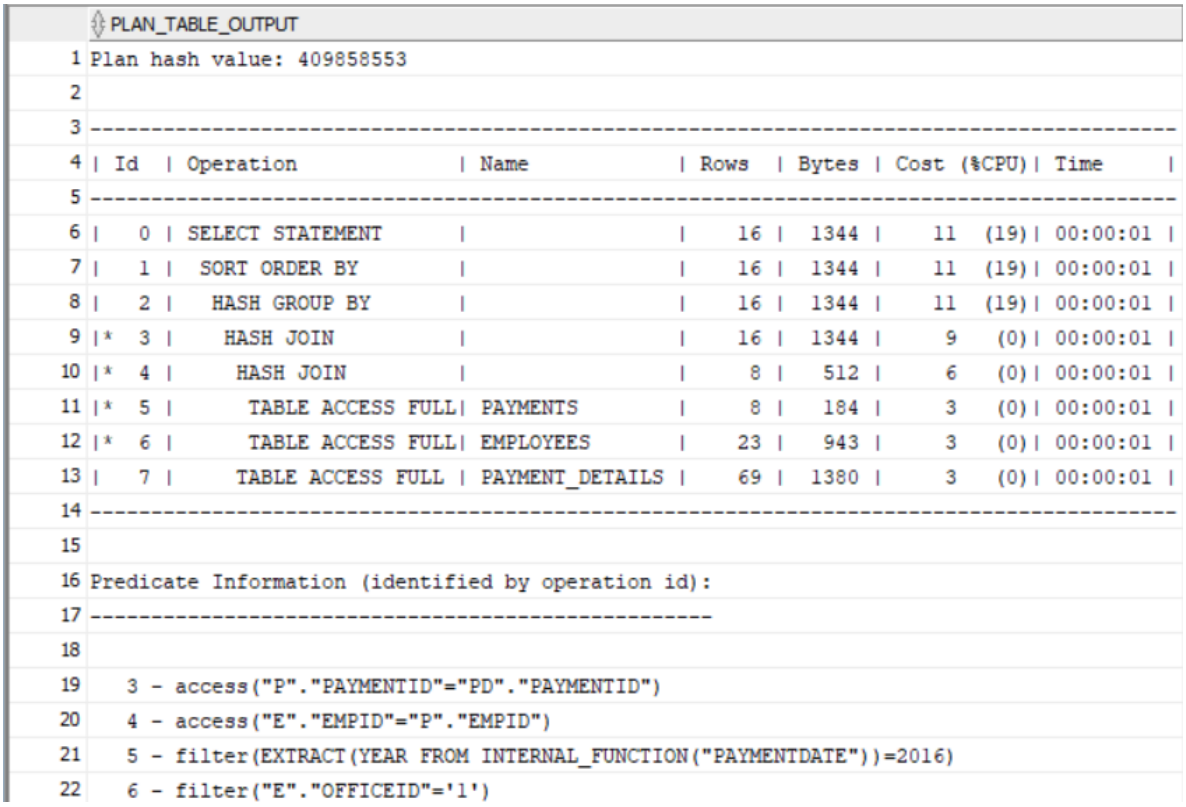
Kết quả ban đầu:

	EMPID	EMPNAME	DOANHSO
1	27	Ammamaria Koche	22
2	10	Clemens Petrasch	16
3	28	Crysta Roussell	10
4	21	Gerick Figure	7
5	23	Ludovika Mars	4
6	17	Staci Labrom	3

Thực hiện Explain query:

```
EXPLAIN PLAN FOR
SELECT E.EMPID, EMPNAME, SUM(ExportQuantity) AS DOANHSO
FROM OFFICES O, EMPLOYEES E, PAYMENTS P, PAYMENT_DETAILS PD
WHERE O.OFFICEID=E.OFFICEID
AND E.EMPID=P.EMPID
AND P.PAYMENTID=PD.PAYMENTID
AND EXTRACT(YEAR FROM PAYMENTDATE)=2016
AND O.OFFICEID='1'
GROUP BY E.EMPID, EMPNAME
ORDER BY DOANHSO DESC;


select * from table(dbms_xplan.display);
```

Kết quả Explain:



