

Oracle Database

Section 1. Giới thiệu Oracle Database

Cơ sở dữ liệu là gì

Cơ sở dữ liệu là một tập hợp có tổ chức các dữ liệu có cấu trúc được lưu trữ điện tử trong hệ thống máy tính.

Trước khi có hệ thống cơ sở dữ liệu, cấu trúc flat file (file phẳng) thường được sử dụng để lưu trữ dữ liệu. Ví dụ: đây là tệp giá trị được phân tách bằng dấu phẩy (CSV) lưu trữ thông tin employee:

```
first name, last name, phone
John, Doe, (408)-245-2345
Jane, Doe, (503)-234-2355
...
```

Tệp CSV có ba cột được gọi là field và hàng được gọi là record. Khi số lượng hàng trong flat file tăng lên, ví dụ: hàng triệu hàng, nó sẽ không thể quản lý được.

Vào những năm 1970, bắt đầu dùng mô hình quan hệ để quản lý cơ sở dữ liệu. Mô hình quan hệ giải quyết nhiều vấn đề do mô hình flat file gây ra. Theo mô hình của ông, dữ liệu được tổ chức theo các thực thể và thuộc tính, thay vì kết hợp mọi thứ trong một cấu trúc duy nhất.

Entity - Thực thể là một người, địa điểm hoặc đồ vật và các thuộc tính mô tả người, địa điểm và đồ vật. Ví dụ: bạn có thể sử dụng mô hình quan hệ để sắp xếp thông tin nhân viên thành một thực thể employee với các thuộc tính: tên, họ và số điện thoại:

Employees	
EmployeeId	int
FirstName	varchar2(100)
LastName	varchar2(100)
Phone	varchar2(15)

Mỗi employee có thể có một hoặc nhiều liên hệ, có thể tạo một thực thể contact và liên kết thực thể employee với thực thể contact thông qua mối quan hệ được gọi là một-nhiều.



Mô hình quan hệ tốt hơn mô hình flat file vì nó loại bỏ dữ liệu trùng lặp, ví dụ: nếu bạn đặt thông tin employee và thông tin contact trên cùng một tệp. Employee có nhiều contact sẽ xuất hiện ở nhiều hàng.

Hệ thống quản lý cơ sở dữ liệu quan hệ, hay gọi tắt là **RDBMS** , **quản lý dữ liệu quan hệ**. Cơ sở dữ liệu Oracle là một RDBMS có thị phần lớn nhất.

Bên cạnh Cơ sở dữ liệu Oracle, còn có các sản phẩm RDBMS khác. Dưới đây là một số điều đáng chú ý:

- SQL Server của Microsoft.
- MySQL – cơ sở dữ liệu nguồn mở phổ biến nhất, cũng của Oracle.
- PostgreSQL – cơ sở dữ liệu nguồn mở tiên tiến nhất.

Section 2. Cài đặt Oracle Database Server

Cài đặt cơ sở dữ liệu Oracle

Để cài đặt cơ sở dữ liệu Oracle trên máy tính, cần tải xuống file installer từ trang tải xuống của trang web Oracle.

Sau khi có các tệp cài đặt ở định dạng ZIP, cần giải nén chúng vào một thư mục cụ thể trên máy tính của mình.

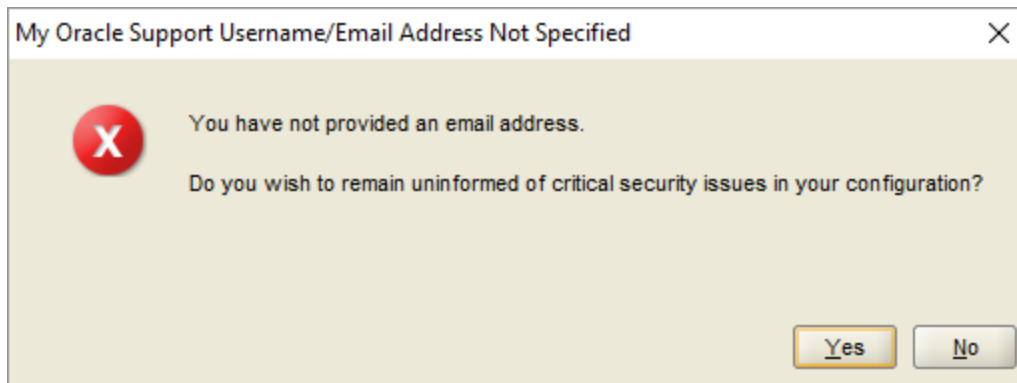
Hình ảnh sau đây thể hiện cấu trúc thư mục chứa các file cài đặt Oracle sau khi giải nén.



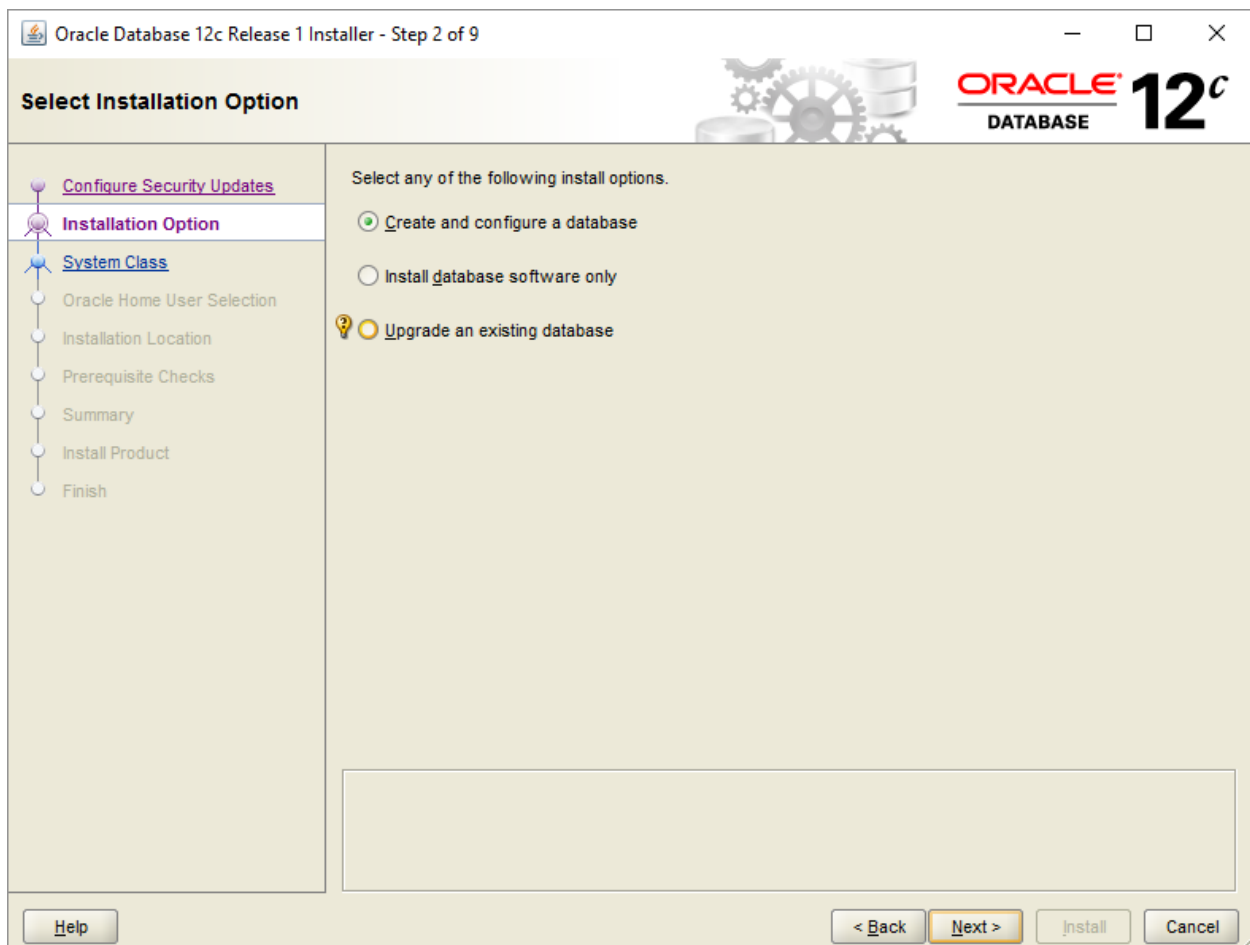
Bây giờ cần nhấp đúp vào file setup.exe để bắt đầu quá trình cài đặt. Sẽ có 9 bước, hầu hết được thực hiện tự động.

Bước 1 . Trình cài đặt yêu cầu bạn cung cấp địa chỉ email của mình để nhận các vấn đề và cập nhật bảo mật mới nhất. Bạn có thể bỏ qua nó bằng cách nhấp vào nút Tiếp theo:

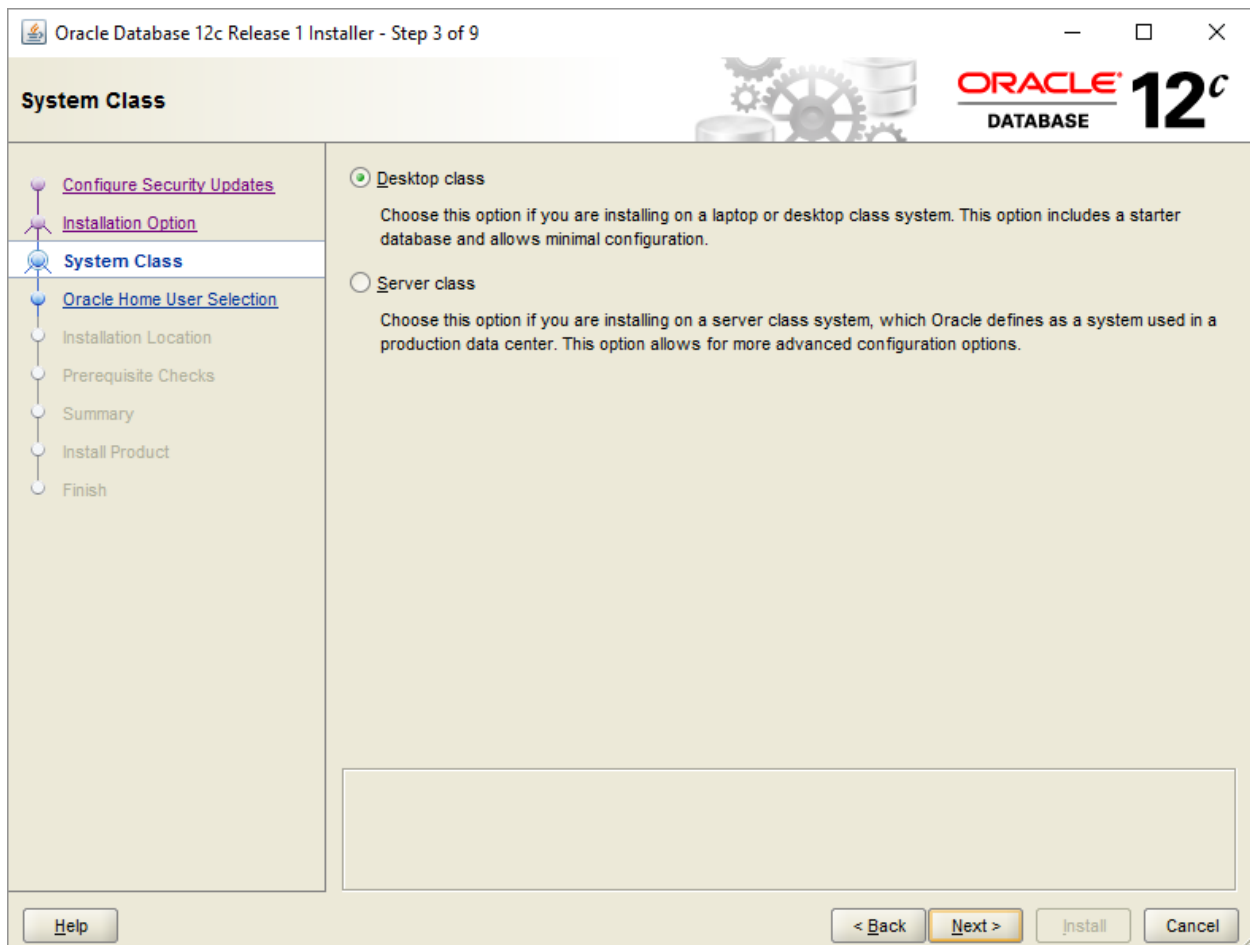
Do mình không cung cấp địa chỉ email nên trình cài đặt cơ sở dữ liệu Oracle xác nhận, bạn chỉ cần nhấn nút No để tiếp tục.



Bước 2. Ở bước 2, trình cài đặt Oracle sẽ hỏi bạn xem muốn tạo và đặt cấu hình cơ sở dữ liệu, chỉ cài đặt phần mềm cơ sở dữ liệu hay chỉ nâng cấp cơ sở dữ liệu hiện có. Vì cài đặt cơ sở dữ liệu Oracle lần đầu tiên nên hãy chọn tùy chọn 1 và nhấp vào nút Tiếp theo.



Bước 3. Trình cài đặt cho phép chọn system class. Vì cài đặt Oracle trên máy tính chứ không phải máy chủ nên chọn tùy chọn đầu tiên: desktop-class và nhấp vào nút Next.



Bước 4. Bước này cho phép chỉ định tài khoản người dùng Windows để cài đặt và định cấu hình Oracle Home để tăng cường bảo mật. Chọn tùy chọn thứ ba: "Use Windows Built-in Account".

Oracle Database 12c Release 1 Installer - Step 4 of 9

Specify Oracle Home User

Oracle recommends that you specify a standard Windows User Account (not an Administrator account) to install and configure the Oracle Home for enhanced security. This account is used for running the Windows Services for the Oracle Home. Do not log in using this account to perform administrative tasks.

☐ Use Existing Windows User

User Name:

Password:

☐ Create New Windows User

User Name:

Password:

Confirm Password:

The newly created user is denied Windows logon privileges.

☒ Use Windows Built-in Account

Help < Back Next > Install Cancel

Bước 5. trong bước này, (1) có thể chọn thư mục sẽ cài đặt cơ sở dữ liệu Oracle, (2) tên và mật khẩu cơ sở dữ liệu, (3) pluggable database name.

Oracle Database 12c Release 1 Installer - Step 5 of 9

Typical Install Configuration

Perform full database installation with basic configuration.

1

2

3

Configure Security Updates

Installation Option

System Class

Oracle Home User Selection

Typical Installation

Prerequisite Checks

Summary

Install Product

Finish

Oracle base: C:\app\ Browse...

Software location: C:\app\product\12.1.0\dbhome_1 Browse...

Database file location: C:\app\oradata Browse...

Database edition: Enterprise Edition (6.0GB)

Character set: Default (WE8MSWIN1252)

Global database name: orcl

Administrative password:

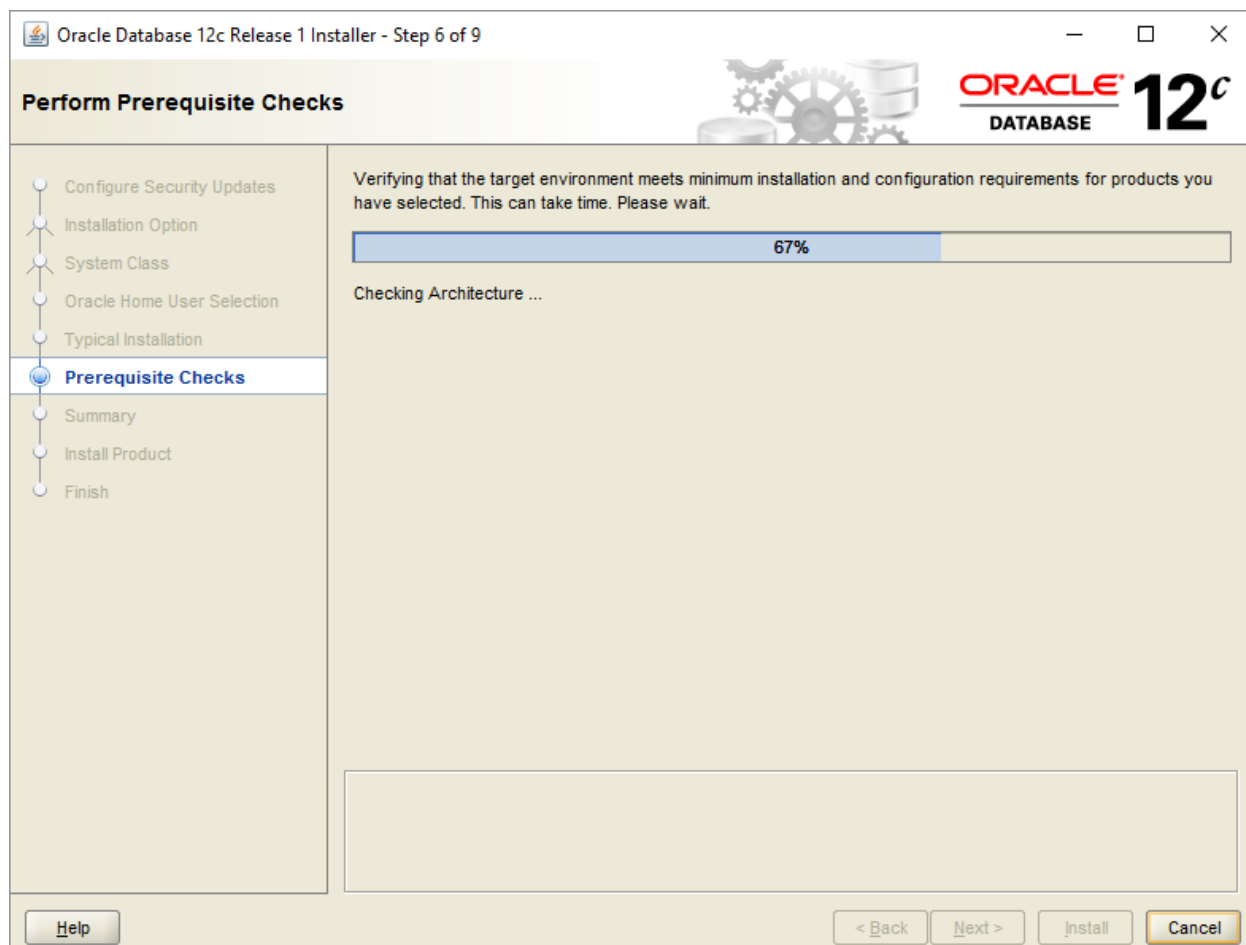
Confirm password:

☒ Create as Container database

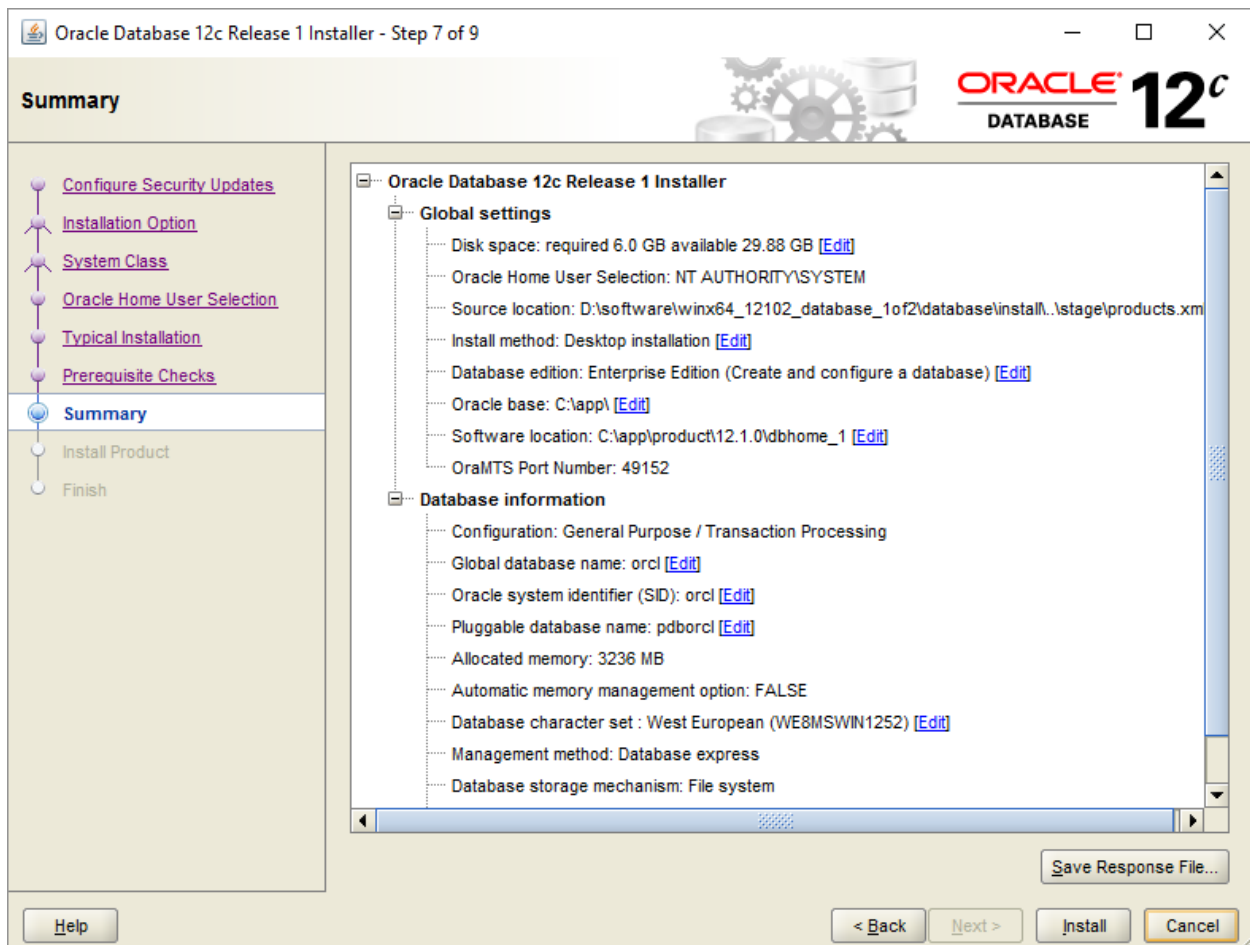
Pluggable database name: pdborcl

Help < Back Next > Install Cancel

Bước 6. Trình cài đặt thực hiện kiểm tra điều kiện tiên quyết.