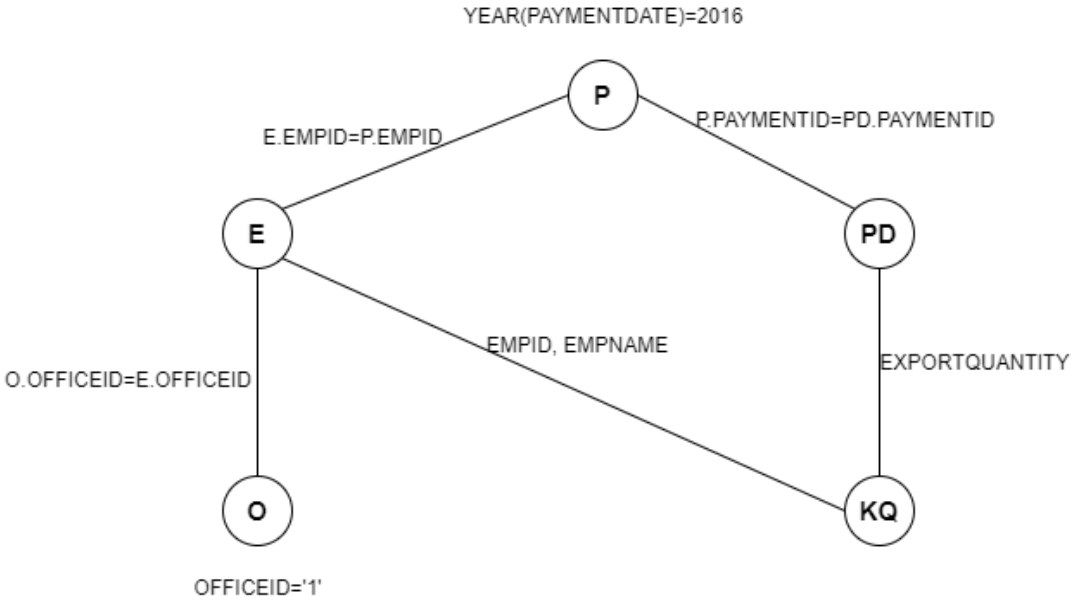
```
PLAN_TABLE_OUTPUT
Plan hash value: 409858553

-------------------------------------------------------------------------------------------
| Id  | Operation            | Name            | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |                 |    16 |  1344 |    11  (19)| 00:00:01 |
|   1 |  SORT ORDER BY       |                 |    16 |  1344 |    11  (19)| 00:00:01 |
|   2 |   HASH GROUP BY      |                 |    16 |  1344 |    11  (19)| 00:00:01 |
|*  3 |    HASH JOIN         |                 |    16 |  1344 |     9   (0)| 00:00:01 |
|*  4 |     HASH JOIN        |                 |     8 |   512 |     6   (0)| 00:00:01 |
|*  5 |      TABLE ACCESS FULL| PAYMENTS        |     8 |   184 |     3   (0)| 00:00:01 |
|*  6 |      TABLE ACCESS FULL| EMPLOYEES       |    23 |   943 |     3   (0)| 00:00:01 |
|   7 |     TABLE ACCESS FULL | PAYMENT_DETAILS |    69 |  1380 |     3   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   3 - access("P"."PAYMENTID"="PD"."PAYMENTID")
   4 - access("E"."EMPID"="P"."EMPID")
   5 - filter(EXTRACT(YEAR FROM INTERNAL_FUNCTION("PAYMENTDATE"))=2016)
   6 - filter("E"."OFFICEID"='1')
```

2. Tối ưu hóa câu truy vấn:

Các từ viết tắt:

Từ viết tắt	Từ đầy đủ
O	OFFICES
E	EMPLOYEES
P	PAYMENTS
PD	PAYMENT_DETAILS

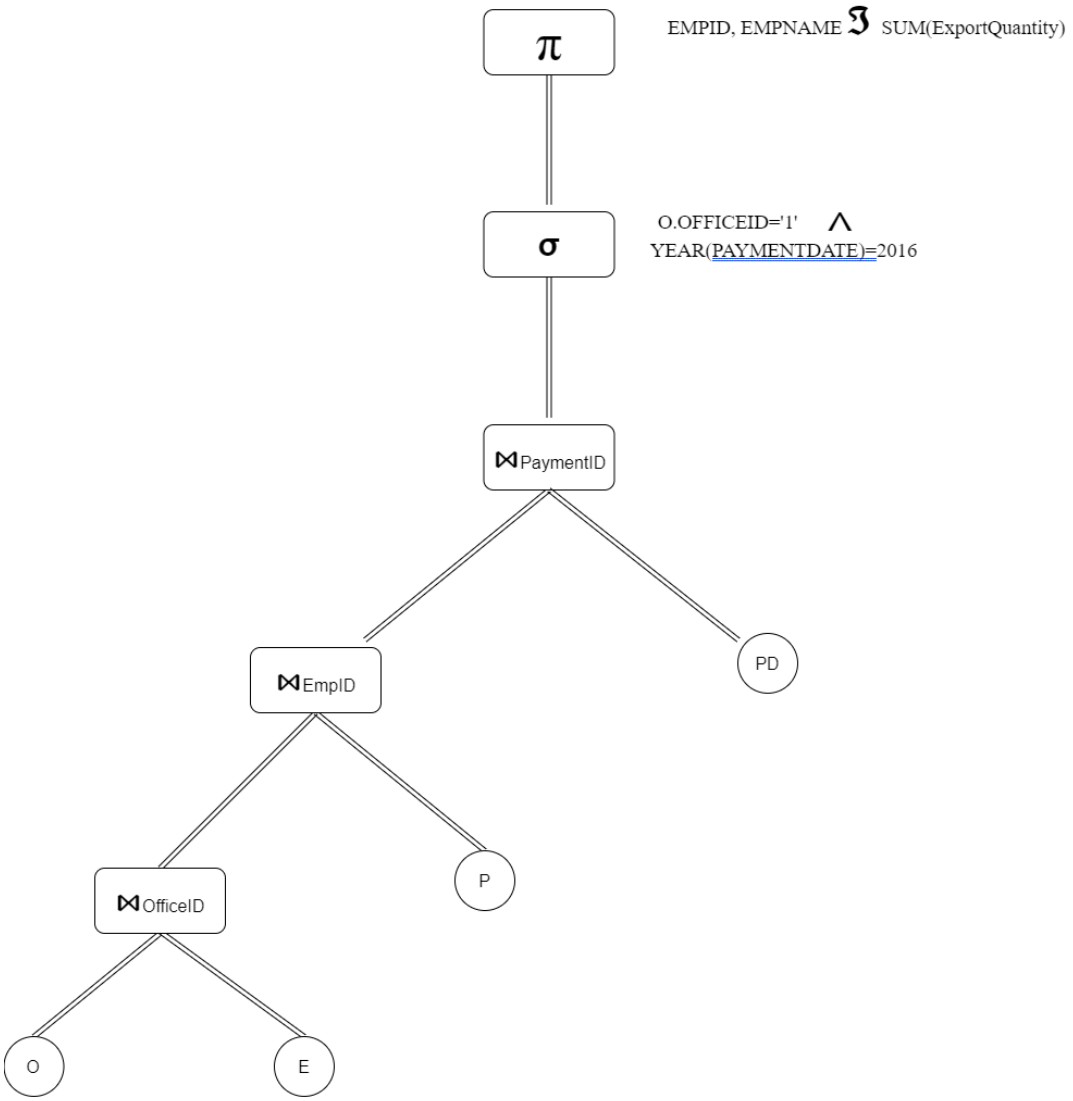
- **Kiểm tra câu truy vấn Q viết đúng ngữ nghĩa hay không**