Bước 7 . Trình cài đặt hiển thị tóm tắt các thông tin như cài đặt chung, thông tin cơ sở dữ liệu, v.v. Cần xem lại thông tin và nhấp vào nút cài đặt nếu mọi thứ đều ổn.



Bước 8. Trình cài đặt bắt đầu cài đặt cơ sở dữ liệu Oracle. Sẽ mất vài phút để hoàn thành, tùy thuộc vào máy tính của bạn.



Install Product

- Configure Security Updates
- Installation Option
- System Class
- Oracle Home User Selection
- Typical Installation
- Prerequisite Checks
- Summary
- Install Product**
- Finish

Progress

0%

Central Inventory is not locked.

Status

✦ Oracle Database installation	In Progress
• Prepare	Pending
• Copy files	Pending
• Setup	Pending
Setup Oracle Base	Pending
Oracle Database configuration	Pending

Details

Retry

Skip

Help

< Back

Next >

Install

Cancel



Install Product

- Configure Security Updates
- Installation Option
- System Class
- Oracle Home User Selection
- Typical Installation
- Prerequisite Checks
- Summary
- Install Product**
- Finish

Progress



Status

✚	Oracle Database installation	In Progress
✓	• Prepare	Succeeded
	• Copy files	Pending
	• Setup	Pending
	Setup Oracle Base	Pending
	Oracle Database configuration	Pending

[Details](#) [Retry](#) [Skip](#)

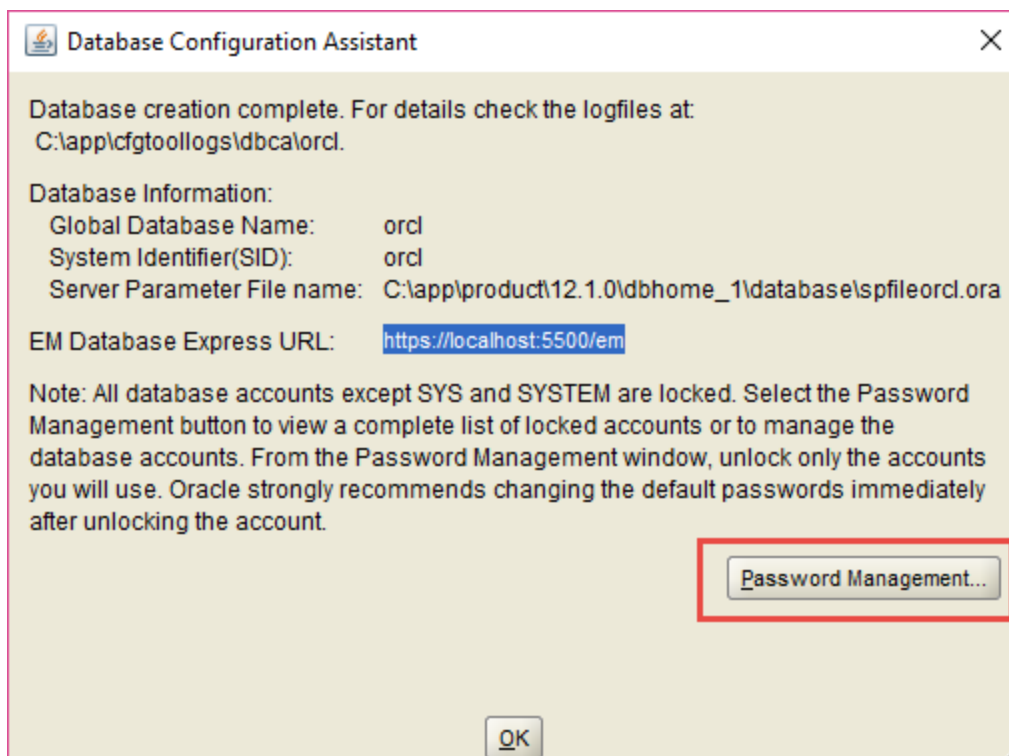
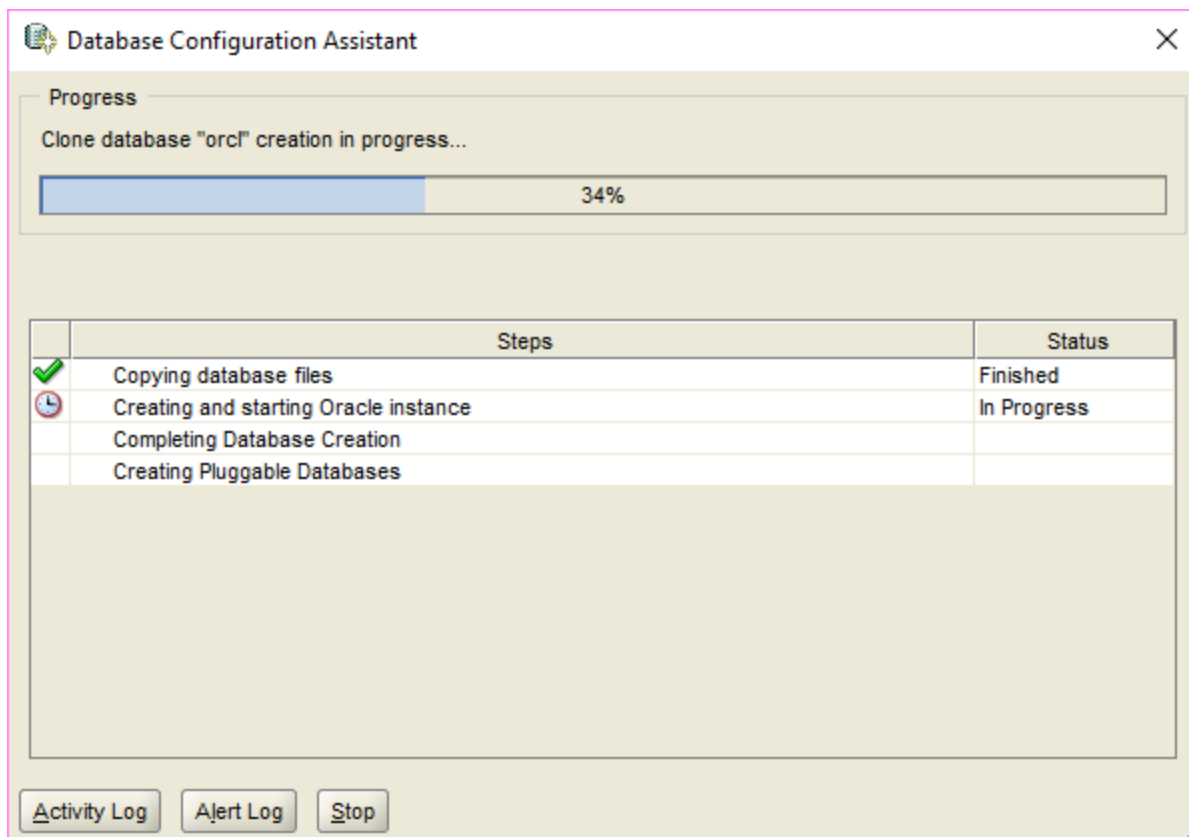
[Help](#)

< Back


Next >

Install

Cancel



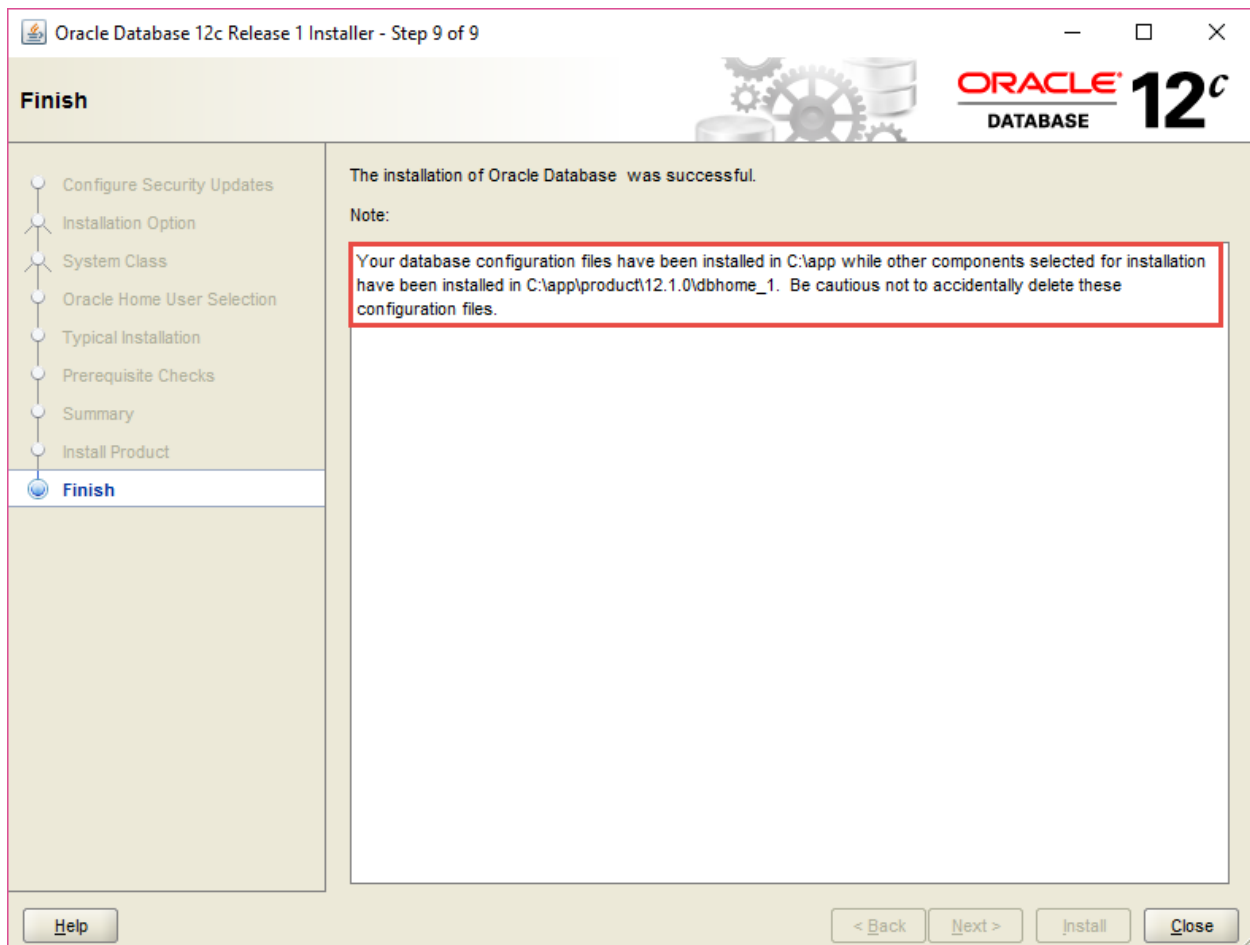
Nhập mật khẩu cho tài khoản SYS và SYSTEM rồi nhấn vào nút OK.

 Password Management ✕

Lock / unlock database user accounts and / or change the default passwords:

User Name	Lock Account? ▲	New Password	Confirm Password
SYS			
SYSTEM			
AUDSYS	✓		
GSMUSER	✓		
SPATIAL_WFS_ADMIN_USR	✓		
SPATIAL_CSW_ADMIN_USR	✓		
APEX_PUBLIC_USER	✓		
SYSDG	✓		
DIP	✓		
SYSBACKUP	✓		
MDDATA	✓		
GSMCATUSER	✓		
SYSKM	✓		
ORACLE_OCM	✓		
OLAPSYS	✓		
SI_INFORMTN_SCHEMA	✓		
SI_INFORMTN_SCHEMA	✓		

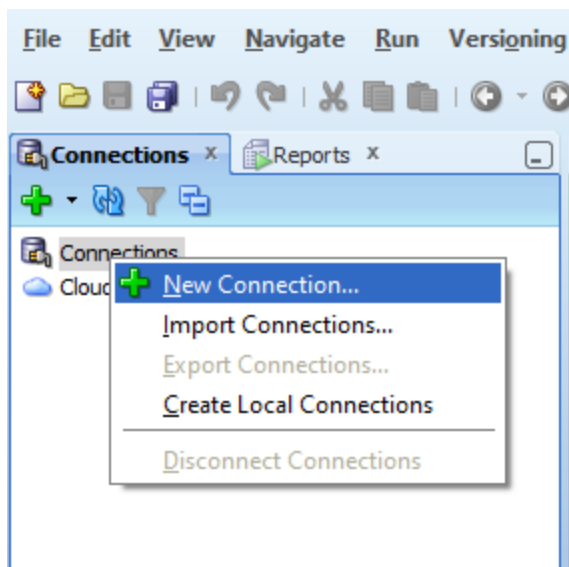
Bước 9. Sau khi quá trình cài đặt hoàn tất thành công, trình cài đặt sẽ thông báo cho bạn như trong ảnh chụp màn hình sau. Bấm vào nút Đóng để đóng cửa sổ.



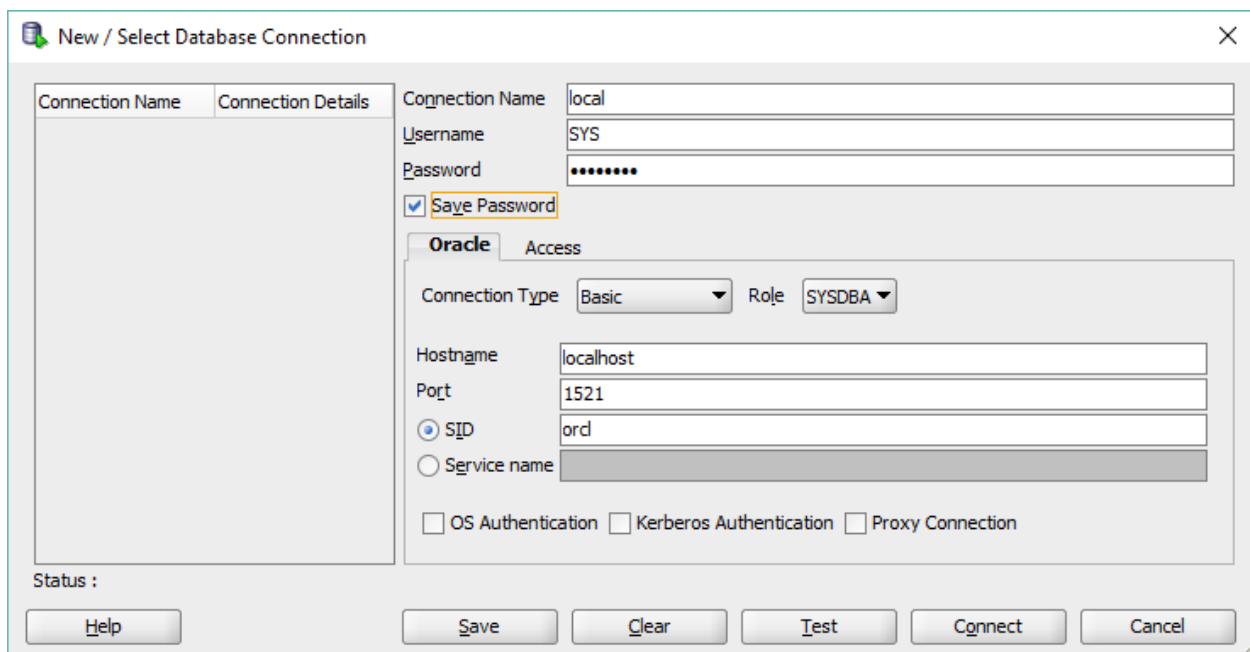
Kết nối với cơ sở dữ liệu Oracle

Đầu tiên, khởi chạy Oracle SQL developer do Oracle cung cấp.

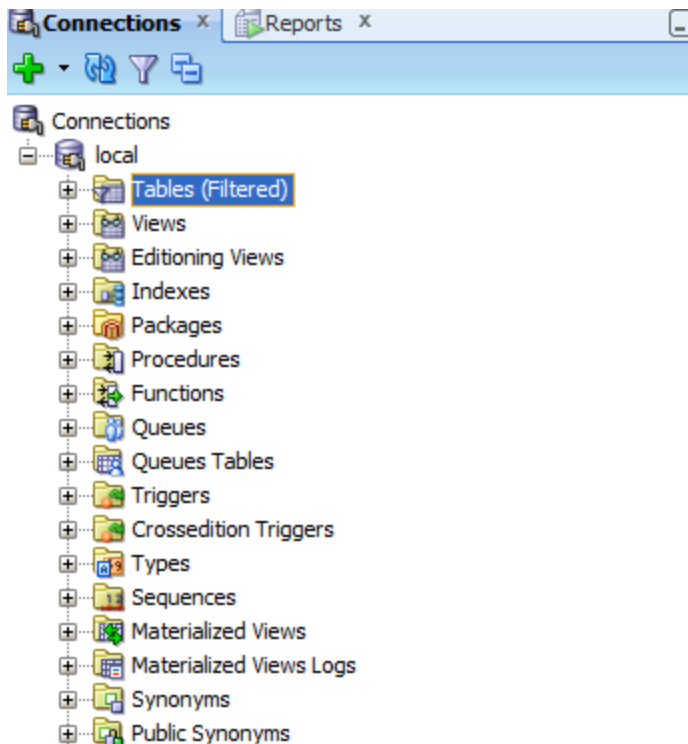
Thứ hai, nhấp chuột phải vào nút connections và chọn mục **New Connection...** để tạo kết nối mới.



Thứ ba, nhập thông tin đã cung cấp trong quá trình cài đặt như trong ảnh chụp màn hình sau. Nhấp vào nút Kết nối để Connect với Oracle Database.



SQL Developer sẽ hiển thị tất cả các đối tượng như hình bên dưới.



Section 3. Create Oracle Sample Database

Creating a new user and granting privileges

Đầu tiên, khởi chạy chương trình SQL*plus bằng dòng lệnh:

```
sqlplus
```

Khi SQL*Plus được khởi chạy, nó sẽ nhắc bạn nhập tên người dùng và mật khẩu. Hãy tiếp tục đăng nhập với tư cách **sys** bằng mật khẩu đã nhập trong quá trình cài đặt Oracle database

```
Enter user-name: sys as sysdba
```

```
Enter password:
```

Trước khi tạo người dùng mới, bạn cần thay đổi cơ sở dữ liệu để open bằng cách thực hiện lệnh sau:

```
SQL> ALTER DATABASE OPEN;
```

```
Database altered.
```

Sau đó, bạn tạo một new user để tạo cơ sở dữ liệu mẫu trong cơ sở dữ liệu bằng câu lệnh sau:

```
SQL> CREATE USER OT IDENTIFIED BY Orc11234;
```

```
User created.
```

Sau đó, bạn cấp đặc quyền cho người dùng bằng cách sử dụng câu lệnh sau:


```
SQL> GRANT CONNECT, RESOURCE, DBA TO OT;
```

```
Grant succeeded.
```

Creating database tables

- Kết nối với cơ sở dữ liệu Oracle với user mới được tạo.
- Thực thi các câu lệnh trong file example database

Oracle Basics

Section 1. Querying data

Tìm hiểu cách truy vấn dữ liệu từ Oracle Database. Bắt đầu với một truy vấn đơn giản cho phép truy xuất dữ liệu từ một bảng duy nhất.

- [SELECT](#) – cách truy vấn dữ liệu từ một bảng.

Tóm tắt : học cách sử dụng SELECT câu lệnh Oracle để truy vấn dữ liệu từ một bảng duy nhất.

Trong Oracle, các bảng bao gồm các cột và hàng. Ví dụ: table customers trong cơ sở dữ liệu sample có các cột sau: **customer_id**, **name**, **address**, **website** và **credit_limit**

CUSTOMERS
* CUSTOMER_ID
NAME
ADDRESS
WEBSITE
CREDIT_LIMIT

Để lấy dữ liệu từ một hoặc nhiều cột của bảng, sử dụng câu lệnh **SELECT** với cú pháp sau:

```
SELECT
column_1,
column_2,
...
FROM
table_name;
```

Trong SELECT:

- Đầu tiên, chỉ định tên bảng mà muốn truy vấn dữ liệu.
- Thứ hai, chỉ ra các cột mà muốn trả về dữ liệu. Nếu có nhiều cột thì các cột cách nhau bằng dấu phẩy (,).

Lưu ý rằng câu lệnh SELECT rất phức tạp và bao gồm nhiều như ORDER BY, GROUP BY, HAVING, JOIN.

Ví dụ về SELECT Oracle

Hãy lấy một số ví dụ về cách sử dụng câu lệnh **SELECT** Oracle để hiểu cách thức hoạt động của nó.

A) truy vấn dữ liệu từ một cột

Để lấy tên khách hàng từ bảng customers, sử dụng câu lệnh sau:

```
SELECT
  name
FROM
  customers;
```

Hình ảnh sau đây minh họa kết quả:

NAME
Kimberly-Clark
Hartford Financial Services Group
Kraft Heinz
Fluor
AECOM
Jabil Circuit
CenturyLink
General Mills
Southern
Thermo Fisher Scientific

B) Truy vấn dữ liệu từ nhiều cột

Để truy vấn dữ liệu từ nhiều cột, chỉ định danh sách tên cột được phân tách bằng dấu phẩy.

Ví dụ sau đây cho thấy cách truy vấn dữ liệu từ các cột customer_id, name, và credit_limit của bảng customer.

```
SELECT
  customer_id,
  name,
  credit_limit
FROM
  customers;
```

Sau đây cho thấy kết quả:

CUSTOMER_ID	NAME	CREDIT_LIMIT
35	Kimberly-Clark	400
36	Hartford Financial Services Group	400
38	Kraft Heinz	500
40	Fluor	500
41	AECOM	500
44	Jabil Circuit	500
45	CenturyLink	500
47	General Mills	600
48	Southern	600
50	Thermo Fisher Scientific	700

C) Truy vấn dữ liệu từ tất cả các cột của bảng

Ví dụ sau lấy tất cả các hàng từ tất cả các cột của bảng customers:

```
SELECT
customer_id,
name,
address,
website,
credit_limit
FROM
customers;
```

Đây là kết quả:

CUSTOMER_ID	NAME	ADDRESS	WEBSITE	CREDIT_LIMIT
1	Raytheon	514 W Superior St, Kokomo, IN	http://www.raytheon.com	100
2	Plains GP Holdings	2515 Bloyd Ave, Indianapolis, IN	http://www.plainsallamerican.com	100
3	US Foods Holding	8768 N State Rd 37, Bloomington, IN	http://www.usfoods.com	100
4	AbbVie	6445 Bay Harbor Ln, Indianapolis, IN	http://www.abbvie.com	100
5	Centene	4019 W 3Rd St, Bloomington, IN	http://www.centene.com	100
6	Community Health Systems	1608 Portage Ave, South Bend, IN	http://www.chs.net	100
7	Alcoa	23943 Us Highway 33, Elkhart, IN	http://www.alcoa.com	100
8	International Paper	136 E Market St # 800, Indianapolis, IN	http://www.internationalpaper.com	100
9	Emerson Electric	1905 College St, South Bend, IN	http://www.emerson.com	100
10	Union Pacific	3512 Rockville Rd # 137C, Indianapolis, IN	http://www.up.com	200

Để thuận tiện, có thể sử dụng dấu hoa thị viết tắt (*) để Oracle trả về dữ liệu từ tất cả các cột của bảng như sau:

```
SELECT * FROM customers;
```

Lưu ý rằng không bao giờ nên sử dụng dấu hoa thị (*) trong dự án. Cách tốt nhất là chỉ định rõ ràng các cột mà muốn truy vấn dữ liệu ngay cả khi muốn truy xuất dữ liệu từ tất cả các cột của bảng.

Section 2. Sorting data

[ORDER BY](#) – sắp xếp tập kết quả của truy vấn theo thứ tự tăng dần hoặc giảm dần.

Giới thiệu về Oracle

Trong Oracle, một bảng lưu trữ các hàng của nó theo thứ tự không xác định bất kể thứ tự hàng nào được chèn vào cơ sở dữ liệu. Để truy vấn các hàng theo thứ tự tăng dần hoặc giảm dần theo một cột, bạn phải hướng dẫn rõ ràng cho Cơ sở dữ liệu Oracle rằng bạn muốn làm như vậy.

Ví dụ: bạn có thể muốn liệt kê tất cả khách hàng theo tên của họ theo thứ tự bảng chữ cái hoặc hiển thị tất cả khách hàng theo thứ tự giới hạn tín dụng từ thấp nhất đến cao nhất.

Để sắp xếp dữ liệu, bạn thêm **ORDER BY** vào **SELECT** như sau:

```
SELECT
    column_1,
    column_2,
    column_3,
    ...
FROM
    table_name
ORDER BY
    column_1 [ASC | DESC] [NULLS FIRST | NULLS LAST],
    column_1 [ASC | DESC] [NULLS FIRST | NULLS LAST],
    ...
```

Để sắp xếp tập kết quả theo một cột, liệt kê cột đó sau ORDER BY.

Sau tên cột là thứ tự sắp xếp có thể là:

- **ASC**: để sắp xếp theo thứ tự tăng dần
- **DESC**: để sắp xếp theo thứ tự giảm dần

Theo mặc định, **ORDER BY** sắp xếp các hàng theo thứ tự tăng dần cho dù có chỉ định **ASC** hay không. Nếu bạn muốn sắp xếp các hàng theo thứ tự giảm dần, bạn sử dụng **DESC** rõ ràng.

- **NULLS FIRST** đặt các giá trị NULL trước các giá trị không phải NULL
- **NULLS LAST** đặt các giá trị NULL sau các giá trị không phải NULL.

Mệnh đề ORDER BY cho phép bạn sắp xếp dữ liệu theo nhiều cột trong đó mỗi cột có thể có thứ tự sắp xếp khác nhau.

Lưu ý rằng **ORDER BY** luôn là cuối cùng trong một câu lệnh **SELECT**

Ví dụ về : ORDER BY

Chúng tôi sẽ sử dụng bảng **customers** trong cơ sở dữ liệu mẫu để trình diễn.

CUSTOMERS
* CUSTOMER_ID
NAME
ADDRESS
WEBSITE
CREDIT_LIMIT

Câu lệnh sau lấy name, address, và credit limit của khách hàng từ bảng **customers**:

```
SELECT
    name,
    address,
    credit_limit
FROM
    customers;
```

NAME	ADDRESS	CREDIT_LIMIT
Kimberly-Clark	1660 University Ter, Ann Arbor, MI	400
Hartford Financial Services Group	15713 N East St, Lansing, MI	400
Kraft Heinz	10315 Hickman Rd, Des Moines, IA	500
Fluor	1928 Sherwood Dr, Council Bluffs, IA	500
AECOM	2102 E Kimberly Rd, Davenport, IA	500
Jabil Circuit	221 3Rd Ave Se # 300, Cedar Rapids, IA	500
CenturyLink	2120 Heights Dr, Eau Claire, WI	500
General Mills	6555 W Good Hope Rd, Milwaukee, WI	600
Southern	1314 N Stoughton Rd, Madison, WI	600

Có thể thấy, thứ tự của các hàng không được chỉ định.

A) Ví dụ sắp xếp các hàng theo 1 cột

Để sắp xếp dữ liệu khách hàng theo tên theo thứ tự bảng chữ cái tăng dần, bạn sử dụng câu lệnh sau:

```
SELECT
    name,
    address,
    credit_limit
FROM
    customers
ORDER BY
    name ASC;
```

NAME	ADDRESS	CREDIT_LIMIT
3M	Via Frenzy 6903, Roma,	1200
ADP	Langstr 14, Zuerich, ZH	700
AECOM	2102 E Kimberly Rd, Davenport, IA	500
AES	33 Fulton St, Poughkeepsie, NY	1200
AIG	12817 Coastal Hwy, Ocean City, MD	2400
AT&T	55 Church Hill Rd, Reading, PA	1200
AbbVie	6445 Bay Harbor Ln, Indianapolis, IN	100
Abbott Laboratories	3310 Dixie Ct, Saginaw, MI	200
Advance Auto Parts	2674 Collingwood St, Detroit, MI	3700
Aetna	200 E Fort Ave, Baltimore, MD	2400

Để sắp xếp tên khách hàng theo thứ tự ABC giảm dần, bạn sử dụng DESC như sau:

```
SELECT
    name,
    address,
    credit_limit
FROM
    customers
ORDER BY
    name DESC;
```

Hình ảnh sau đây thể hiện kết quả khách hàng sắp xếp tên theo thứ tự ABC giảm dần:

NAME	ADDRESS	CREDIT_LIMIT
eBay	Via Del Disegno 194, Milano,	1500
Yum Brands	Ruella Delle Spiriti, Roma,	500
Xerox	9936 Dexter Ave, Detroit, MI	400
Xcel Energy	1540 Stripes Crt, Baden-Daettwil, AG	400
World Fuel Services	Theresienstr 15, Munich,	2400
Whole Foods Market	4200 Yosemite Ave S, Minneapolis, MN	1200
Whirlpool	18305 Van Dyke St, Detroit, MI	200
Western Refining	5565 Baynton St, Philadelphia, PA	2400
Western Digital	33 Pine St, Lockport, NY	1200
WestRock	Chrottenweg, Bern, BE	5000

B) Ví dụ sắp xếp hàng theo nhiều cột

Để sắp xếp nhiều cột, bạn phân tách từng cột trong ORDER BY bằng dấu phẩy.

CONTACTS
* CONTACT_ID
FIRST_NAME
LAST_NAME
EMAIL
PHONE
CUSTOMER_ID

Ví dụ: để sắp xếp các liên hệ theo tên theo thứ tự tăng dần và họ của họ theo thứ tự giảm dần, sử dụng câu lệnh sau:

```
SELECT
    first_name,
    last_name
FROM
    contacts
ORDER BY
    first_name,
    last_name DESC;
```

Xem kết quả sau:

Corliss	Mcneil
Cristine	Bell
Daina	Combs
Daniel	Glass
Daniel	Costner
Darron	Robertson
Debra	Herring
Dell	Wilkinson
Delpha	Golden
Deneen	Hays
Denny	Daniel
Diane	Higgins
Dianne	Sen
Dianne	Derek
Dick	Lamb
Don	Hansen
Doretha	Tyler
Dorotha	Wong

Trong kết quả này:

- Đầu tiên, tên đầu tiên được sắp xếp theo thứ tự tăng dần.
- Thứ hai, nếu hai tên giống nhau thì họ được sắp xếp theo thứ tự giảm dần,

C) Ví dụ sắp xếp các hàng theo vị trí của cột

Không cần chỉ định tên cột để sắp xếp dữ liệu. Nếu muốn, có thể sử dụng vị trí của cột trong ORDER BY.

Xem tuyên bố sau:

```
SELECT
    name,
    credit_limit
FROM
    customers
ORDER BY
    2 DESC,
    1;
```

Trong ví dụ này, vị trí của name là 1 và credit_limit là 2.

C) Ví dụ về sắp xếp các hàng có giá trị NULL

LOCATIONS
* LOCATION_ID
ADDRESS
POSTAL_CODE
CITY
STATE
COUNTRY_ID

Ví dụ: câu lệnh sau sắp xếp các vị trí theo trạng thái theo thứ tự tăng dần và đặt giá trị NULL lên đầu.

```
SELECT
  country_id,
  city,
  state
FROM
  locations
ORDER BY
  state ASC NULLS FIRST;
```

COUNTRY_ID	CITY	STATE
IT	Roma	(null)
IT	Venice	(null)
JP	Hiroshima	(null)
SG	Singapore	(null)
CN	Beijing	(null)
UK	London	(null)
CH	Bern	BE
DE	Munich	Bavaria
US	South San Francisco	California
MX	Mexico City	Distrito Federal,
CH	Geneva	Geneve
IN	Bombay	Maharashtra
UK	Stretford	Manchester
US	South Brunswick	New Jersey
AU	Sydney	New South Wales
CA	Toronto	Ontario
UK	Oxford	Oxford
BR	Sao Paulo	Sao Paulo
US	Southlake	Texas
JP	Tokyo	Tokyo Prefecture
NL	Utrecht	Utrecht
US	Seattle	Washington
CA	Whitehorse	Yukon

Để đặt các giá trị NULL sau các giá trị không phải NULL, sử dụng `NULLS LAST` như trong câu lệnh sau:

```
SELECT
    country_id,
    city,
    state
FROM
    locations
ORDER BY
    state
ASC NULLS LAST;
```

Đây là kết quả:

COUNTRY_ID	CITY	STATE
CH	Bern	BE
DE	Munich	Bavaria
US	South San Francisco	California
MX	Mexico City	Distrito Federal,
CH	Geneva	Geneve
IN	Bombay	Maharashtra
UK	Stretford	Manchester
US	South Brunswick	New Jersey
AU	Sydney	New South Wales
CA	Toronto	Ontario
UK	Oxford	Oxford
BR	Sao Paulo	Sao Paulo
US	Southlake	Texas
JP	Tokyo	Tokyo Prefecture
NL	Utrecht	Utrecht
US	Seattle	Washington
CA	Whitehorse	Yukon
SG	Singapore	(null)

D) Sắp xếp các hàng theo kết quả của hàm hoặc biểu thức

`ORDER BY` cho phép bạn áp dụng một hàm, ví dụ: câu lệnh sau sử dụng `UPPER()` trong `ORDER BY` để sắp xếp tên khách hàng không phân biệt chữ hoa chữ thường:

```
SELECT
    customer_id,
    name
FROM
```

```
customers
ORDER BY
UPPER( name );
```

Sau đây minh họa kết quả:

CUSTOMER_ID	NAME
190	3M
19	Abbott Laboratories
4	AbbVie
287	ADP
168	Advance Auto Parts
41	AECOM
80	AES
138	Aetna
16	Aflac
141	AIG

E) Ví dụ sắp xếp theo ngày

ORDERS
* ORDER_ID
CUSTOMER_ID
STATUS
SALESMAN_ID
ORDER_DATE

Ví dụ này sử dụng ORDER BY để sắp xếp đơn hàng theo ngày đặt hàng:

```
SELECT
    order_id,
    customer_id,
    status,
    order_date
FROM
    orders
ORDER BY
    order_date DESC;
```

ORDER_ID	CUSTOMER_ID	STATUS	ORDER_DATE
88	6	Shipped	01-NOV-17
94	1	Shipped	27-OCT-17
1	4	Pending	15-OCT-17
14	48	Shipped	28-SEP-17
15	49	Shipped	27-SEP-17
17	17	Shipped	27-SEP-17
36	51	Shipped	05-SEP-17
57	68	Shipped	24-AUG-17
28	6	Canceled	15-AUG-17
29	44	Shipped	14-AUG-17
31	46	Canceled	12-AUG-17
30	45	Shipped	12-AUG-17
60	1	Shipped	30-JUN-17
21	21	Pending	27-MAY-17
20	20	Shipped	27-MAY-17
40	55	Shipped	11-MAY-17

Section 3. Filtering data

DISTINCT

DISTINCT được sử dụng trong **SELECT** để lọc các hàng trùng lặp trong tập kết quả. Nó đảm bảo rằng các hàng trả về là duy nhất cho cột hoặc các cột được chỉ định trong **SELECT**.

Sau đây minh họa cú pháp của SELECT DISTINCT:

```
SELECT
    DISTINCT column_1,
FROM
    table_name;
```

Trong câu lệnh này, các giá trị trong column_1 được so sánh để xác định các giá trị trùng lặp.

Để lấy dữ liệu duy nhất dựa trên nhiều cột, chỉ cần xác định danh sách cột trong SELECT như sau:

```
SELECT
    DISTINCT column_1,
    column_2,
    column_3
FROM
    table_name;
```

Trong cú pháp này, sự kết hợp của các giá trị trong column_1, column_2 và column_3 được sử dụng để xác định tính duy nhất của dữ liệu.

DISTINCT chỉ có thể được sử dụng trong SELECT.

Các ví dụ DISTINCT của Oracle CHỌN

Hãy xem xét một số ví dụ về cách sử dụng SELECT DISTINCT để xem nó hoạt động như thế nào.

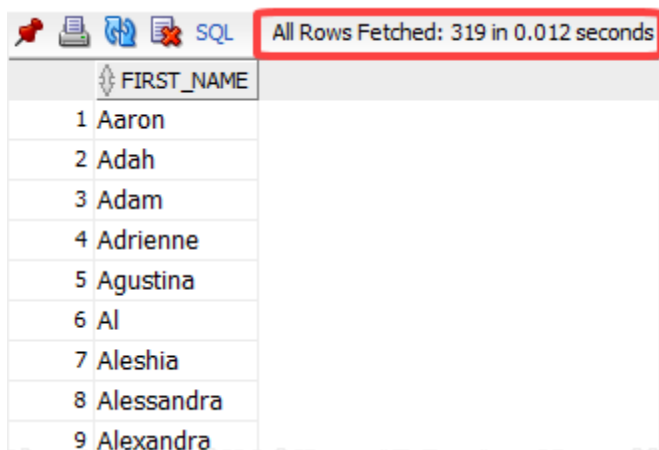
A) Ví dụ về một cột SELECT DISTINCT

CONTACTS
* CONTACT_ID
FIRST_NAME
LAST_NAME
EMAIL
PHONE
CUSTOMER_ID

Ví dụ sau truy xuất tất cả tên liên hệ:

```
SELECT
  first_name
FROM
  contacts
ORDER BY
  first_name;
```

Truy vấn trả về 319 hàng, cho biết contacts có 319 hàng.



FIRST_NAME
1 Aaron
2 Adah
3 Adam
4 Adrienne
5 Agustina
6 Al
7 Aleshia
8 Alessandra
9 Alexandra

Để có được tên liên hệ duy nhất, bạn thêm DISTINCT vào câu lệnh trên SELECT như sau:

```
SELECT DISTINCT
  first_name
FROM
  contacts
ORDER BY
  first_name;
```

Bây giờ, tập kết quả có 302 hàng, nghĩa là đã loại bỏ 17 hàng trùng lặp.