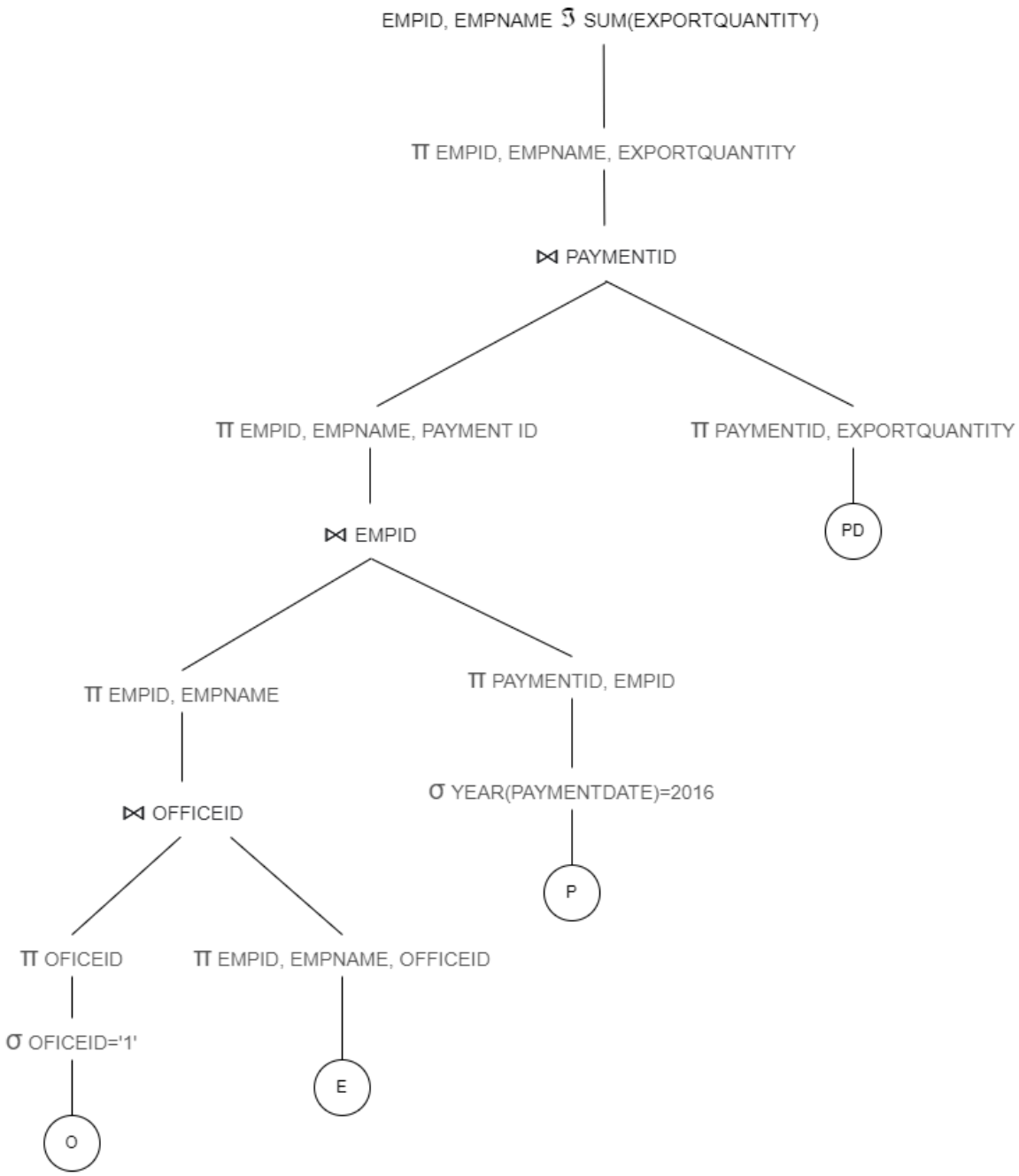
=> Đồ thị liên thông nên câu truy vấn đúng ngữ nghĩa

Phân rã truy vấn để tối ưu hóa toàn cục:

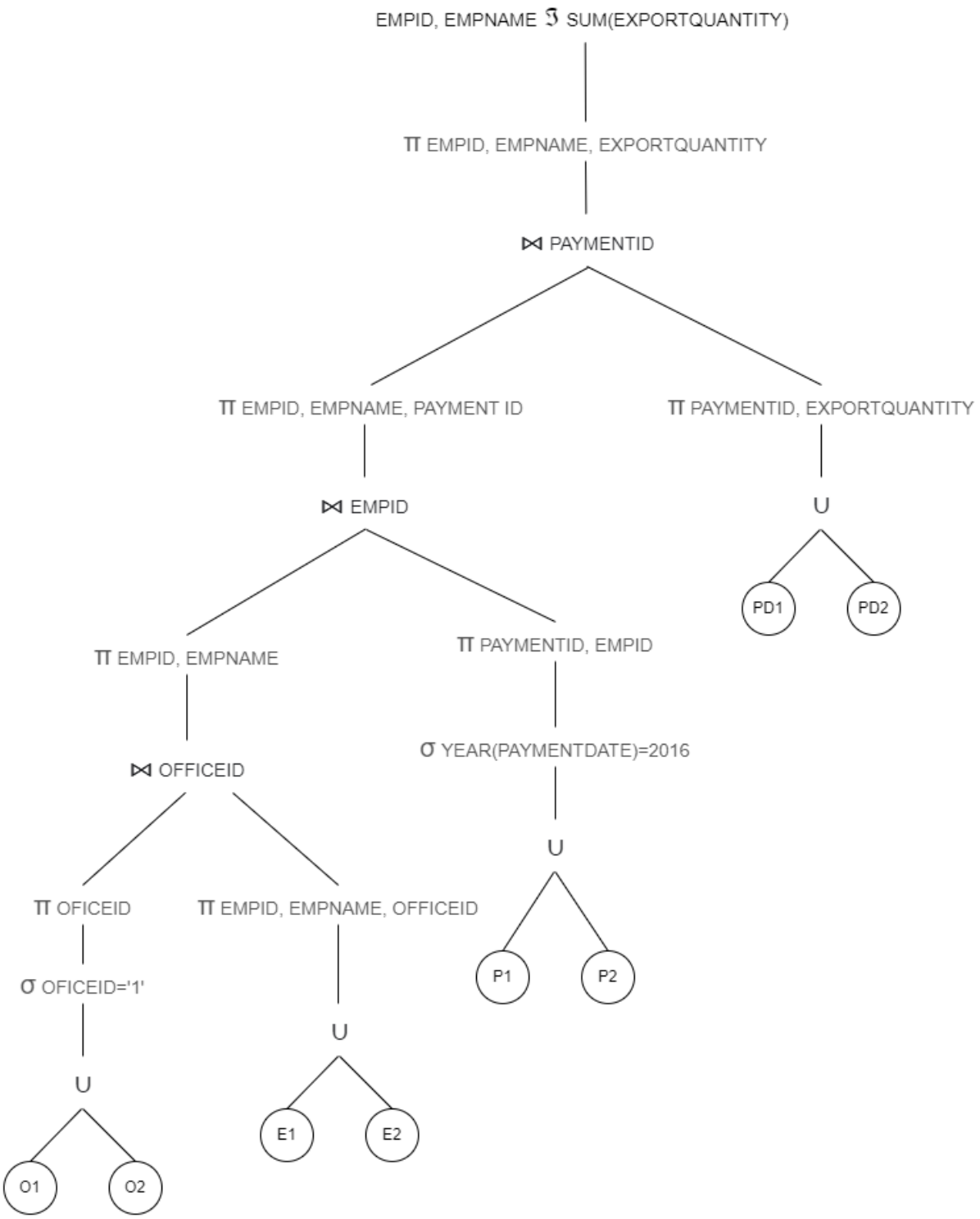
- **Cây truy vấn quan hệ ban đầu**



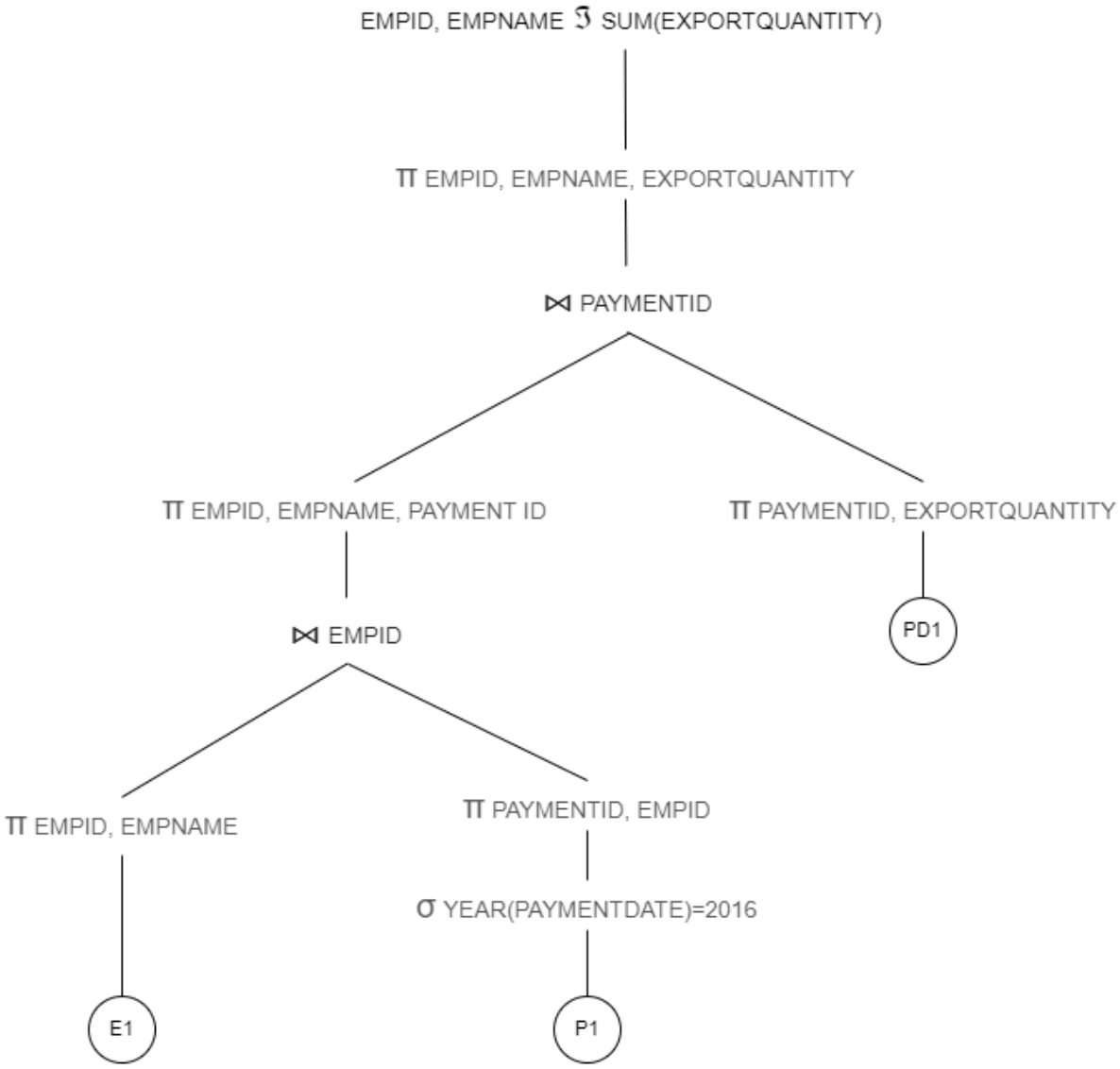
- **Cây truy vấn sau khi tối ưu hóa toàn cục:**



Cây truy vấn dựa vào lược đồ phân mảnh:



Cây truy vấn sau khi tối ưu hóa:



Câu truy vấn sau khi tối ưu hóa:

```
SELECT EMPID, EMPNAME, SUM(EXPORTQUANTITY) AS SL
FROM(
  SELECT EMPID, EMPNAME, EXPORTQUANTITY
  FROM
    (SELECT E1.EMPID, EMPNAME, PAYMENTID
    FROM
```

```
        (SELECT EMPID, EMPNAME
        FROM EMPLOYEES) E1
        JOIN
        (SELECT PAYMENTID, EMPID
        FROM PAYMENTS
        WHERE EXTRACT(YEAR FROM PAYMENTDATE)=2016) P1
        ON E1.EMPID=P1.EMPID
    ) EP1
    JOIN
    (SELECT PAYMENTID, EXPORTQUANTITY
    FROM PAYMENT_DETAILS
    ) PD1
    ON EP1.PAYMENTID=PD1.PAYMENTID)
GROUP BY EMPID, EMPNAME;
```

Thực hiện Explain query câu truy vấn tối ưu hóa:

```
EXPLAIN PLAN FOR
SELECT EMPID, EMPNAME, SUM(EXPORTQUANTITY) AS SL
FROM(
  SELECT EMPID, EMPNAME, EXPORTQUANTITY
  FROM
    (SELECT E1.EMPID, EMPNAME, PAYMENTID
    FROM
      (SELECT EMPID, EMPNAME
      FROM EMPLOYEES) E1
      JOIN
      (SELECT PAYMENTID, EMPID
      FROM PAYMENTS
      WHERE EXTRACT(YEAR FROM PAYMENTDATE)=2016) P1
```

```