	FIRST_NAME
1	Aaron
2	Adah
3	Adam
4	Adrienne
5	Agustina
6	Al
7	Aleshia

B) Ví dụ về nhiều cột của Oracle SELECT DISTINCT

Xem bảng sau order_items:

ORDER_ITEMS
* ORDER_ID
* ITEM_ID
PRODUCT_ID
QUANTITY
UNIT_PRICE

```
SELECT
  DISTINCT product_id,
  quantity
FROM
  ORDER_ITEMS
ORDER BY
  product_id;
```

Sau đây minh họa kết quả:

PRODUCT_ID	QUANTITY
1	43
1	57
1	95
1	127
1	135
2	65
2	99
3	46
3	101
4	82

Trong ví dụ này, cả hai giá trị product_id và quantity đều được sử dụng để đánh giá tính duy nhất của các hàng.

C) Oracle CHỌN DISTINCT và NULL

LOCATIONS
* LOCATION_ID
ADDRESS
POSTAL_CODE
CITY
STATE
COUNTRY_ID

Câu lệnh sau lấy dữ liệu từ cột trạng thái có nhiều giá trị NULL:

```
SELECT
    DISTINCT state
FROM
    locations
ORDER BY
    state NULLS FIRST;
```

Kết quả:

STATE
(null)
BE
Bavaria
California
Distrito Federa
Geneve
Maharashtra
Manchester
New Jersey

WHERE

WHERE chỉ định điều kiện tìm kiếm cho các hàng được SELECT trả về. Sau đây minh họa cú pháp của WHERE:

```
SELECT
    select_list
FROM
    table_name
WHERE
    search_condition
ORDER BY
    sort_expression;
```

WHERE xuất hiện sau FROM nhưng trước ORDER BY. Từ khóa sau WHERE là **search_condition** xác định điều kiện mà các hàng trả về phải đáp ứng.

Ngoài **SELECT**, bạn có thể sử dụng WHERE trong câu lệnh DELETE hoặc UPDATE để chỉ định những hàng nào cần cập nhật hoặc xóa.

Ví dụ về Oracle WHERE

PRODUCTS
* PRODUCT_ID
PRODUCT_NAME
DESCRIPTION
STANDARD_COST
LIST_PRICE
CATEGORY_ID

A) Chọn các hàng bằng cách sử dụng toán tử đẳng thức đơn giản

Ví dụ sau chỉ trả về các sản phẩm có tên là 'Kingston':

```
SELECT
    product_name,
    description,
    list_price,
    category_id
FROM
    products
WHERE
    product_name = 'Kingston';
```

Hình ảnh sau đây minh họa kết quả:

PRODUCT_NAME	DESCRIPTION	LIST_PRICE	CATEGORY_ID
Kingston	Speed:DDR3-1333,Type:240-pin DIMM,CAS:9Module:4x16GBSize:64GB	671.38	5
Kingston	Speed:DDR3-1600,Type:240-pin DIMM,CAS:11Module:4x8GBSize:32GB	653.5	5
Kingston	Speed:DDR3-1600,Type:240-pin DIMM,CAS:11Module:4x16GBSize:64GB	644	5
Kingston	Speed:DDR4-2133,Type:288-pin DIMM,CAS:15Module:4x16GBSize:64GB	741.63	5

1. Đầu tiên, FROM chỉ định bảng để truy vấn dữ liệu.
2. Thứ hai, WHERE lọc các hàng dựa trên điều kiện, ví dụ: product_name = 'Kingston').
3. Thứ ba, SELECT đã chọn các cột cần được trả về.

B) Chọn các hàng bằng toán tử so sánh

Ngoài toán tử đẳng thức (=), Oracle còn cung cấp cho bạn nhiều toán tử so sánh:

Toán tử	Mô tả
=	Đẳng thức, so sánh bằng
!=,<>	Bất bình đẳng
>	Lớn hơn
<	Nhỏ hơn
>=	Lớn hơn hoặc bằng
<=	Nhỏ hơn hoặc bằng
IN	Bằng bất kỳ giá trị nào trong danh sách các giá trị

Toán tử	Mô tả
ANY/ SOME / ALL	So sánh một giá trị với một danh sách hoặc subquery . Nó phải được đặt trước bởi một toán tử khác như =, >, <.
NOT IN	Không bằng bất kỳ giá trị nào trong danh sách giá trị
[NOT] BETWEEN n and m	Tương đương với [Không] $\geq n$ và $\leq y$.
[NOT] EXISTS	Trả về true nếu subquery trả về ít nhất một hàng
IS [NOT] NULL	Kiểm tra NULL

Ví dụ: để lấy sản phẩm có giá lớn hơn 500, sử dụng câu lệnh sau:

```
SELECT
    product_name,
    list_price
FROM
    products
WHERE
    list_price > 500;
```

PRODUCT_NAME	LIST_PRICE
Gigabyte GA-Z270X-Gaming 9	503.98
Asus Rampage V Edition 10	519.99
Supermicro H8DG6-F	525.99
MSI X99A GODLIKE GAMING CARBON	549.59
Asus Z10PE-D8 WS	561.59
Asus RAMPAGE V EXTREME	572.96
Asus ROG MAXIMUS IX EXTREME	573.99
Asus X99-E-10G WS	649
Intel DP35DPM	789.79

C) Chọn các hàng thỏa mãn một số điều kiện

Để kết hợp các điều kiện, bạn có thể sử dụng toán tử logic AND, OR and NOT.

Ví dụ: để lấy tất cả các motherboards thuộc category id 1 và có giá lớn hơn 500, sử dụng câu lệnh sau:

```
SELECT
    product_name,
    list_price
FROM
```

```

products
WHERE
    list_price > 500
    AND category_id = 4;

```

PRODUCT_NAME	LIST_PRICE
Gigabyte GA-Z270X-Gaming 9	503.98
Asus Rampage V Edition 10	519.99
Supermicro H8DG6-F	525.99
MSI X99A GODLIKE GAMING CARBON	549.59
Intel Core i7-5930K	554.99
Asus Z10PE-D8 WS	561.59
Intel Xeon E5-1650 V3	564.89
Asus RAMPAGE V EXTREME	572.96
Asus ROG MAXIMUS IX EXTREME	573.99

D) Chọn các hàng có giá trị nằm giữa hai khoảng giá trị

Để tìm các hàng có giá trị nằm giữa hai khoảng giá trị, bạn sử dụng BETWEEN trong WHERE.

Ví dụ:

```

SELECT
    product_name,
    list_price
FROM
    products
WHERE
    list_price BETWEEN 650 AND 680
ORDER BY
    list_price;

```

Hình ảnh sau đây minh họa tập kết quả:

PRODUCT_NAME	LIST_PRICE
Kingston	653.5
Corsair Dominator Platinum	659.99
Intel Core i7-3930K	660
Kingston	671.38
G.Skill Ripjaws V Series	677.99
Intel Core i7-7820X	678.75

Lưu ý rằng các biểu thức sau là tương đương:

```
list_price BETWEEN 650 AND 680  
list_price >= 650 AND list_price <= 680
```

E) Chọn các hàng nằm trong danh sách các giá trị

Để truy vấn các hàng nằm trong danh sách giá trị, bạn sử dụng toán tử IN như sau:

```
SELECT  
    product_name,  
    category_id  
FROM  
    products  
WHERE  
    category_id IN(1, 4)  
ORDER BY  
    product_name;
```

Sau đây minh họa kết quả:

PRODUCT_NAME	CATEGORY_ID
AMD Opteron 6378	1
ASRock C2750D4I	4
ASRock E3C224D4M-16RE	4
ASRock EP2C602-4L/D16	4
ASRock EP2C612 WS	4
ASRock Fatal1ty X299 Professional Gaming i9	4
ASRock X299 Taichi	4
ASRock X99 Extreme11	4
ASRock Z270 SuperCarrier	4
Asus KGPE-D16	4

Cách diễn đạt:

```
category_id IN (1, 4)
```

Tương tự

```
category_id = 1 OR category_id = 4
```

F) Chọn các hàng chứa giá trị là một phần của chuỗi

Câu lệnh sau truy xuất một sản phẩm có tên bắt đầu bằng Asus:

```

SELECT
    product_name,
    list_price
FROM
    products
WHERE
    product_name LIKE 'Asus%'
ORDER BY
    list_price;

```

Trong ví dụ này, chúng tôi đã sử dụng toán tử LIKE để khớp các hàng dựa trên mẫu đã chỉ định.

AND

Giới thiệu về toán tử AND

Toán AND tử là toán tử logic kết hợp các biểu thức Boolean và trả về true nếu cả hai biểu thức đều đúng. Nếu một trong các biểu thức sai, AND sẽ trả về false.

Cú pháp của AND như sau:

expression_1 AND expression_2

Bảng sau minh họa kết quả khi bạn kết hợp giá trị đúng, sai và NULL bằng AND:

	TRUE	FALSE	NULL
TRUE	TRUE	FALSE	NULL
FALSE	FALSE	FALSE	FALSE
NULL	NULL	FALSE	NULL

Thông thường, chúng ta sử dụng AND trong WHERE của các câu lệnh SELECT, DELETE, và UPDATE để tạo điều kiện cho dữ liệu khớp. Ngoài ra, chúng ta sử dụng AND trong của mệnh đề **JOIN** để tạo thành điều kiện join.

Ví dụ về toán tử AND

ORDERS
* ORDER_ID
CUSTOMER_ID
STATUS
SALESMAN_ID
ORDER_DATE

A) Oracle AND để kết hợp hai ví dụ về biểu thức Boolean

Ví dụ sau tìm thấy đơn hàng của customer 2 có trạng thái pending:

```
SELECT
    order_id,
    customer_id,
    status,
    order_date
FROM
    orders
WHERE
    status = 'Pending'
    AND customer_id = 2
ORDER BY
    order_date;
```

Đây là kết quả:

ORDER_ID	CUSTOMER_ID	STATUS	ORDER_DATE
78	2	Pending	14-DEC-15
44	2	Pending	20-FEB-17

B) Oracle AND kết hợp nhiều hơn hai ví dụ về biểu thức Boolean

Bạn có thể sử dụng nhiều AND để kết hợp các biểu thức Boolean.

Ví dụ: câu lệnh sau truy xuất các đơn hàng đáp ứng tất cả các điều kiện sau:

- Đặt hàng vào năm 2017
- salesman là 60
- có trạng thái **shipped**.

```
SELECT
    order_id,
    customer_id,
    status,
    order_date
FROM
    orders
WHERE
    status = 'Shipped'
    AND salesman_id = 60
```

```

AND EXTRACT(YEAR FROM order_date) = 2017
ORDER BY
order_date;

```

ORDER_ID	CUSTOMER_ID	STATUS	ORDER_DATE
77	1	Shipped	02-JAN-17
99	49	Shipped	07-JAN-17
104	18	Shipped	01-FEB-17

C) Kết hợp AND và OR

```

SELECT
    order_id,
    customer_id,
    status,
    salesman_id,
    order_date
FROM
    orders
WHERE
    (
        status = 'Canceled'
        OR status = 'Pending'
    )
    AND customer_id = 44
ORDER BY
    order_date;

```

ORDER_ID	CUSTOMER_ID	STATUS	SALESMAN_ID	ORDER_DATE
10	44	Pending	(null)	24-JAN-17
69	44	Canceled	54	17-MAR-17

OR

Toán tử **OR** là toán tử logic kết hợp các biểu thức Boolean và trả về true nếu một trong các biểu thức là đúng.

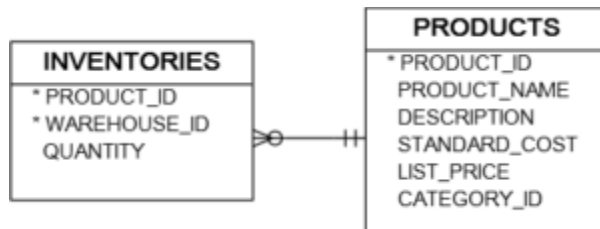
Sau đây minh họa cú pháp của OR:

```
expression_1 OR expression_2
```

	TRUE	FALSE	NULL
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	NULL
NULL	TRUE	NULL	NULL

FETCH

Một số RDBMS như MySQL và PostgreSQL có LIMIT cho phép bạn truy xuất một phần hàng được tạo bởi truy vấn .



Cơ sở dữ liệu Oracle không có mệnh đề **LIMIT**. Tuy nhiên, kể từ bản phát hành 12c, nó cung cấp một mệnh đề tương tự nhưng linh hoạt hơn, được gọi là mệnh đề giới hạn hàng.

Bằng cách sử dụng mệnh đề giới hạn hàng, bạn có thể viết lại truy vấn sử dụng mệnh đề LIMIT trên như sau:

```
SELECT
    product_name,
    quantity
FROM
    inventories
INNER JOIN products
    USING(product_id)
ORDER BY
    quantity DESC
FETCH NEXT 5 ROWS ONLY;
```

PRODUCT_NAME	QUANTITY
Kingston SA400S37/120G	353
Kingston SA400S37/120G	320
Zotac ZT-P10810D-10P	304
Gigabyte GV-N1070WF2OC-8GD	304
Kingston SA400S37/120G	294

Trong tuyên bố này, mệnh đề giới hạn hàng là:

```
FETCH NEXT 5 ROWS ONLY;
```

Tương tự như câu lệnh sử dụng LIMIT ở trên, mệnh đề giới hạn hàng trả về 5 sản phẩm hàng đầu có mức tồn kho cao nhất.

Cú pháp mệnh đề Oracle *FETCH*

```
[ OFFSET offset ROWS]
FETCH NEXT [ row_count | percent PERCENT ] ROWS [ ONLY | WITH TIES ]
```

Mệnh đề *OFFSET*

Mệnh đề *OFFSET* chỉ định số lượng hàng cần bỏ qua trước khi bắt đầu giới hạn hàng. *OFFSET* là tùy chọn. Nếu bạn bỏ qua nó thì giới hạn hàng bắt đầu từ hàng đầu tiên.

offset phải là một số hoặc một biểu thức có giá trị là một số. Tuân theo các quy tắc sau:

- Nếu offset là âm thì nó được coi là 0.
- Nếu giá trị chênh lệch là NULL hoặc lớn hơn số hàng được truy vấn trả về thì không có hàng nào được trả về.
- Nếu offset bao gồm một phân số thì phần phân số sẽ bị cắt bớt.

Mệnh đề *FETCH*

Mệnh đề *FETCH* chỉ định số lượng hàng hoặc tỷ lệ phần trăm của hàng cần trả về.

```
FETCH NEXT 1 ROWS
FETCH FIRST 1 ROW
```

ONLY | WITH TIES

ONLY: trả về chính xác số hàng hoặc tỷ lệ phần trăm của các hàng sau **FETCH NEXT (hoặc FIRST)**.

WITH TIES: trả về các hàng bổ sung các hàng có cùng giá trị khi được sắp xếp. Lưu ý rằng nếu sử dụng **WITH TIES**, phải sử dụng **ORDER BY**. Nếu không, truy vấn sẽ không trả về các hàng bổ sung.

Ví dụ về mệnh đề Oracle *FETCH*

A) Ví dụ về N hàng trên cùng

```
SELECT
    product_name,
    quantity
FROM
    inventories
INNER JOIN products
    USING(product_id)
ORDER BY
    quantity DESC
FETCH NEXT 10 ROWS ONLY;
```

PRODUCT_NAME	QUANTITY
Kingston SA400S37/120G	353
Kingston SA400S37/120G	320
Zotac ZT-P10810D-10P	304
Gigabyte GV-N1070WF2OC-8GD	304
Kingston SA400S37/120G	294
G.Skill Trident Z	282
G.Skill Ripjaws V Series	276
Corsair Vengeance LPX	275
G.Skill Trident Z	275
Zotac ZT-P10810C-10P	273

B) Ví dụ VỚI TIES

```
SELECT
    product_name,
    quantity
FROM
    inventories
INNER JOIN products
    USING(product_id)
ORDER BY
    quantity DESC
FETCH NEXT 10 ROWS WITH TIES;
```

	PRODUCT_NAME	QUANTITY
1	Kingston SA400S37/120G	353
2	Kingston SA400S37/120G	320
3	Zotac ZT-P10810D-10P	304
4	Gigabyte GV-N1070WF2OC-8GD	304
5	Kingston SA400S37/120G	294
6	G.Skill Trident Z	282
7	G.Skill Ripjaws V Series	276
8	Corsair Vengeance LPX	275
9	G.Skill Trident Z	275
10	Zotac ZT-P10810C-10P	273
11	MSI GeForce GTX 1080 TI ARMOR 11G OC	273
12	Zotac ZT-P10810G-10P	273

C) Ví dụ về giới hạn theo phần trăm của hàng

```
SELECT
    product_name,
    quantity
FROM
    inventories
INNER JOIN products
```

```

        USING(product_id)
ORDER BY
    quantity DESC
FETCH FIRST 5 PERCENT ROWS ONLY;

```

PRODUCT_NAME	QUANTITY
Kingston SA400S37/120G	353
Kingston SA400S37/120G	320
Gigabyte GV-N1070WF2OC-8GD	304
Zotac ZT-P10810D-10P	304
Kingston SA400S37/120G	294
G.Skill Trident Z	282
G.Skill Ripjaws V Series	276
Corsair Vengeance LPX	275
G.Skill Trident Z	275
Zotac ZT-P10810C-10P	273
MSI GeForce GTX 1080 TI ARMOR 11G OC	273
Zotac ZT-P10810G-10P	273
MSI GeForce GTX 1080 Ti GAMING X 11G	272

D) Ví dụ OFFSET

```

SELECT
    product_name,
    quantity
FROM
    inventories
INNER JOIN products
    USING(product_id)
ORDER BY
    quantity DESC
OFFSET 10 ROWS
FETCH NEXT 10 ROWS ONLY;

```

	PRODUCT_NAME	QUANTITY
1	Kingston SA400S37/120G	353
2	Kingston SA400S37/120G	320
3	Zotac ZT-P10810D-10P	304
4	Gigabyte GV-N1070WF2OC-8GD	304
5	Kingston SA400S37/120G	294
6	G.Skill Trident Z	282
7	G.Skill Ripjaws V Series	276
8	Corsair Vengeance LPX	275
9	G.Skill Trident Z	275
10	Zotac ZT-P10810C-10P	273
11	MSI GeForce GTX 1080 TI ARMOR 11G OC	273
12	Zotac ZT-P10810G-10P	273
13	MSI GeForce GTX 1080 Ti GAMING X 11G	272
14	G.Skill Ripjaws 4 Series	272
15	MSI X99A GODLIKE GAMING CARBON	271
16	Kingston SA400S37/120G	268
17	Samsung MZ-75E1T0B/AM	267
18	Samsung MZ-V6E500	267
19	Zotac ZT-P10810D-10P	266
20	Gigabyte GV-N1070WF2OC-8GD	266

OFFSET 10
ROWS

FETCH NEXT
10 ROWS
ONLY

IN

Toán tử Oracle IN xác định xem một giá trị có khớp với bất kỳ giá trị nào trong danh sách hoặc subquery

Subquery là một truy vấn được lồng trong một truy vấn khác.

Cú pháp của IN xác định xem một biểu thức có khớp với danh sách các giá trị hay không như sau:

```
expression [NOT] IN (v1,v2,...)
```

và cú pháp của một biểu thức khớp với subquery:

```
expression [NOT] IN (subquery)
```

A) Ví dụ về IN

```
SELECT
```

```
order_id,
```

```

customer_id,
status,
salesman_id
FROM
orders
WHERE
salesman_id IN (
54,
55,
56
)
ORDER BY
order_id;

```

ORDER_ID	CUSTOMER_ID	STATUS	SALESMAN_ID
1	4	Pending	56
5	5	Canceled	56
44	2	Pending	55
49	61	Shipped	55
50	62	Pending	55
54	65	Shipped	56
56	67	Canceled	55
61	2	Shipped	54
69	44	Canceled	54
71	46	Shipped	54

B) Oracle NOT IN example

```

SELECT
order_id,
customer_id,
status,
salesman_id
FROM
orders
WHERE
status NOT IN(
'Shipped',
'Canceled'
)
ORDER BY
order_id;

```

ORDER_ID	CUSTOMER_ID	STATUS	SALESMAN_ID
1	4	Pending	56
10	44	Pending	(null)
16	16	Pending	(null)
21	21	Pending	(null)
44	2	Pending	55
46	58	Pending	62
50	62	Pending	55
55	66	Pending	59
68	9	Pending	(null)

C) Oracle IN subquery example

```

SELECT
    employee_id,
    first_name,
    last_name
FROM
    employees
WHERE
    employee_id IN(
        SELECT
            DISTINCT salesman_id
        FROM
            orders
        WHERE
            status = 'Canceled'
    );
ORDER BY
    first Name;

```

EMPLOYEE_ID	FIRST_NAME	LAST_NAME
61	Daisy	Ortiz
56	Evie	Harrison
64	Florence	Freeman
62	Freya	Gomez
55	Grace	Ellis
60	Isabelle	Marshall
54	Lily	Fisher
57	Scarlett	Gibson

D) Oracle NOT IN subquery example

```

SELECT
    customer_id,
    name
FROM

```

```

        customers
WHERE
        customer_id NOT IN(
            SELECT
                customer_id
            FROM
                orders
        );

```

CUSTOMER_ID	NAME
35	Kimberly-Clark
36	Hartford Financial Services Group
38	Kraft Heinz
40	Fluor
72	Icahn Enterprises
74	Performance Food Group
76	DISH Network
77	FirstEnergy
80	AES
81	CarMax

E) Oracle IN vs. OR

```

SELECT
    customer_id,
    status,
    salesman_id
FROM
    orders
WHERE
    salesman_id IN(
        60,
        61,
        62
    )
ORDER BY
    customer_id;

```


CUSTOMER_ID	STATUS	SALESMAN_ID
1	Shipped	62
1	Shipped	62
1	Shipped	60
3	Shipped	62
4	Shipped	61
5	Pending	60
6	Shipped	61
7	Canceled	61
8	Canceled	61
16	Shipped	62

```

SELECT
    customer_id,
    status,
    salesman_id
FROM
    orders
WHERE
    salesman_id = 60
    OR salesman_id = 61
    OR salesman_id = 62
ORDER BY
    customer_id;

```

BETWEEN

Toán tử BETWEEN cho phép bạn chỉ định một phạm vi để kiểm tra. Khi bạn sử dụng BETWEEN để tạo điều kiện tìm kiếm cho các hàng được SELECT trả về, chỉ những hàng có giá trị nằm trong phạm vi đã chỉ định mới được trả về.

Sau đây minh họa cú pháp của BETWEEN:

```
expression [ NOT ] BETWEEN low AND high
```

Toán tử NOT BETWEEN phủ nhận kết quả của BETWEEN .

Toán BETWEEN thường được sử dụng trong WHERE của câu lệnh SELECT, DELETE, và UPDATE.

A) Ví dụ về **BETWEEN** sử dụng giá trị số

PRODUCTS
* PRODUCT_ID
PRODUCT_NAME
DESCRIPTION
STANDARD_COST
LIST_PRICE
CATEGORY_ID

```
SELECT
    product_name,
    standard_cost
FROM
    products
WHERE
    standard_cost BETWEEN 500 AND 600
ORDER BY
    standard_cost;
```

Trong ví dụ này, so sánh các giá trị trong standard_cost với phạm vi từ 500 đến 600.

PRODUCT_NAME	STANDARD_COST
Asus Z10PE-D8 WS	504.14
G.Skill Trident X	507.32
Intel Core i7-3930K	509.32
G.Skill Ripjaws V Series	510.93
Intel Core i7-7820X	511.1
G.Skill Ripjaws V Series	517.78
EVGA 06G-P4-4998-KR	521.03
Intel Core i7-4930K	527.69
Intel Xeon E5-2630 V4	528.95
G.Skill Trident Z	532.27
G.Skill Trident Z	533.21
Zotac ZT-Pl0810C-10P	535.03
Intel Xeon E5-1650 V4	535.47
Corsair Dominator Platinum	537.63
Corsair Dominator Platinum	538.55

Để truy vấn các sản phẩm không nằm trong khoảng từ 500 đến 600, bạn thêm NOT vào truy vấn trên như sau:

```

SELECT
    product_name,
    standard_cost
FROM
    products
WHERE
    standard_cost NOT BETWEEN 500 AND 600
ORDER BY
    product_name;

```

PRODUCT_NAME	STANDARD_COST
ADATA ASU800SS-128GT-C	37.78
ADATA ASU800SS-512GT-C	113.29
AMD 100-5056062	1343.84
AMD 100-505989	2128.67
AMD 100-506061	706.99
AMD FirePro S7000	936.42
AMD FirePro W9100	2483.38
AMD Opteron 6378	651.92
ASRock C2750D4I	339.55

B) BETWEEN với date

ORDERS
* ORDER_ID
CUSTOMER_ID
STATUS
SALESMAN_ID
ORDER_DATE

Câu lệnh sau đây trả về các đơn hàng được khách hàng đặt trong khoảng thời gian từ December 1, 2016, đến December 31, 2016:

```

SELECT
    order_id,
    customer_id,
    status,
    order_date
FROM
    orders
WHERE
    order_date BETWEEN DATE '2016-12-01' AND DATE '2016-12-31'
ORDER BY
    order_date;

```

Đây là kết quả:

ORDER_ID	CUSTOMER_ID	STATUS	ORDER_DATE
87	7	Canceled	01-DEC-16
85	4	Pending	01-DEC-16
83	16	Shipped	02-DEC-16
82	44	Shipped	03-DEC-16
81	49	Shipped	13-DEC-16
80	3	Shipped	13-DEC-16
79	2	Shipped	14-DEC-16
102	45	Shipped	20-DEC-16

LIKE

Ví dụ: có thể muốn tìm những contacts có last names bắt đầu bằng 'st' hoặc first names kết thúc bằng 'er'. Trong trường hợp này, bạn sử dụng toán tử LIKE.

Cú pháp của toán tử LIKE như sau:

```
expression [NOT] LIKE pattern [ ESCAPE escape_characters ]
```

1)expression

expression là tên cột hoặc biểu thức mà bạn muốn kiểm tra dựa trên pattern.

2)pattern

pattern là một chuỗi để tìm kiếm trong tệp expression. pattern bao gồm các ký tự đại diện sau:

- % (phần trăm) khớp với bất kỳ chuỗi nào có 0 hoặc nhiều ký tự.
- _ (gạch dưới) khớp với bất kỳ ký tự đơn nào.

3)escape_character

- escape_character là ký tự xuất hiện phía trước ký tự đại diện để chỉ định rằng ký tự đại diện không được hiểu là ký tự đại diện mà là ký tự thông thường.
- escape_character nếu được chỉ định, phải là một ký tự
- LIKE trả về true nếu expression khớp với pattern. Nếu không, nó trả về sai.
- NOT, nếu được chỉ định, sẽ phủ nhận kết quả của LIKE.

A) Ví dụ về ký tự đại diện %

```
SELECT
    first_name,
    last_name,
    phone
FROM
    contacts
WHERE
    last_name LIKE 'St%'
ORDER BY
    last_name;
```

FIRST_NAME	LAST_NAME	PHONE
Josie	Steele	+41 69 012 3581
Bill	Stein	+39 6 012 4501
Birgit	Stephenson	+1 608 123 4374
Herman	Stokes	+39 49 012 4777
Violeta	Stokes	+1 810 123 4212
Gonzalo	Stone	+1 301 123 4814
Flor	Stone	+1 317 123 4104

Để tìm số điện thoại của những liên hệ có last names kết thúc bằng chuỗi 'er', sử dụng câu lệnh sau:

```
SELECT
    first_name,
    last_name,
    phone
FROM
    contacts
WHERE
    last_name LIKE '%er'
ORDER BY
    last_name;
```

Đây là kết quả:

FIRST_NAME	LAST_NAME	PHONE
Shamika	Bauer	+91 11 012 4853
Stephaine	Booker	+39 55 012 4559
Charlene	Booker	+41 61 012 3537
Annice	Boyer	+1 518 123 4618
Shelia	Brewer	+49 89 012 4129
Annabelle	Butler	+91 80 012 3737
Nichol	Carter	+91 11 012 4813
Barbie	Carter	+41 5 012 3573
Sharee	Carver	+1 215 123 4738
Agustina	Conner	+1 612 123 4399
Daniel	Costner	+1 812 123 4153

Để thực hiện so khớp không phân biệt chữ hoa chữ thường, sử dụng một trong hai hàm LOWER() hoặc UPPER() như sau:

```
UPPER( last_name ) LIKE 'ST%'
LOWER(last_name LIKE 'st%'
```

B) _ ví dụ về ký tự đại diện

```
SELECT
    first_name,
    last_name,
    email,
    phone
FROM
    contacts
WHERE
    first_name LIKE 'Je_i'
ORDER BY
    first_name;
```

Đây là kết quả:

FIRST_NAME	LAST_NAME	EMAIL	PHONE
Jeni	Levy	jeni.levy@centene.com	+1 812 123 4129
Jeri	Randall	jeri.randall@nike.com	+49 90 012 4131

Mẫu này 'Je_i' khớp với bất kỳ chuỗi nào bắt đầu bằng 'Je', theo sau là một ký tự và theo sau là 'i' ví dụ: Jeri hoặc Jeni, nhưng không phải Jenni.

C) Ví dụ về ký tự đại diện hỗn hợp

```
SELECT
    first_name,
    last_name,
    email,
    phone
FROM
    contacts
WHERE
    first_name LIKE 'Je %';
```

FIRST_NAME	LAST_NAME	EMAIL	PHONE
Jeannie	Poole	jeannie.poole@aboutmcdonalds.com	+91 80 012 4637
Jeni	Levy	jeni.levy@centene.com	+1 812 123 4129
Jeri	Randall	jeri.randall@nike.com	+49 90 012 4131
Jerica	Brooks	jerica.brooks@northropgrumman.com	+91 11 012 4811
Jermaine	Cote	jermaine.cote@wfscorp.com	+49 91 012 4133
Jess	Nguyen	jess.nguyen@searsholdings.com	+39 2 012 4773
Jessika	Merritt	jessika.merritt@bnymellon.com	+1 612 123 4397

D) Ví dụ về mệnh đề ESCAPE

Mệnh đề ESCAPE cho phép bạn tìm các chuỗi bao gồm một hoặc nhiều ký tự đại diện.

Ví dụ: một bảng có thể bao gồm dữ liệu có ký tự phần trăm % chẳng hạn như giá trị chiết khấu và tỷ lệ khấu hao.

Để tìm kiếm chuỗi 25%, bạn sử dụng ESCAPE đề như sau:

```
LIKE '%25!%' ESCAPE '!'
```

Nếu bạn không sử dụng ESCAPE, Oracle sẽ trả về bất kỳ hàng nào có chuỗi 25.

```
SELECT
    product_id,
    discount_message
FROM
    discounts
WHERE
    discount_message LIKE '%25!%' ESCAPE '!';
```

Kết quả là như sau:

PRODUCT_ID	DISCOUNT_MESSAGE
	2 Buy 1 and Get 25% OFF on 2nd

IS NULL và IS NOT NULL

NULL đặc biệt ở chỗ nó không phải là một giá trị như số, chuỗi ký tự hoặc ngày giờ, do đó, bạn không thể so sánh nó với bất kỳ giá trị nào khác như zero (0) hoặc một chuỗi trống (""). Nói chung, NULL thậm chí không bằng NULL.

ORDERS
* ORDER_ID
CUSTOMER_ID
STATUS
SALESMAN_ID
ORDER_DATE

Cột salesman_id lưu trữ id nhân viên bán hàng

Câu lệnh sau SELECT cố gắng trả lại tất cả các đơn đặt hàng không có nhân viên bán hàng chịu trách nhiệm:

```
SELECT * FROM orders
WHERE salesman_id = NULL
ORDER BY order_date DESC;
```

Nó trả về một hàng trống.

Truy vấn sử dụng toán tử so sánh (=) để so sánh các giá trị từ salesman_id với NULL, điều này không chính xác.

Để kiểm tra xem một giá trị có NULL hay không, bạn nên sử dụng IS NULL như sau:

```
expression | column IS NULL
```

Toán tử IS NULL trả về true nếu biểu thức hoặc cột là NUL. Nếu không, nó trả về sai.

Truy vấn sau đây trả về tất cả các đơn đặt hàng không có nhân viên bán hàng chịu trách nhiệm:

```
SELECT * FROM orders
WHERE salesman_id IS NULL
ORDER BY order_date DESC;
```

Đây là đầu ra của truy vấn:

ORDER_ID	CUSTOMER_ID	STATUS	SALESMAN_ID	ORDER_DATE
14	48	Shipped	(null)	28-SEP-17
15	49	Shipped	(null)	27-SEP-17
17	17	Shipped	(null)	27-SEP-17
36	51	Shipped	(null)	05-SEP-17
29	44	Shipped	(null)	14-AUG-17
31	46	Canceled	(null)	12-AUG-17
30	45	Shipped	(null)	12-AUG-17
20	20	Shipped	(null)	27-MAY-17
21	21	Pending	(null)	27-MAY-17
3	5	Shipped	(null)	26-APR-17

Tương tự, Để phủ định **IS NULL**, bạn sử dụng **IS NOT NULL** toán tử như sau:

```
expression | column IS NOT NULL
```

Ví dụ:

```
SELECT * FROM orders
WHERE salesman_id IS NOT NULL
ORDER BY order_date DESC;
```

Section 4. Joining tables

Oracle Joins

Phép nối Oracle được sử dụng để kết hợp các cột từ hai hoặc nhiều bảng dựa trên giá trị của các cột liên quan. Các cột liên quan thường là (các) cột khóa chính của bảng đầu tiên và (các) cột khóa ngoại của bảng thứ hai.

Oracle hỗ trợ inner join, left join, right join, full outer join và cross join..

Lưu ý rằng bạn có thể join một bảng với chính bảng đó để truy vấn dữ liệu phân cấp bằng cách sử dụng inner join, left join, và right join.. Kiểu tham gia này được gọi là self-join.

Thiết lập bảng mẫu

Sẽ cần tạo hai bảng mới có cùng cấu trúc để thực hành:

```

CREATE TABLE palette_a (
  id INT PRIMARY KEY,
  color VARCHAR2 (100) NOT NULL
);

CREATE TABLE palette_b (
  id INT PRIMARY KEY,
  color VARCHAR2 (100) NOT NULL
);

INSERT INTO palette_a (id, color)
VALUES (1, 'Red');

INSERT INTO palette_a (id, color)
VALUES (2, 'Green');

INSERT INTO palette_a (id, color)
VALUES (3, 'Blue');

INSERT INTO palette_a (id, color)
VALUES (4, 'Purple');

-- insert data for the palette_b
INSERT INTO palette_b (id, color)
VALUES (1, 'Green');

INSERT INTO palette_b (id, color)
VALUES (2, 'Red');

INSERT INTO palette_b (id, color)
VALUES (3, 'Cyan');

INSERT INTO palette_b (id, color)
VALUES (4, 'Brown');

```

Các bảng có một số màu phổ biến như Red và Green. Hãy gọi bảng palette_a bên trái và bảng palette_b bên phải:

ID	COLOR	ID	COLOR
1	Red	1	Green
2	Green	2	Red
3	Blue	3	Cyan
4	Purple	4	Brown

Oracle inner join

Câu lệnh sau join bảng bên trái với bảng bên phải bằng cách sử dụng các giá trị trong cột color:

```
SELECT
  a.id id_a,
  a.color color_a,
  b.id id_b,
  b.color color_b
FROM
  palette_a a
INNER JOIN palette_b b ON a.color = b.color;
```

Tương tự như

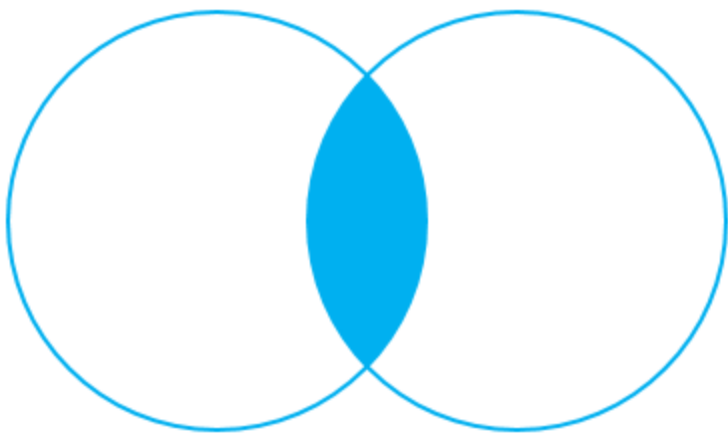
```
SELECT
  a.id id_a,
  a.color color_a,
  b.id id_b,
  b.color color_b
FROM
  palette_a a
WHERE a.color = b.color;
```

Đây là đầu ra:

ID_A	COLOR_A	ID_B	COLOR_B
2	Green	1	Green
1	Red	2	Red

Như có thể thấy rõ từ kết quả, phép nối bên trong trả về các hàng từ bảng bên trái khớp với các hàng từ bảng bên phải.

Sơ đồ Venn sau đây minh họa phép join bên trong khi kết hợp hai tập kết quả:



INNER JOIN

Oracle left join

Câu lệnh sau join bảng bên trái với bảng bên phải bằng cách sử dụng phép left join (hoặc left outer join):

```
SELECT
  a.id id_a,
  a.color color_a,
  b.id id_b,
  b.color color_b
FROM
  palette_a a
LEFT JOIN palette_b b ON a.color = b.color;
```

Tương tự

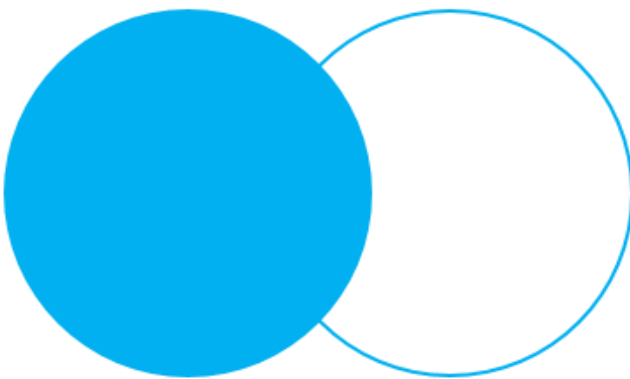
```
SELECT
  a.id id_a,
  a.color color_a,
  b.id id_b,
  b.color color_b
FROM
  palette_a a
WHERE a.color = b.color(+);
```

Đầu ra được hiển thị như sau:

ID_A	COLOR_A	ID_B	COLOR_B
2	Green	1	Green
1	Red	2	Red
3	Blue	(null)	(null)
4	Purple	(null)	(null)

left join trả về tất cả các hàng từ bảng bên trái với các hàng phù hợp nếu có ở bảng bên phải. Nếu không tìm thấy hàng nào phù hợp từ bảng bên phải, left join sẽ có giá trị null cho các cột của bảng bên phải:

Sơ đồ Venn sau đây minh họa phép nối trái:



LEFT OUTER JOIN

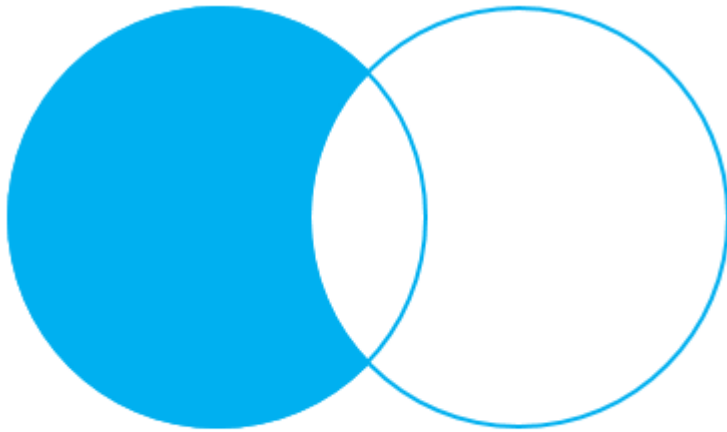
Đôi khi, bạn chỉ muốn lấy những hàng từ bảng bên trái không tồn tại ở bảng bên phải. Để đạt được điều này, bạn sử dụng left join và một WHERE để loại trừ các hàng khỏi bảng bên phải.

```
SELECT
    a.id id_a,
    a.color color_a,
    b.id id_b,
    b.color color_b
FROM
    palette_a a
LEFT JOIN palette_b b ON a.color = b.color
WHERE b.id IS NULL;
```

Đây là đầu ra:

ID_A	COLOR_A	ID_B	COLOR_B
3	Blue	(null)	(null)
4	Purple	(null)	(null)

Sơ đồ Venn sau đây minh họa left join với việc loại trừ các hàng khỏi bảng bên phải:



LEFT OUTER JOIN – only rows from the left table

Oracle right join

Right join hoặc right outer join là phiên bản đảo ngược của left join. Right join tạo một tập kết quả chứa tất cả các hàng từ bảng bên phải với các hàng khớp từ bảng bên trái. Nếu không có kết quả trùng khớp thì về trái sẽ có giá trị rỗng.