      ON E1.EMPID=P1.EMPID
    ) EP1
    JOIN
    (SELECT PAYMENTID, EXPORTQUANTITY
    FROM PAYMENT_DETAILS
    ) PD1
    ON EP1.PAYMENTID=PD1.PAYMENTID)
GROUP BY EMPID, EMPNAME;
```

Explain query sau khi tối ưu hóa:

```
PLAN_TABLE_OUTPUT
Plan hash value: 3687521256

-------------------------------------------------------------------------------------------
| Id  | Operation            | Name            | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |                 |    16 |  1232 |    10  (10)| 00:00:01 |
|   1 |  HASH GROUP BY       |                 |    16 |  1232 |    10  (10)| 00:00:01 |
|*  2 |   HASH JOIN          |                 |    16 |  1232 |     9   (0)| 00:00:01 |
|*  3 |    HASH JOIN         |                 |     8 |   456 |     6   (0)| 00:00:01 |
|*  4 |     TABLE ACCESS FULL| PAYMENTS        |     8 |   184 |     3   (0)| 00:00:01 |
|   5 |     TABLE ACCESS FULL| EMPLOYEES       |    23 |   782 |     3   (0)| 00:00:01 |
|   6 |    TABLE ACCESS FULL | PAYMENT_DETAILS |    69 |  1380 |     3   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("PAYMENTID"="PAYMENTID")
   3 - access("EMPID"="EMPID")
   4 - filter(EXTRACT(YEAR FROM INTERNAL_FUNCTION("PAYMENTDATE"))=2016)
```