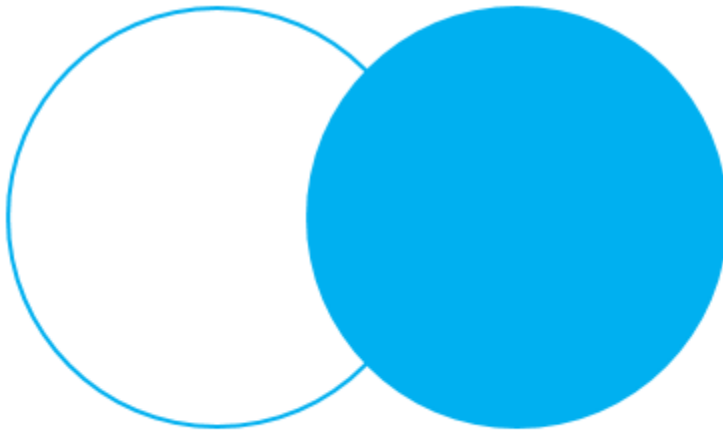
Ví dụ sau sử dụng phép nối phải để nối bảng bên trái với bảng bên phải:

```
SELECT
  a.id id_a,
  a.color color_a,
  b.id id_b,
  b.color color_b
FROM
  palette_a a
RIGHT JOIN palette_b b ON a.color = b.color;
```

Đây là đầu ra:

ID_A	COLOR_A	ID_B	COLOR_B
1	Red	2	Red
2	Green	1	Green
(null)	(null)	4	Brown
(null)	(null)	3	Cyan

Sơ đồ Venn sau đây minh họa phép nối đúng:



RIGHT OUTER JOIN

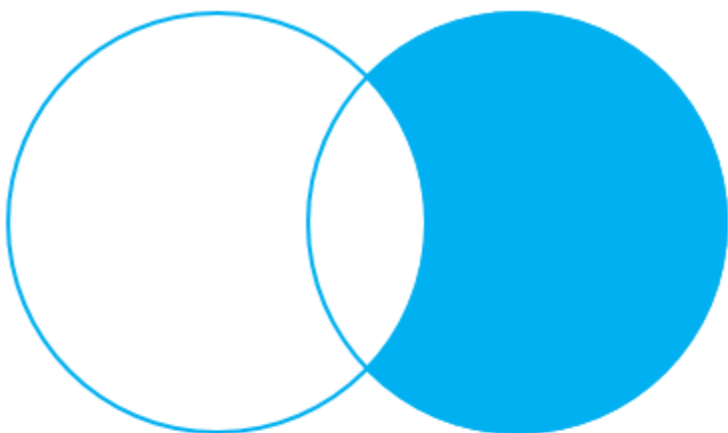
Tương tự, chỉ có thể lấy các hàng từ bảng bên phải chứ không lấy được các hàng từ bảng bên trái bằng cách thêm mệnh đề WHERE vào câu lệnh trên như trong truy vấn sau:

```
SELECT
  a.id id_a,
  a.color color_a,
  b.id id_b,
  b.color color_b
FROM
  palette_a a
RIGHT JOIN palette_b b ON a.color = b.color
WHERE a.id IS NULL;
```

Đây là đầu ra:

ID_A	COLOR_A	ID_B	COLOR_B
(null)	(null)	4	Brown
(null)	(null)	3	Cyan

Sơ đồ Venn sau đây minh họa phép nối phải với việc loại trừ các hàng khỏi bảng bên trái:



RIGHT OUTER JOIN – only rows from the right table

Oracle full outer join

Oracle full outer join hay full join của Oracle trả về một tập kết quả chứa tất cả các hàng từ cả hai bảng bên trái và bên phải, với các hàng khớp từ cả hai phía nếu có. Nếu không có sự trùng khớp thì bên còn thiếu sẽ có giá trị rỗng.

Ví dụ sau đây cho thấy phép nối ngoài đầy đủ của bảng bên trái và bên phải:

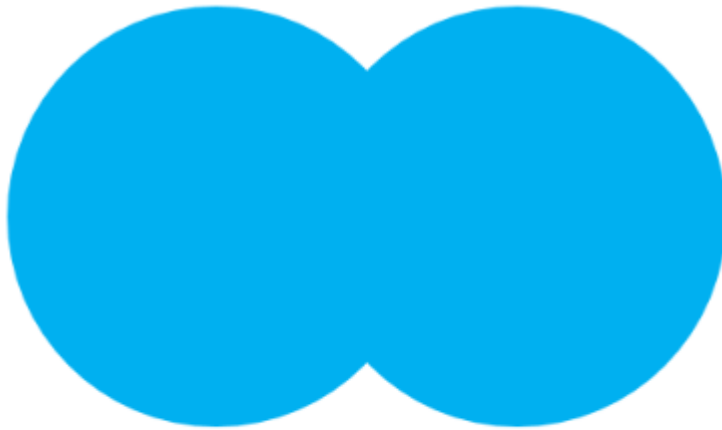
```
SELECT
  a.id id_a,
  a.color color_a,
  b.id id_b,
  b.color color_b
FROM
  palette_a a
FULL OUTER JOIN palette_b b ON a.color = b.color;
```

Hình ảnh sau đây minh họa tập kết quả của phép nối ngoài đầy đủ:

ID_A	COLOR_A	ID_B	COLOR_B
1	Red	2	Red
2	Green	1	Green
3	Blue	(null)	(null)
4	Purple	(null)	(null)
(null)	(null)	4	Brown
(null)	(null)	3	Cyan

Lưu ý rằng từ khóa OUTER là tùy chọn.

Sơ đồ Venn sau đây minh họa phép nối ngoài đầy đủ:



FULL OUTER JOIN

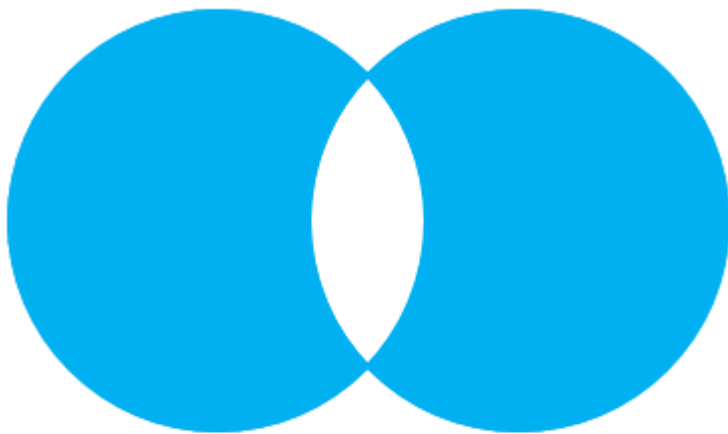
Để có được một tập hợp các hàng duy nhất từ 2 bảng, bạn thực hiện cùng một phép full join và sau đó loại trừ các hàng mà bạn không muốn ở cả hai bên bằng cách sử dụng mệnh đề WHERE như sau :

```
SELECT
    a.id id_a,
    a.color color_a,
    b.id id_b,
    b.color color_b
FROM
    palette_a a
FULL JOIN palette_b b ON a.color = b.color
WHERE a.id IS NULL OR b.id IS NULL;
```

Đây là kết quả:

ID_A	COLOR_A	ID_B	COLOR_B
(null)	(null)	3	Cyan
(null)	(null)	4	Brown
3	Blue	(null)	(null)
4	Purple	(null)	(null)

Sơ đồ Venn sau đây minh họa hoạt động trên:



FULL OUTER JOIN – only rows unique to both tables

INNER JOIN

Để truy vấn dữ liệu từ hai hoặc nhiều bảng có liên quan, bạn sử dụng INNER JOIN. Câu lệnh sau minh họa cách nối hai bảng T1 và T2.

```
SELECT
  *
FROM
  T1
INNER JOIN T2 ON join_predicate;
```

Oracle INNER JOIN với mệnh đề USING

Ngoài mệnh đề ON, có thể sử dụng mệnh đề USING để chỉ định cột nào cần kiểm tra sự bằng nhau khi nối các bảng.

Sau đây minh họa cú pháp của mệnh đề INNER JOIN với USING.

```
SELECT
  *
FROM
  T1
INNER JOIN T2 USING( c1, c2, ... );
```

Lưu ý rằng các cột được liệt kê trong USING như c1 và c2 phải có sẵn trong cả T1 và T2.

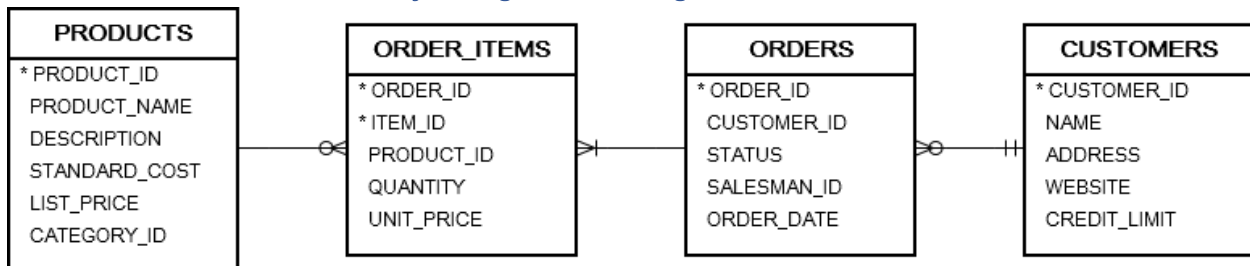
```

SELECT
  *
FROM
  orders
INNER JOIN order_items USING( order_id )
ORDER BY
  order_date DESC;

```

ORDER_ID	CUSTOMER_ID	STATUS	SALESMAN_ID	ORDER_DATE	ITEM_ID	PRODUCT_ID	QUANTITY	UNIT_PRICE
88	6	Shipped	61	01-NOV-17	2	11	106	2015.11
88	6	Shipped	61	01-NOV-17	1	278	139	677.99
94	1	Shipped	62	27-OCT-17	6	4	111	2699.99
94	1	Shipped	62	27-OCT-17	5	181	143	999.99
94	1	Shipped	62	27-OCT-17	8	258	73	57.98
94	1	Shipped	62	27-OCT-17	4	80	133	564.89
94	1	Shipped	62	27-OCT-17	7	172	37	358.49
94	1	Shipped	62	27-OCT-17	2	186	146	1449.98
94	1	Shipped	62	27-OCT-17	3	218	86	1388.89
94	1	Shipped	62	27-OCT-17	9	12	33	824.98
94	1	Shipped	62	27-OCT-17	1	255	38	90.99
1	4	Pending	56	15-OCT-17	10	64	147	525.99

Oracle INNER JOIN – joining nhiều bảng



inner join có thể nối nhiều hơn hai bảng. Trong thực tế, bạn nên giới hạn số lượng bảng được join để tránh vấn đề về hiệu suất. Câu lệnh sau đây chỉ ra cách nối ba bảng: orders, order_items, và customers.

```

SELECT
  name AS customer_name,
  order_id,
  order_date,
  item_id,
  quantity,
  unit_price
FROM
  orders
INNER JOIN order_items USING(order_id)
INNER JOIN customers USING(customer_id)

```

```
ORDER BY
  order_date DESC,
  order_id DESC,
  item_id ASC;
```

Đây là tập kết quả một phần:

CUSTOMER_NAME	ORDER_ID	ORDER_DATE	ITEM_ID	QUANTITY	UNIT_PRICE
Community Health Systems	88	01-NOV-17	1	139	677.99
Community Health Systems	88	01-NOV-17	2	106	2015.11
Raytheon	94	27-OCT-17	1	38	90.99
Raytheon	94	27-OCT-17	2	146	1449.98
Raytheon	94	27-OCT-17	3	86	1388.89
Raytheon	94	27-OCT-17	4	133	564.89
Raytheon	94	27-OCT-17	5	143	999.99
Raytheon	94	27-OCT-17	6	111	2699.99
Raytheon	94	27-OCT-17	7	37	358.49
Raytheon	94	27-OCT-17	8	73	57.98
Raytheon	94	27-OCT-17	9	33	824.98
AbbVie	1	15-OCT-17	1	116	469.99
AbbVie	1	15-OCT-17	2	77	686.99
AbbVie	1	15-OCT-17	3	52	525.99
AbbVie	1	15-OCT-17	4	87	1029.99
AbbVie	1	15-OCT-17	5	131	645.99
AbbVie	1	15-OCT-17	6	95	640.99
AbbVie	1	15-OCT-17	7	41	645.2
AbbVie	1	15-OCT-17	8	129	383.98

Câu truy vấn tương tự khi không sử dụng USING hoặc ON

```
SELECT
  name AS customer_name,
  order_id,
  order_date,
  item_id,
  quantity,
  unit_price
FROM
  orders, order_items, customers
WHERE
  orders.order_id = order_items.orders
  AND orders.customer_id = customers.customer_id
ORDER BY
  order_date DESC,
  order_id DESC,
```

```
item_id ASC;
```

LEFT JOIN

Câu lệnh sau minh họa cú pháp của mệnh đề LEFT JOIN khi nối hai bảng T1 và T2:

```
SELECT
    column_list
FROM
    T1
LEFT JOIN T2 ON
    join_predicate;
```

Oracle LEFT JOIN with USING clause

```
SELECT
    column_list
FROM
    T1
LEFT JOIN T2 USING(c1,c2,c3, ...);
```

Câu lệnh sau đây minh họa cách sử dụng câu lệnh LEFT JOIN với USING:

```
SELECT
    name,
    order_id,
    status,
    order_date
FROM
    customers
LEFT JOIN orders
    USING(customer_id)
ORDER BY
    name;
```

NAME	ORDER_ID	STATUS	ORDER_DATE
3M	(null)	(null)	(null)
ADP	(null)	(null)	(null)
AECOM	24	Shipped	07-SEP-16
AES	(null)	(null)	(null)
AIG	(null)	(null)	(null)
AT&T	(null)	(null)	(null)
AbbVie	2	Shipped	26-APR-15
AbbVie	1	Pending	15-OCT-17
AbbVie	85	Pending	01-DEC-16
AbbVie	63	Shipped	30-JUN-16
Abbott Laboratories	19	Shipped	27-MAY-16
Advance Auto Parts	(null)	(null)	(null)
Aetna	(null)	(null)	(null)
Aflac	75	Shipped	10-FEB-17

Oracle LEFT JOIN – join multiple tables

Câu lệnh sau sử dụng LEFT JOIN để nối ba bảng: orders, employees và customers:

```
SELECT
    order_id,
    name AS customer_name,
    status,
    first_name,
    last_name
FROM
    orders
LEFT JOIN employees ON
    employee_id = salesman_id
LEFT JOIN customers ON
    customers.customer_id = orders.customer_id
ORDER BY
    order_date DESC;
```

ORDER_ID	CUSTOMER_NAME	STATUS	FIRST_NAME	LAST_NAME
88	Community Health Systems	Shipped	Daisy	Ortiz
94	Raytheon	Shipped	Freya	Gomez
1	AbbVie	Pending	Evie	Harrison
14	Southern	Shipped	(null)	(null)
17	AutoNation	Shipped	(null)	(null)
15	NextEra Energy	Shipped	(null)	(null)
36	American Electric Power	Shipped	(null)	(null)
57	AutoZone	Shipped	Scarlett	Gibson
28	Community Health Systems	Canceled	Scarlett	Gibson
29	Jabil Circuit	Shipped	(null)	(null)
30	CenturyLink	Shipped	(null)	(null)
31	Supervalu	Canceled	(null)	(null)
60	Raytheon	Shipped	Freya	Gomez
20	Dollar General	Shipped	(null)	(null)

RIGHT JOIN

Câu lệnh sau minh họa cú pháp của mệnh đề RIGHT JOIN khi nối hai bảng T1 và T2:

```
SELECT
    column_list
FROM
    T1
RIGHT OUTER JOIN T2 ON
    join_predicate;
```

Ví dụ sau truy xuất tất cả nhân viên bán hàng và đơn đặt hàng của họ nếu có:

```
SELECT
    first_name,
    last_name,
    order_id,
    status
FROM
    orders
RIGHT JOIN employees ON
    employee_id = salesman_id
WHERE
    job_title = 'Sales Representative'
ORDER BY
    first_name,
    last_name;
```

FIRST_NAME	LAST_NAME	ORDER_ID	STATUS
Alice	Wells	(null)	(null)
Charlotte	Webb	(null)	(null)
Chloe	Cruz	51	Shipped
Chloe	Cruz	55	Pending
Chloe	Cruz	92	Shipped
Chloe	Cruz	95	Shipped
Chloe	Cruz	41	Shipped
Chloe	Cruz	4	Shipped
Chloe	Cruz	59	Shipped
Daisy	Ortiz	102	Shipped

Oracle RIGHT JOIN with USING clause

```
SELECT
    column_list
FROM
    T1
RIGHT OUTER JOIN T2 USING(c1,c2,c3);
```

Trong truy vấn này, các cột được liệt kê trong mệnh đề USING phải được trình bày trong cả hai bảng T1 và T2

Câu lệnh sau tương đương với câu trên:

```
SELECT
    column_list
FROM
    T1
RIGHT OUTER JOIN T2 ON
    T1.c1 = T2.c1
    AND T1.c2 = T2.c2
    AND T1.c3 = T2.c3;
```

Câu lệnh sau đây minh họa cách sử dụng mệnh đề RIGHT OUTER JOIN với USING:

```
SELECT
    name,
    order_id,
    status
FROM
    orders
RIGHT JOIN customers
    USING(customer_id)
ORDER BY
    name;
```


NAME	ORDER_ID	STATUS
3M	(null)	(null)
ADP	(null)	(null)
AECOM	24	Shipped
AES	(null)	(null)
AIG	(null)	(null)
AT&T	(null)	(null)
AbbVie	2	Shipped
AbbVie	1	Pending
AbbVie	85	Pending
AbbVie	63	Shipped
Abbott Laboratories	19	Shipped
Advance Auto Parts	(null)	(null)

FULL OUTER JOIN

Giả sử bạn có hai bảng T1 và T2. Phần sau đây minh họa **FULL OUTER JOIN** của hai bảng:

```
SELECT
    select_list
FROM
    T1
FULL OUTER JOIN T2 ON join_condition;
```

Đối với mỗi hàng trong bảng T1, phép **FULL OUTER JOIN** sẽ so sánh nó với mọi hàng trong bảng T2.

Sơ đồ Venn này minh họa phép nối ngoài đầy đủ của hai bảng:



FULL OUTER JOIN

Ví dụ về THAM GIA ĐẦY ĐỦ CỦA Oracle

Đầu tiên, tạo hai bảng members và projects. Giả sử mỗi thành viên có thể tham gia 0 hoặc một dự án và mỗi dự án có thể có 0 hoặc nhiều thành viên:

```
CREATE TABLE projects(  
    project_id INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,  
    project_name VARCHAR2(100) NOT NULL  
);  
  
CREATE TABLE members(  
    member_id INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,  
    member_name VARCHAR2(100) NOT NULL,  
    project_id INT,  
    FOREIGN KEY (project_id) REFERENCES projects(project_id)  
);
```

Thứ hai, thêm dữ liệu vào bảng projects và members:

```
INSERT INTO projects(project_name)  
VALUES('ERP');  
  
INSERT INTO projects(project_name)  
VALUES('Sales CRM');  
  
INSERT INTO members(member_name, project_id)  
VALUES('John Doe',1);  
  
INSERT INTO members(member_name, project_id)  
VALUES ('Jane Doe',1);  
  
INSERT INTO members(member_name, project_id)  
VALUES ('Jack Daniel',null);
```

Thứ ba, sử dụng full outer join để truy vấn dữ liệu từ members và projects:

```
SELECT  
    member_name,  
    project_name  
FROM  
    members m  
FULL OUTER JOIN projects p ON p.project_id = m.project_id  
ORDER BY  
    member_name;
```

Đây là đầu ra:

MEMBER_NAME	PROJECT_NAME
Jack Daniel	(null)
Jane Doe	ERP
John Doe	ERP
(null)	Sales CRM

Jack Daniel không tham gia dự án nào, Jane Doe và John Doe tham gia dự án ERP, còn dự án Sales CRM không có thành viên.

Để tìm dự án chưa có thành viên nào, bạn sử dụng truy vấn sau:

```
SELECT
    project_name,
    member_name
FROM
    members m
    FULL OUTER JOIN projects p
        ON p.project_id = m.project_id
WHERE
    member_name IS NULL
ORDER BY
    member_name;
```

PROJECT_NAME	MEMBER_NAME
Sales CRM	(null)

Self-join

Self join phép join một bảng với chính nó. Tính năng tự tham gia self join khi so sánh các hàng trong bảng hoặc truy vấn dữ liệu phân cấp.

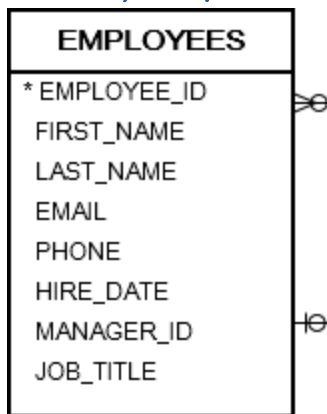
Tự nối sử dụng các join khác như inner join và left join. Ngoài ra, nó còn sử dụng table alias để gán các tên khác nhau cho bảng trong cùng một truy vấn.

Phần sau đây minh họa cách bảng được join với chính nó:

```
SELECT
    column_list
FROM
    T t1
INNER JOIN T t2 ON
    join_predicate;
```

Lưu ý rằng ngoài INNER JOIN, bạn có thể sử dụng LEFT JOIN trong câu lệnh trên.

A) Ví dụ về sử dụng Oracle self join để truy vấn dữ liệu phân cấp



Bảng `employees` lưu trữ các thông tin cá nhân như id, name, job title. Ngoài ra còn có `manager_id` cột lưu trữ các cấp quản lý giữa các nhân viên.

Để truy xuất dữ liệu nhân viên và người quản lý từ `employees`, bạn sử dụng tính năng self join như trong câu lệnh sau:

```
SELECT
    (e.first_name || ' ' || e.last_name) employee,
    (m.first_name || ' ' || m.last_name) manager,
    e.job_title
FROM
    employees e
LEFT JOIN employees m ON
    m.employee_id = e.manager_id
ORDER BY
    manager;
```

Hình ảnh sau đây cho thấy kết quả:

EMPLOYEE	MANAGER	JOB_TITLE
Tommy Bailey		President
Evie Harrison	Ava Sullivan	Sales Representative
Grace Ellis	Ava Sullivan	Sales Representative
Lily Fisher	Ava Sullivan	Sales Representative
Sophia Reynolds	Ava Sullivan	Sales Representative
Sophie Owens	Ava Sullivan	Sales Representative
Poppy Jordan	Ava Sullivan	Sales Representative
Louie Richardson	Blake Cooper	Programmer
Georgia Mills	Callum Jenkins	Shipping Clerk
Maisie Nichols	Callum Jenkins	Shipping Clerk
Eleanor Grant	Callum Jenkins	Shipping Clerk
Hannah Knight	Callum Jenkins	Shipping Clerk
Connor Haves	Callum Jenkins	Stock Clerk

B) Sử dụng Oracle self join để so sánh các hàng trong cùng một bảng

Câu lệnh sau đây tìm thấy tất cả nhân viên có cùng ngày thuê:

```
SELECT
    e1.hire_date,
    (e1.first_name || ' ' || e1.last_name) employee1,
    (e2.first_name || ' ' || e2.last_name) employee2
FROM
    employees e1
INNER JOIN employees e2 ON
    e1.employee_id > e2.employee_id
    AND e1.hire_date = e2.hire_date
ORDER BY
    e1.hire_date DESC,
    employee1,
    employee2;
```

HIRE_DATE	EMPLOYEE1	EMPLOYEE2
07-DEC-16	Rory Kelly	Elliot Brooks
28-SEP-16	Kai Long	Tyler Ramirez
20-AUG-16	Sophia Reynolds	Austin Flores
17-AUG-16	Amelie Hudson	Mohammad Peterson
21-JUN-16	Bella Stone	Ivy Burns
14-JUN-16	Jasmine Hunt	Seth Foster
07-JUN-16	Gracie Gardner	Harper Spencer
07-JUN-16	Rose Stephens	Gracie Gardner
07-JUN-16	Rose Stephens	Harper Spencer
07-JUN-16	Summer Payne	Gracie Gardner
07-JUN-16	Summer Payne	Harper Spencer
07-JUN-16	Summer Payne	Rose Stephens
21-APR-16	Elsie Henry	Matilda Stevens
10-APR-16	Reggie Simmons	Liam Henderson
24-MAR-16	Lucy Crawford	Sienna Simpson
24-MAR-16	Lucy Crawford	Sophie Owens
24-MAR-16	Rosie Morales	Lucy Crawford
24-MAR-16	Rosie Morales	Sienna Simpson
24-MAR-16	Rosie Morales	Sophie Owens
24-MAR-16	Sienna Simpson	Sophie Owens

Section 5. Grouping data

GROUP BY

Giới thiệu về Oracle GROUP BY

Mệnh đề GROUP BY được sử dụng trong SELECT để nhóm các hàng thành một tập hợp các hàng tóm tắt theo giá trị của cột hoặc biểu thức. Mệnh đề GROUP BY trả về một hàng cho mỗi nhóm.

Mệnh đề GROUP BY thường được sử dụng với các hàm tổng hợp như AVG(), COUNT(), MAX() và MIN(). SUM() Trong trường hợp này, hàm tổng hợp trả về thông tin tóm tắt cho mỗi nhóm. Ví dụ: cho các nhóm sản phẩm thuộc một số danh mục, hàm AVG() trả về giá trung bình của sản phẩm trong mỗi danh mục. Sau đây minh họa cú pháp của GROUP BY mệnh đề Oracle:

```
SELECT
    column_list
FROM
    T
GROUP BY c1,c2,c3;
```

Mệnh đề GROUP BY xuất hiện sau FROM. Trong trường hợp WHERE được sử dụng, GROUP BY phải được đặt sau WHERE như trong truy vấn sau:

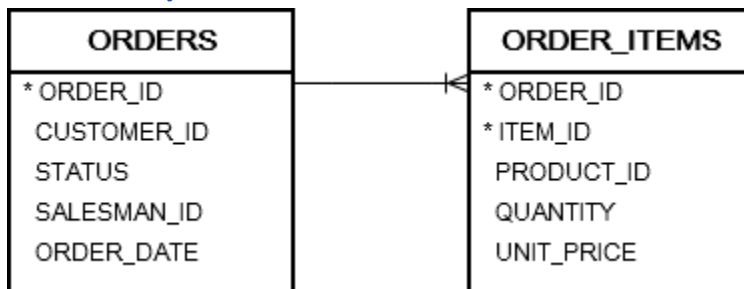
```
SELECT
    column_list
FROM
    T
WHERE
    condition
GROUP BY c1, c2, c3;
```

Mệnh đề GROUP BY nhóm các hàng theo giá trị trong các cột nhóm như c1, c2 và c3.

Nếu bạn muốn chỉ định nhiều cấp độ nhóm cần được tính toán cùng một lúc, bạn sử dụng ROLLUP cú pháp sau:

```
SELECT
    column_list
FROM
    T
GROUP BY
    ROLLUP(c1,c2,c3);
```

Ví dụ về Oracle GROUP BY



A) Oracle GROUP BY ví dụ cơ bản

Câu lệnh sau sử dụng GROUP BY để tìm các trạng thái đơn hàng duy nhất từ orders bảng:

```
SELECT
    status
FROM
    orders
GROUP BY
    status;
```

Ngôn ngữ mã: SQL (Ngôn ngữ truy vấn có cấu trúc) (sql)

STATUS
Shipped
Pending
Canceled

Câu lệnh này có tác dụng tương tự như câu lệnh sau sử dụng toán tử DISTINCT:

```
SELECT
    DISTINCT status
FROM
    orders;
```

B) Oracle GROUP BY với một ví dụ về hàm tổng hợp

Câu lệnh sau trả về số lượng đơn hàng của khách hàng:

```
SELECT
    customer_id,
    COUNT( order_id )
FROM
    orders
GROUP BY
    customer_id
ORDER BY
    customer_id;
```

CUSTOMER_ID	COUNT(ORDER_ID)
1	4
2	4
3	4
4	4
5	4
6	4
7	4
8	4
9	4

Trong ví dụ này, chúng tôi đã nhóm các đơn hàng theo khách hàng và sử dụng COUNT() để trả về số lượng đơn hàng cho mỗi nhóm.

Để có được dữ liệu ý nghĩa hơn, bạn có thể join bảng orders với customers như sau:

```
SELECT
    name,
```



```

COUNT( order_id )
FROM
  orders
INNER JOIN customers
  USING(customer_id)
GROUP BY
  name
ORDER BY
  name;

```

Đây là kết quả:

NAME	COUNT(ORDER_ID)
AECOM	1
AbbVie	4
Abbott Laboratories	1
Aflac	4
Alcoa	4
American Electric Power	1
AutoNation	4
AutoZone	1
Baker Hughes	1
Bank of New York Mellon Corp.	1

C) Oracle GROUP BY với một ví dụ về biểu thức

Ví dụ sau nhóm các đơn hàng theo năm và trả về số lượng đơn hàng mỗi năm.

```

SELECT
  EXTRACT(YEAR FROM order_date) YEAR,
  COUNT( order_id )
FROM
  orders
GROUP BY
  EXTRACT(YEAR FROM order_date)
ORDER BY
  YEAR;

```

Trong ví dụ này, đã sử dụng EXTRACT() để lấy thông tin năm từ ngày của đơn hàng.

Hình ảnh sau đây minh họa kết quả:

YEAR	COUNT(ORDER_ID)
2015	14
2016	50
2017	41

D) Oracle GROUP BY with WHERE clause example

Ví dụ này sử dụng GROUP BY có WHERE để trả về số lượng đơn hàng đã vận chuyển cho mỗi SELECT

```

name,
COUNT( order_id )
FROM orders
  INNER JOIN customers USING(customer_id)
WHERE
  status = 'Shipped'
GROUP BY
  name
ORDER BY
  name;
```

NAME	COUNT(ORDER_ID)
AECOM	1
AbbVie	2
Abbott Laboratories	1
Aflac	2
Alcoa	2
American Electric Power	1
AutoNation	3
AutoZone	1
Baker Hughes	1
Becton Dickinson	1
Bristol-Myers Squibb	1
Centene	2
CenturyLink	4
Community Health Systems	3
DTE Energy	1
Dollar General	1

Lưu ý rằng Oracle luôn đánh giá điều kiện ở mệnh đề WHERE trước mệnh đề GROUP BY.

E) Oracle GROUP BY with ROLLUP example

Câu lệnh sau tính toán số tiền bán hàng và nhóm chúng theo customer_id, status, và (customer_id, status):

```

SELECT
    customer_id,
    status,
    SUM( quantity * unit_price ) sales
FROM
    orders
INNER JOIN order_items
    USING(order_id)
GROUP BY
    ROLLUP(
        customer_id,
        status
    );

```

CUSTOMER_ID	STATUS	SALES
1	Pending	372002.04
1	Shipped	2406081.53
1	(null)	2778083.57
2	Pending	1632114.96
2	Shipped	655593.37
2	(null)	2287708.33
3	Pending	508078.3
3	Shipped	543168.6
3	(null)	1051246.9

HAVING

Giới thiệu về mệnh đề HAVING

Mệnh đề HAVING là mệnh đề tùy chọn của SELECT. Nó được sử dụng để lọc các group được mệnh đề trả về **GROUP BY**. Đây là lý do tại sao HAVING được sử dụng với GROUP BY.

Sau đây minh họa cú pháp của HAVING:

```

SELECT
    column_list
FROM
    T
GROUP BY
    c1
HAVING
    group_condition;

```

Trong câu này, HAVING xuất hiện ngay sau GROUP BY.

Nếu bạn sử dụng HAVING không có GROUP BY thì HAVING đó hoạt động giống như WHERE.

Lưu ý rằng mệnh HAVING để lọc các group trong khi WHERE mệnh để lọc các hàng. Đây là sự khác biệt chính giữa mệnh đề HAVING và WHERE.

Ví dụ về mệnh đề HAVING của Oracle

ORDER_ITEMS
* ORDER_ID
* ITEM_ID
PRODUCT_ID
QUANTITY
UNIT_PRICE

A) Ví dụ đơn giản về HAVING

Câu lệnh sau sử dụng GROUP BY để truy xuất các đơn hàng và giá trị của chúng từ order_items:

```
SELECT
    order_id,
    SUM( unit_price * quantity ) order_value
FROM
    order_items
GROUP BY
    order_id
ORDER BY
    order_value DESC;
```

Đây là kết quả:

ORDER_ID	ORDER_VALUE
70	1278962.17
46	1269323.77
78	1198331.59
1	1143716.87
68	1088670.12
27	1084871.49
32	1081679.88
92	1050939.97
59	1043144.72
76	953702.32
104	950118.04
60	926416.51

Để tìm những đơn hàng có giá trị lớn hơn 1 triệu, bạn thêm HAVING mệnh đề như sau:

```

SELECT
    order_id,
    SUM( unit_price * quantity ) order_value
FROM
    order_items
GROUP BY
    order_id
HAVING
    SUM( unit_price * quantity ) > 1000000
ORDER BY
    order_value DESC;

```

Kết quả là:

ORDER_ID	ORDER_VALUE
70	1278962.17
46	1269323.77
78	1198331.59
1	1143716.87
68	1088670.12
27	1084871.49
32	1081679.88
92	1050939.97
59	1043144.72

Trong ví dụ này:

- Đầu tiên, mệnh đề GROUP BY nhóm các thứ tự theo id của chúng và tính toán các giá trị thứ tự bằng SUM() hàm.
- Sau đó, mệnh đề HAVING lọc tất cả các đơn hàng có giá trị nhỏ hơn hoặc bằng 1,000,000.

B) HAVING với ví dụ về điều kiện phức tạp

Bạn có thể sử dụng điều kiện lọc phức tạp trong mệnh đề HAVING để để lọc các nhóm.

Ví dụ: câu lệnh sau tìm các đơn hàng có giá trị lớn hơn 500,000 và số lượng sản phẩm trong mỗi đơn hàng nằm trong khoảng từ 10 đến 12:

```

SELECT
    order_id,
    COUNT( item_id ) item_count,
    SUM( unit_price * quantity ) total
FROM

```

```

    order_items
GROUP BY
    order_id
HAVING
    SUM( unit_price * quantity ) > 500000 AND
    COUNT( item_id ) BETWEEN 10 AND 12
ORDER BY
    total DESC,
    item_count DESC;

```

Đây là kết quả:

ORDER_ID	ITEM_COUNT	TOTAL
46	11	1269323.77
76	10	953702.32
13	10	863963.89
91	11	852564.76
67	10	819908.58
58	12	804432.32
23	10	796470.07
87	11	776785.85
12	12	752198.64
75	11	698612.98
19	10	697288.63
40	11	676068.67
29	11	508588.59

Section 6. Intermediate Functions and Conditional Expressions

Functions

LENGTH

Hàm Oracle LENGTH() trả về số ký tự của một chuỗi đã chỉ định. Nó đo độ dài của chuỗi theo ký tự được xác định bởi bộ ký tự đầu vào.

```
LENGTH(string_expression);
```

Ví dụ

```
SELECT
  'Oracle LENGTH' string,
  LENGTH('Oracle LENGTH') Len
FROM
  dual;
```

	STRING	LEN
1	Oracle LENGTH	13

Câu lệnh sau đây sắp xếp các nhân viên theo độ dài tên của họ. Nó sử dụng LENGTH() trong ORDER BY:

```
SELECT
  first_name,
  LENGTH(first_name)
FROM
  employees
ORDER BY
  LENGTH(first_name) DESC;
```

	FIRST_NAME	LENGTH(FIRST_NAME)
1	Charlotte	9
2	Annabelle	9
3	Elizabeth	9
4	Frederick	9
5	Florence	8
6	Scarlett	8
7	Mohammad	8
8	Isabella	8
9	Isabelle	8
10	Abigail	7

LOWER & UPPER

LOWER

Hàm LOWER() chuyển đổi tất cả các chữ cái trong chuỗi thành chữ thường.

Cú pháp

```
LOWER(string)
```

Ví dụ:

Để tìm kiếm những liên hệ có last name là Hill, HILL, hoặc hill, bạn có thể sử dụng LOWER() hàm trong WHERE như sau:

```
SELECT
    first_name,
    last_name,
    email
FROM
    contacts
WHERE
    LOWER( last_name ) = 'hill';
```

Lưu ý rằng giá trị đầu vào phải ở dạng chữ thường để truy vấn hoạt động như mong đợi.

Đây là kết quả:

	FIRST_NAME	LAST_NAME	EMAIL
1	Caitlin	HILL	caitlin.hill@kraftheinzcompany.com
2	Roseline	Hill	roseline.hill@airproducts.com

UPPER

Hàm UPPER () chuyển đổi tất cả các chữ cái trong chuỗi thành chữ hoa.

Cú pháp

```
UPPER (string)
```

Ví dụ:

Để tìm kiếm những liên hệ có last name là Hill, HILL, hoặc hill, bạn có thể sử dụng UPPER () hàm trong WHERE như sau:

```
SELECT
    contact_id,
    first_name,
    last_name,
    email
FROM
```



```
contacts
WHERE
  UPPER( last_name ) = 'HILL';
```

Lưu ý rằng giá trị đầu vào phải ở dạng chữ hoa để truy vấn hoạt động như mong đợi.

Đây là kết quả:

	FIRST_NAME	LAST_NAME	EMAIL
1	Caitlin	HILL	caitlin.hill@kraftheinzcompany.com
2	Roseline	Hill	roseline.hill@airproducts.com

SUBSTR

Hàm Oracle SUBSTR() trích xuất một chuỗi con từ một chuỗi với nhiều tùy chọn linh hoạt khác nhau.

Cú pháp

```
SUBSTR( str, start_position [, substring_length, [, occurrence ]] );
```

Hàm SUBSTR() chấp nhận ba đối số:

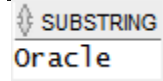
- **str** là chuỗi mà bạn muốn trích xuất chuỗi con. Kiểu dữ liệu của nó có thể là CHAR, VARCHAR2, NCHAR, NVARCHAR2
- **start_position** là một số nguyên xác định nơi chuỗi con bắt đầu. Phần sau đây giải thích tác dụng của start_position:
 - Nếu start_position là 0 thì phần đầu của chuỗi con sẽ ở ký tự đầu tiên của str.
 - Trường hợp start_position là dương thì SUBSTR() sẽ đếm từ đầu str để xác định ký tự đầu tiên của chuỗi con.
 - Nếu start_position là số âm thì SUBSTR() sẽ đếm ngược từ cuối str để tìm ký tự đầu tiên của chuỗi con.
- **substring_length** xác định số lượng ký tự trong chuỗi con. Nếu substring_length bị bỏ qua, SUBSTR() hàm sẽ trả về tất cả các ký tự bắt đầu từ start_position. Trong trường hợp a substring_length nhỏ hơn 1, SUBSTR() trả về null.

Giá trị trả về

Hàm SUBSTR() trả về chuỗi con str bắt đầu bằng start_position và có độ dài substring_length.

Ví dụ

```
SELECT
  SUBSTR( 'Oracle Substring', 1, 6 ) SUBSTRING
FROM
  dual;
```

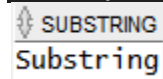


Câu lệnh sau trả về cùng chuỗi con như trên nhưng sử dụng start_position giá trị âm:

```
SELECT
  SUBSTR( 'Oracle Substring', - 16, 6 ) SUBSTRING
FROM
  dual;
```

Hãy xem xét ví dụ sau:

```
SELECT
  SUBSTR( 'Oracle Substring', 8 ) SUBSTRING
FROM
  dual;
```



Trong ví dụ này, chúng tôi đã bỏ qua đối số thứ ba (substring_length) do đó SUBSTR()hàm trả về tất cả các ký tự bắt đầu từ ký tự thứ 8 của chuỗi chính.