CHƯƠNG 6: CƠ CHẾ NHÂN BẢN TRONG ORACLE

1. Nhân bản Oracle là gì?

Oracle Replication cho phép các tổ chức tạo, đồng bộ hóa và phân phối dữ liệu trên nhiều địa điểm, cho phép chia sẻ dữ liệu với người dùng như đối tác hoặc nhà cung cấp, tổng hợp dữ liệu từ các văn phòng khác nhau, cục bộ và toàn cầu. Nó có thể giảm chi phí phân tích với giảm tải truy vấn, bằng cách chia nhỏ quá trình xử lý giao dịch trực tuyến (OLTP) và tạo báo cáo trên các hệ thống khác nhau.

2. .Nhân bản Oracle để làm gì?

Trong một môi trường phân tán, dù là đồng nhất hay không đồng nhất, Oracle Replication cho phép người dùng truy cập thông tin khi nào và ở đâu họ cần. Sao chép cơ sở dữ liệu Oracle tạo ra nhiều bản sao được đồng bộ hóa của cơ sở dữ liệu Oracle cho các mục đích như báo cáo kinh doanh, thử nghiệm, sao lưu để khắc phục sự cố và xử lý dữ liệu phân tán. Phần mềm sao chép của Oracle có thể cung cấp dữ liệu theo thời gian thực để các công ty có thể cải thiện hiệu suất của các cơ sở dữ liệu và ứng dụng quan trọng, cung cấp tính khả dụng của dữ liệu trong toàn tổ chức và hưởng lợi từ các phân tích và báo cáo theo thời gian thực cho những hiểu biết quan trọng về kinh doanh.

3. Nhân bản Oracle hoạt động như thế nào?

Bản sao của Oracle cho phép người dùng truy cập thông tin bằng cách cung cấp dữ liệu trên nhiều vị trí hoặc hệ thống. Bản sao Oracle tạo các bản sao cơ sở dữ liệu được đồng bộ hóa để sử dụng cho các mục đích khác nhau như báo cáo, kiểm tra và sao lưu. Theo mặc định, Oracle hỗ trợ hai hình thức sao chép khác nhau: sao chép cơ bản và nâng cao.

- Sao chép cơ bản (Basic replication): Trong sao chép cơ bản, các bản sao dữ liệu cung cấp quyền truy cập chỉ đọc vào dữ liệu bảng bắt nguồn từ trang web chính/chính. Các ứng dụng có thể truy vấn dữ liệu từ các bản sao dữ liệu cục bộ để tránh tắc nghẽn mạng nhưng phải truy cập dữ liệu tại các trang chính khi cần cập nhật. Oracle hỗ trợ các môi trường sao chép chỉ đọc cơ bản bằng cách sử dụng ảnh chụp bảng chỉ đọc. Kiểu sao chép này cho phép tạo các bản sao chỉ đọc.

- Sao chép nâng cao (Advanced replication): Các tính năng sao chép nâng cao của Oracle cho phép các ứng dụng cập nhật các bản sao bảng trong toàn bộ hệ thống cơ sở dữ liệu được sao chép. Điều này có nghĩa là mỗi bản sao trong hệ thống có thể cung cấp cả quyền truy cập đọc và cập nhật vào bất kỳ dữ liệu bảng nào để đảm bảo tính nhất quán của giao dịch và tính toàn vẹn của dữ liệu.

Hệ thống sao chép Oracle hoạt động với các thành phần cơ bản sau: đối tượng sao chép, nhóm sao chép và trang web sao chép. Các thành phần này rất cần thiết để cho phép sao chép Oracle hoạt động hiệu quả.

- Các đối tượng sao chép: Một đối tượng sao chép có thể được mô tả như một đối tượng cơ sở dữ liệu nằm trong nhiều máy chủ trong một hệ thống cơ sở dữ liệu phân tán. Bất kỳ cập nhật nào được thực hiện đối với một đối tượng sao chép trên bất kỳ vị trí nào mà nó được tìm thấy đều được thực hiện trên tất cả các bản sao tại các trang web khác. Các đối tượng sao chép có thể được sao chép bằng cách sử dụng các công cụ sao chép của Oracle bao gồm bảng, dạng xem, chỉ mục, trình kích hoạt, trình tự, từ đồng nghĩa, thủ tục, chức năng, gói và thân gói.
- Nhóm sao chép: Nhóm sao chép là tập hợp các đối tượng sao chép tương tự có liên quan với nhau. Oracle quản lý các đối tượng sao chép bằng cách sử dụng các nhóm sao chép trong môi trường sao chép nơi các đối tượng trong nhóm được xử lý và quản lý cùng nhau. Điều này giúp dễ dàng tổ chức các đối tượng lược đồ để hỗ trợ một ứng dụng cơ sở dữ liệu cụ thể.
- Các trang web sao chép:Một nhóm sao chép có thể tồn tại ở các vị trí sao chép khác nhau. Có hai loại site mà môi trường sao chép có thể có, site chính và site chụp nhanh. Một trang sao chép cũng có thể đồng thời là trang chính và trang chụp nhanh. Trang chủ là trung tâm điều khiển để quản lý nhóm sao chép và các đối tượng trong nhóm. Nó duy trì một bản sao hoàn chỉnh của tất cả các đối tượng trong nhóm sao chép. Các trang web chính trong môi trường sao chép nhiều trang chủ giao tiếp trực tiếp với nhau để đảm bảo các thay đổi về dữ liệu và lược đồ được đồng bộ hóa trong nhóm sao chép. Ảnh chụp nhanh tại một trang ảnh chụp nhanh có thể chứa tất cả hoặc một tập hợp con của dữ liệu bảng trong một nhóm chính. Chúng có thể chứa hình

ảnh hoặc ảnh chụp nhanh của bảng từ một thời điểm nhất định. Ảnh chụp nhanh được làm mới định kỳ để đồng bộ hóa nó với trang chính.

4. Các phương pháp nhân bản cơ sở dữ liệu Oracle

Việc chọn cách tiếp cận phù hợp để sao chép Oracle của bạn phụ thuộc vào một số yếu tố, bao gồm mục tiêu sao chép của bạn, kích thước của cơ sở dữ liệu, hiệu suất của hệ thống nguồn bị ảnh hưởng như thế nào và liệu bạn có cần sao chép đồng bộ hay sao chép không đồng bộ hay không. Dưới đây là một số cách phổ biến để sao chép cơ sở dữ liệu Oracle.

4.1. Kết xuất đầy đủ và tải (Full Dump and Load)

Trong phương pháp này, bạn bắt đầu bằng cách chọn một bảng mà bạn muốn sao chép. Tiếp theo, bạn xác định khoảng thời gian sao chép (có thể là 4, 8 hoặc 12 giờ) theo yêu cầu của bạn. Đối với mỗi khoảng thời gian, bảng sao chép của bạn được truy vấn và snapshot được tạo. Snapshot được sử dụng để thay thế cho snapshot trước đó.

Cách tiếp cận này hiệu quả đối với các bảng nhỏ (thường dưới 100 triệu hàng). Tuy nhiên, khi bảng tăng kích thước, bạn sẽ phải dựa vào chiến lược sao chép đáng tin cậy hơn. Đó là bởi vì phải mất một khoảng thời gian đáng kể để thực hiện kết xuất.

4.2. Incremental Approach (Table Differencing)

Table Differencing là một cách tiếp cận trong đó một bản sao của bảng nguồn được so sánh định kỳ với một phiên bản cũ hơn của bảng và sự khác biệt được trích xuất.

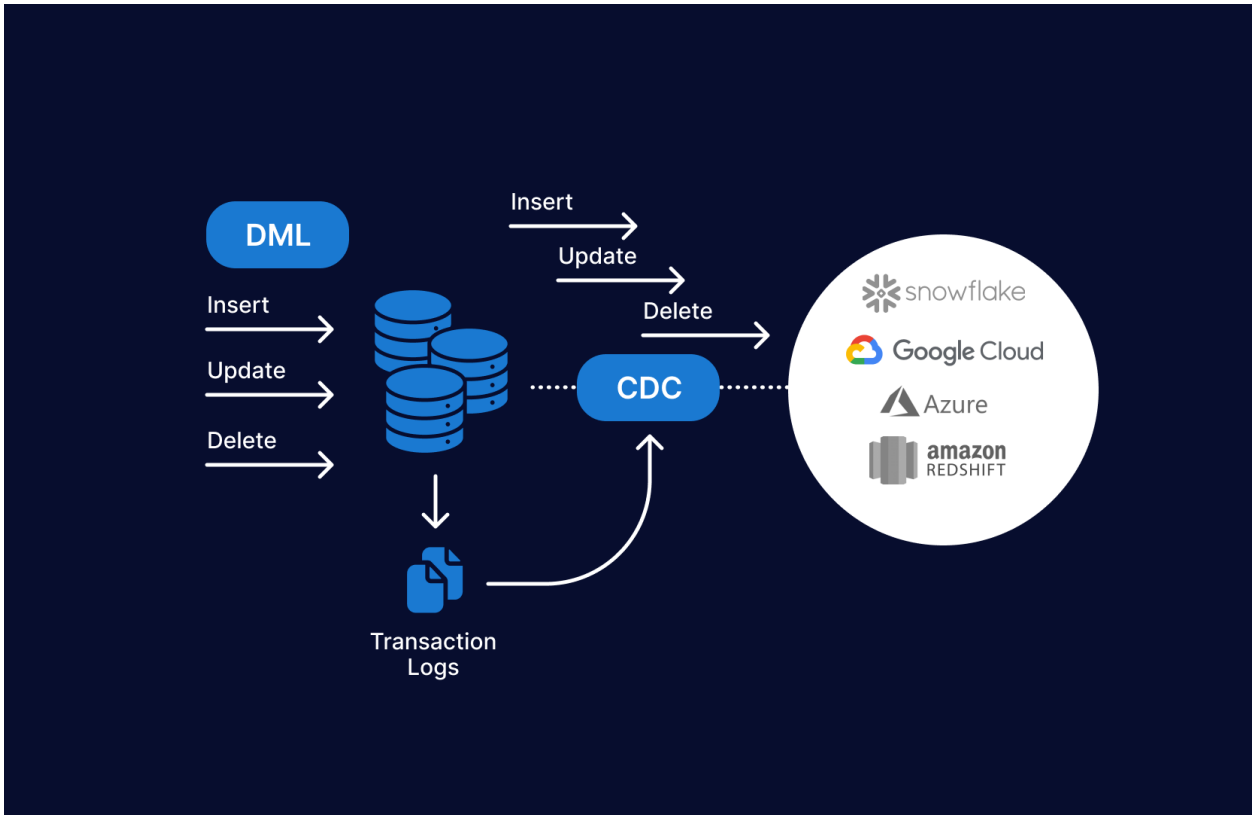
Cách tiếp cận gia tăng cung cấp chế độ xem chính xác về dữ liệu đã thay đổi trong khi chỉ sử dụng các tập lệnh SQL gốc. Tuy nhiên, phương pháp này có thể dẫn đến chi phí tính toán và vận chuyển cao. Ngoài ra, nó không lý tưởng để khôi phục dữ liệu.

4.3. Trigger-Based Approach

Cách tiếp cận này phụ thuộc vào trigger - một chức năng bạn có thể thiết lập để thực thi tự động bất cứ khi nào có thay đổi dữ liệu trong hệ thống cơ sở dữ liệu của bạn.

Oracle đi kèm với một procedure được lưu trữ để đặt bất kỳ trigger nào và theo dõi bảng nguồn để cập nhật. Trigger giúp đạt được bản sao đồng bộ. Tuy nhiên, cách tiếp cận này có thể ảnh hưởng đến hiệu suất của cơ sở dữ liệu nguồn vì các trigger khiến các giao dịch bị trì hoãn.

4.4. Change Data Capture



Change data capture (CDC) là một quy trình phần mềm được quản lý xác định các hàng trong bảng nguồn đã được sửa đổi sau lần sao chép cuối cùng. Điều này làm cho nó hiệu quả hơn và nhanh hơn các phương pháp khác, đặc biệt là những phương pháp sao chép toàn bộ bảng ở mỗi chu kỳ sao chép và sao chép ngay cả những hàng không bị thay đổi.

CDC sao chép các hoạt động create-update-delete (CUD), được viết bằng SQL thông qua các lệnh sau: INSERT, UPDATE và DELETE

Đây là lý do tại sao CDC là một cách tiếp cận tốt hơn để sao chép cơ sở dữ liệu Oracle của bạn:

- Vì CDC chỉ hoạt động với các hàng đã thay đổi, nên CDC sẽ gửi ít dữ liệu hơn từ nguồn tới bản sao, gây tải tối thiểu trên mạng.
- Việc triển khai CDC đúng cách đảm bảo rằng các hoạt động sao chép không ảnh hưởng đến cơ sở dữ liệu sản xuất của bạn. Bằng cách này, bạn có thể giải phóng tài nguyên cho các giao dịch.
- Với CDC, bạn có thể tích hợp dữ liệu theo thời gian thực, giúp bạn xây dựng các phân tích phát trực tuyến..

5. DEMO

Thực hiện nhân bản table CARS, CUSTOMERS ở Office 1 về Office 2:

Tại Office 1:

- Tạo Materialized View Log cho 2 table Customers và Cars

```
CREATE MATERIALIZED VIEW LOG ON CUSTOMERS;
CREATE MATERIALIZED VIEW LOG ON CARS;
```

Tại Office 2 (Sử dụng database link off1 có sẵn để connect tới Office 1):

- Tạo Materialized View cho 2 table với option là tự động Refresh mỗi 60 giây.

```
-- MView Customers
CREATE MATERIALIZED VIEW CUSTOMERS_REP
REFRESH FORCE START WITH SYSDATE NEXT SYSDATE+(1/1440)
AS SELECT * FROM CUSTOMERS@off1;
 -- MView Cars
CREATE MATERIALIZED VIEW CARS_REP
REFRESH FORCE START WITH SYSDATE NEXT SYSDATE+(1/1440)
AS SELECT * FROM CARS@off1;
```

Sau khi tạo Materialized View, sẽ tạo ra CUSTOMERS_REP, CARS_REP có dữ liệu được nhân bản giống với table CUSTOMERS, CARS của Office 1.

Kiểm tra kết quả:

- Insert dữ liệu mới vào 2 table ở Office 1 và commit (Sử dụng tài khoản ở máy Office 1):

```
INSERT INTO CUSTOMERS VALUES ('UT', 'Customer test', 'Gender', to_date('1-1-1999', 'dd-mm-yyyy'), '01233456789', 'HCM');
INSERT INTO CARS VALUES ('CT', 'Car test', 'Vendor test', 2000);
COMMIT;
```

Kiểm tra dữ liệu ở Office 2 (đợi 60 giây) (Sử dụng tài khoản ở máy Office 2):

SELECT * FROM CUSTOMERS_REP WHERE CUSID='UT';

	CUSID	CUSNAME	CUSGENDER	CUSBIRTH	CUSPHONE	CUSADDRESS
1	UT	Customer test	Gender	01-JAN-99	01233456789	HCM

SELECT * FROM CARS_REP WHERE CARID='CT';

	CARID	CARNAME	VENDOR	PRICE
1	CT	Car test	Vendor test	2000

Dữ liệu đã tự động nhân bản thành công.

TÀI LIỆU THAM KHẢO

[1] **https://docs.oracle.com/en/**

[2]