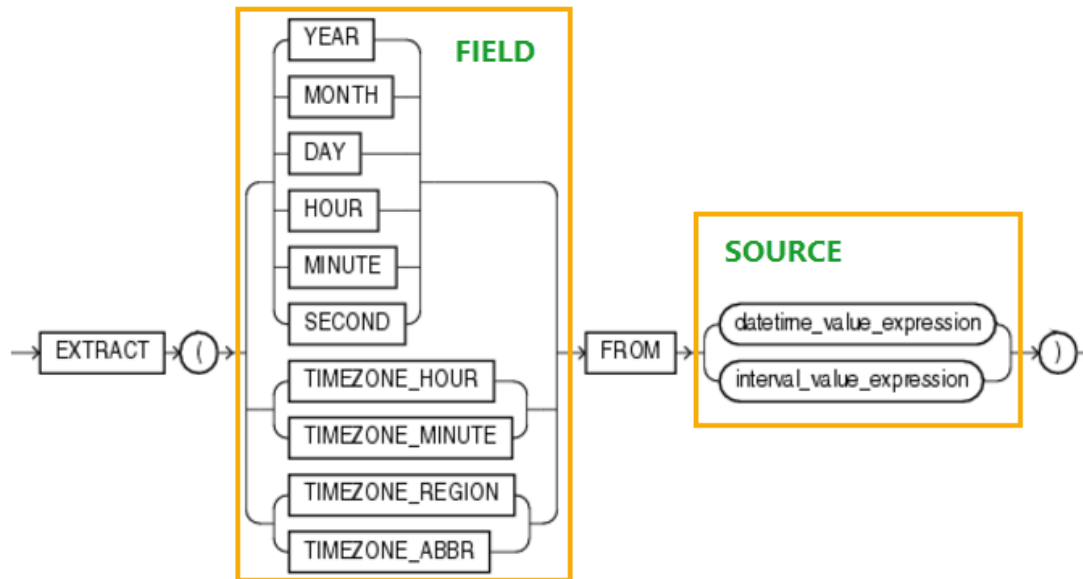
EXTRACT

EXTRACT() trích xuất một thành phần cụ thể (năm, tháng, ngày, giờ, phút, giây, v.v.) từ một giá trị ngày giờ hoặc khoảng thời gian.

Cú pháp

Sau đây minh họa cú pháp của hàm Oracle EXTRACT():

```
EXTRACT(field FROM source)
```



Bảng sau minh họa những trường bạn có thể trích xuất từ loại giá trị nào.

Value Type	Available Fields
DATE	YEAR, MONTH, DAY
INTERVAL YEAR TO MONTH	YEAR, MONTH
INTERVAL DAY TO SECOND	DAY, HOUR, MINUTE, SECOND
TIMESTAMP	YEAR, MONTH, DAY, HOUR, MINUTE, SECOND

A) Extracting fields from DATE values

```

SELECT
  EXTRACT( YEAR FROM TO_DATE( '31-Dec-1999 15:30:20 ', 'DD-Mon-YYYY HH24:MI:SS'
) ) YEAR
FROM
  DUAL;
  
```

B) Extracting fields from INTERVAL YEAR TO MONTH values

```

SELECT
  EXTRACT( YEAR FROM INTERVAL '5-2' YEAR TO MONTH )
FROM
  DUAL;
  
```

C) Extracting fields from TIMESTAMP values

```

SELECT
  EXTRACT( SECOND FROM TIMESTAMP '1999-12-31 23:59:59.10' )
FROM
  dual;
  
```

TO_CHAR

Hàm Oracle TO_CHAR() chuyển đổi một giá trị DATE hoặc INTERVAL thành một chuỗi theo định dạng ngày tháng được chỉ định.

Cú pháp

Sau đây minh họa cú pháp của TO_CHAR():

```
TO_CHAR(expr [, date_format] [, nlsparam]);
```

Oracle TO_CHAR() chấp nhận ba đối số:

- **expr** là một DATE hoặc một INTERVAL giá trị cần được chuyển đổi.
 - Kiểu dữ liệu của expr có thể là DATE, TIMESTAMP, TIMESTAMP WITH TIME ZONE hoặc TIMESTAMP WITH LOCAL TIME ZONE.
- **date_format** là một chuỗi xác định định dạng của chuỗi kết quả.
 - Đối date_format là tùy chọn. Nếu bạn bỏ qua nó, TO_CHAR() sẽ sử dụng định dạng ngày mặc định, định dạng dấu thời gian mặc định.
- **nlsparam** chỉ định ngôn ngữ cho tên và chữ viết tắt của ngày và tháng, ví dụ: Thứ Hai, Thứ Hai, Tháng Một, Tháng Một, v.v., trong chuỗi kết quả.
 - nlsparam có dạng sau: 'NLS_DATE_LANGUAGE = language'
 - Đối số này nlsparam cũng là tùy chọn.

Giá trị trả về

Hàm TO_CHAR() trả về một chuỗi được biểu thị bằng giá trị DATE hoặc INTERVAL theo định dạng đã chỉ định.

Ví dụ

Câu lệnh sau chuyển đổi ngày hệ thống hiện tại thành một chuỗi có định dạng YYYY-MM-DD:

```
SELECT  
  TO_CHAR( sysdate, 'YYYY-MM-DD' )  
FROM  
  dual;
```

Conditional Expressions

CASE WHEN

Biểu thức CASE cho phép bạn thêm logic if-else vào các câu lệnh SQL mà không cần phải gọi một procedure. Biểu thức CASE đánh giá một danh sách các điều kiện và trả về một trong nhiều kết quả có thể có.

Biểu thức CASE có hai định dạng: **Simple CASE expression** và **searched CASE expression**. Cả hai định dạng đều hỗ trợ một mệnh đề ELSE tùy chọn.

Simple CASE expression

Sau đây minh họa cú pháp của Simple CASE expression:

```
CASE e
  WHEN e1 THEN
    r1
  WHEN e2 THEN
    r2
  WHEN en THEN
    rn
  [ ELSE r_else ]
END
```

Ví dụ về biểu thức Simple CASE

PRODUCTS
* PRODUCT_ID
PRODUCT_NAME
DESCRIPTION
STANDARD_COST
LIST_PRICE
CATEGORY_ID

Truy vấn sau đây sử dụng CASE để tính discount cho từng danh mục sản phẩm, ví dụ: CPU 5%, card màn hình 10% và các danh mục sản phẩm khác 8%

```
SELECT
  product_name,
  list_price,
  CASE category_id
    WHEN 1
```

```

    THEN ROUND(list_price * 0.05,2) -- CPU
  WHEN 2
    THEN ROUND(List_price * 0.1,2) -- Video Card
    ELSE ROUND(list_price * 0.08,2) -- other categories
  END discount
FROM
  products
ORDER BY
  product_name;

```

PRODUCT_NAME	LIST_PRICE	DISCOUNT
ADATA ASU800SS-128GT-C	52.65	4.21
ADATA ASU800SS-512GT-C	136.69	10.94
AMD 100-5056062	1499.99	150
AMD 100-505989	2699.99	270
AMD 100-506061	999.99	100
AMD FirePro S7000	1218.5	121.85
AMD FirePro W9100	2998.89	299.89
AMD Opteron 6378	826.99	41.35
ASRock C2750D4I	401.98	32.16
ASRock E3C224D4M-16RE	499.99	40

Lưu ý rằng chúng ta đã sử dụng hàm ROUND() để làm tròn số chiết khấu đến hai chữ số thập phân.

Searched CASE expression

```

CASE
  WHEN e1 THEN r1
  [ WHEN e2 THEN r2]
  ...
  [ELSE
    r_else]
END

```

Ví dụ về Searched CASE expression

Ví dụ sau sử dụng để phân loại sản phẩm dựa trên price của chúng:

```

SELECT
  product_name,
  list_price,
  CASE
    WHEN list_price > 0 AND list_price < 600
      THEN 'Mass'
    WHEN list_price >= 600 AND list_price < 1000
      THEN 'Economy'
  
```

```

    WHEN list_price >= 1000 AND list_price < 2000
    THEN 'Luxury'
    ELSE
        'Grand Luxury'
    END product_group
FROM
    products
WHERE
    category_id = 1
ORDER BY
    product_name;

```

PRODUCT_NAME	LIST_PRICE	PRODUCT_GROUP
AMD Opteron 6378	826.99	Economy
Intel Core 2 Extreme QX6800	1003.98	Luxury
Intel Core 2 Extreme QX9775	892	Economy
Intel Core i7-3930K	660	Economy
Intel Core i7-3960X Extreme Edition	800.74	Economy
Intel Core i7-4770K	799	Economy
Intel Core i7-4790K	620.95	Economy
Intel Core i7-4930K	624.04	Economy
Intel Core i7-4960X Extreme Edition	1805.97	Luxury
Intel Core i7-5930K	554.99	Mass

DECODE

Sau đây minh họa cú pháp của hàm Oracle DECODE():

```
DECODE (e , s1, r1[, s2, r2], ..., [,sn,rn] [, d]);
```

- e là giá trị cần tìm kiếm. Hàm tự động chuyển e về kiểu dữ liệu s1 trước khi so sánh.
- s1, s2,... hoặc sn là biểu thức cần tìm kiếm. Lưu ý s2, s3,... sn được tự động chuyển sang kiểu dữ liệu s1 trước khi so sánh.
- r1, r2, ..., hoặc rn là biểu thức trả về khi e bằng s.
- d là biểu thức trả về khi e không bằng bất kỳ giá trị tìm kiếm nào s1, s2, .. sn.

Ví dụ:

```

SELECT
    product_name,
    list_price,
    DECODE(category_id,
        1, ROUND(list_price * 0.05, 2), -- CPU
        2, ROUND(list_price * 0.1, 2),  -- Video Card
        ROUND(list_price * 0.08, 2)    -- other categories

```

```
) AS discount  
FROM  
  products  
ORDER BY  
  product_name;
```

NVL

Hàm NVL() cho phép bạn thay thế giá trị null bằng một giá trị thay thế có ý nghĩa hơn trong kết quả của truy vấn.

Sau đây cho thấy cú pháp của NVL() hàm:

```
NVL(e1, e2)
```

Hàm NVL() chấp nhận hai đối số. Nếu e1 đánh giá là null thì NVL() hàm trả về e2. Nếu e1 đánh giá là khác null thì NVL() hàm trả về e1.

Ví dụ sau trả về 100 vì đối số đầu tiên không phải là null.

```
SELECT  
  NVL(100,200)  
FROM  
  dual;
```

Ví dụ sau trả về N/A vì đối số đầu tiên là null:

```
SELECT  
  NVL(NULL, 'N/A')  
FROM  
  dual;
```

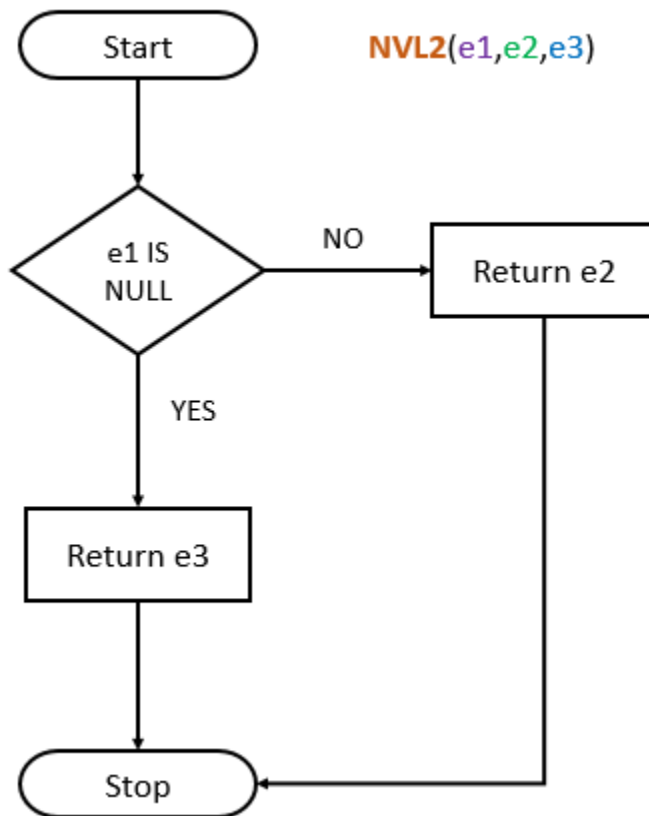
NVL2

Hàm Oracle NVL2() chấp nhận ba đối số. Nếu đối số đầu tiên không rỗng thì nó sẽ trả về đối số thứ hai. Trong trường hợp đối số thứ hai là null thì nó sẽ trả về đối số thứ ba.

Sau đây minh họa cú pháp của hàm Oracle NVL2():

```
NVL2(e1, e2, e3)
```


Sơ đồ sau đây minh họa cách hoạt động của hàm Oracle NVL2()



Câu lệnh sau trả về hai vì đối số đầu tiên là null.

```
SELECT
  NVL2(NULL, 1, 2) -- 2
FROM
  dual;
```

Section 7. Subquery

Subquery

Giới thiệu về Subquery

Subquery - Subquery là một câu lệnh SELECT được lồng bên trong một câu lệnh khác, chẳng hạn như SELECT, INSERT, UPDATE hoặc DELETE. Thông thường, bạn có thể sử dụng subquery ở bất kỳ nơi nào bạn sử dụng biểu thức.

PRODUCTS
* PRODUCT_ID
PRODUCT_NAME
DESCRIPTION
STANDARD_COST
LIST_PRICE
CATEGORY_ID

Truy vấn sau đây sử dụng MAX() để trả về giá niêm yết cao nhất từ productsbảng:

```
SELECT
    MAX( list_price )
FROM
    products;
```

MAX(LIST_PRICE)
8867.99

Để lấy thông tin chi tiết của các sản phẩm đắt nhất, bạn sử dụng câu lệnh ở trên trong truy vấn sau:

```
SELECT
    product_id,
    product_name,
    list_price
FROM
    products
WHERE
    list_price = 8867.99;
```

PRODUCT_ID	PRODUCT_NAME	LIST_PRICE
50	Intel SSDPECME040T401	8867.99

Cần thực hiện hai truy vấn riêng biệt để có được thông tin sản phẩm đắt nhất. Bằng cách sử dụng **subquery**, chúng ta có thể lồng truy vấn đầu tiên vào trong truy vấn thứ hai như trong câu lệnh sau:

```
SELECT
    product_id,
    product_name,
    list_price
FROM
    products
WHERE
    list_price = (
        SELECT
            MAX( list_price )
        FROM
            products
    );
```

Oracle đánh giá toàn bộ truy vấn trên theo hai bước:

- Đầu tiên, thực hiện subquery.
- Thứ hai, sử dụng kết quả của subquery trong truy vấn bên ngoài.

Ưu điểm của subquery Oracle

Đây là những ưu điểm chính của subquery:

- Cung cấp một cách khác để truy vấn dữ liệu yêu cầu join và unions phức tạp .
- Làm cho các truy vấn phức tạp dễ đọc hơn.
- Cho phép cấu trúc một truy vấn phức tạp theo cách có thể tách biệt từng phần.

Ví dụ về subquery của Oracle

A) Oracle subquery in the SELECT

Câu lệnh sau trả về tên product name, list price và giá niêm yết trung bình của sản phẩm theo danh mục của chúng:

```
SELECT
    product_name,
    list_price,
    ROUND(
        (
            SELECT
                AVG( list_price )
```

```

        FROM
            products p1
        WHERE
            p1. category_id = p2.category_id
    ),
    2
) avg_list_price
FROM
    products p2
ORDER BY
    product_name;

```

PRODUCT_NAME	LIST_PRICE	AVG_LIST_PRICE
ADATA ASU800SS-128GT-C	52.65	635.22
ADATA ASU800SS-512GT-C	136.69	635.22
AMD 100-5056062	1499.99	1406.1
AMD 100-505989	2699.99	1406.1
AMD 100-506061	999.99	1406.1
AMD FirePro S7000	1218.5	1406.1
AMD FirePro W9100	2998.89	1406.1
AMD Opteron 6378	826.99	1386.97
ASRock C2750D4I	401.98	402.29

B) Oracle subquery in the FROM

Cú pháp:

```
SELECT * FROM (subquery) [AS] inline_view;
```

Ví dụ: câu lệnh sau trả về 10 đơn hàng có giá trị cao nhất:

```

SELECT
    order_id,
    order_value
FROM
    (
        SELECT
            order_id,
            SUM( quantity * unit_price ) order_value
        FROM
            order_items
        GROUP BY
            order_id
        ORDER BY
            order_value DESC
    )

```

```
)  
FETCH FIRST 10 ROWS ONLY;
```

ORDER_ID	ORDER_VALUE
70	1278962.17
46	1269323.77
78	1198331.59
1	1143716.87
68	1088670.12
27	1084871.49
32	1081679.88
92	1050939.97
59	1043144.72
76	953702.32

Trong truy vấn này:

- Đầu tiên, subquery trả về danh sách order_id và order_value sắp xếp theo order_value từ giảm dần.
- Sau đó, truy vấn bên ngoài lấy 10 hàng đầu tiên từ đầu danh sách.

C) Ví dụ subquery với các toán tử so sánh

Các subquery sử dụng toán tử so sánh, ví dụ: >, >=, <, <=, <>, = thường bao gồm các hàm tổng hợp, bởi vì hàm tổng hợp trả về một giá trị duy nhất có thể được sử dụng để so sánh trong mệnh đề WHERE bên ngoài truy vấn.

Ví dụ: truy vấn sau đây sẽ tìm những sản phẩm có giá lớn hơn giá trung bình.

```
SELECT  
    product_id,  
    product_name,  
    list_price  
FROM  
    products  
WHERE  
    list_price > (  
        SELECT  
            AVG( list_price )  
        FROM  
            products  
    )  
ORDER BY  
    product_name;
```

PRODUCT_ID	PRODUCT_NAME	LIST_PRICE
161	AMD 100-5056062	1499.99
4	AMD 100-505989	2699.99
184	AMD 100-506061	999.99
48	AMD FirePro S7000	1218.5
142	AMD FirePro W9100	2998.89
181	ATI FirePro R5000	999.99
245	ATI FirePro S9050	1699
123	ATI FirePro S9150	3177.44
110	ATI FirePro W9000	3192.97

Truy vấn này hoạt động như sau:

- Đầu tiên, subquery trả về giá niêm yết trung bình của tất cả các sản phẩm.
- Thứ hai, truy vấn bên ngoài lấy các sản phẩm có giá lớn hơn giá trung bình mà subquery trả về.

D) Oracle subquery với IN và NOT IN

Subquery sử dụng IN toán tử thường trả về danh sách có 0 hoặc nhiều giá trị. Sau khi subquery trả về tập kết quả, truy vấn bên ngoài sẽ sử dụng chúng.

Ví dụ: truy vấn sau đây tìm nhân viên bán hàng có doanh số trên 100 nghìn vào năm 2017:

```
SELECT
    employee_id,
    first_name,
    last_name
FROM
    employees
WHERE
    employee_id IN(
        SELECT
            salesman_id
        FROM
            orders
        INNER JOIN order_items
            USING(order_id)
        WHERE
            status = 'Shipped'
        GROUP BY
            salesman_id,
            EXTRACT(
                YEAR
            FROM
```

```

        order_date
    )
HAVING
    SUM( quantity * unit_price ) >= 1000000
    AND EXTRACT(
        YEAR
    FROM
        order_date) = 2017
    AND salesman_id IS NOT NULL
)
ORDER BY
    first_name,
    last_name;

```

EMPLOYEE_ID	FIRST_NAME	LAST_NAME
62	Freya	Gomez
55	Grace	Ellis
60	Isabelle	Marshall

Oracle đánh giá truy vấn này theo hai bước:

- Đầu tiên, subquery trả về danh sách những người bán hàng có doanh số lớn hơn hoặc bằng 1 triệu.
- Thứ hai, truy vấn bên ngoài sử dụng danh sách id nhân viên bán hàng để truy vấn dữ liệu từ bảng employees.

EXISTS và NOT EXISTS.

Giới thiệu về toán tử EXISTS

Toán tử EXISTS là toán tử Boolean trả về giá trị đúng hoặc sai. Toán tử EXISTS thường được sử dụng với subquery để kiểm tra sự tồn tại của các hàng:

```

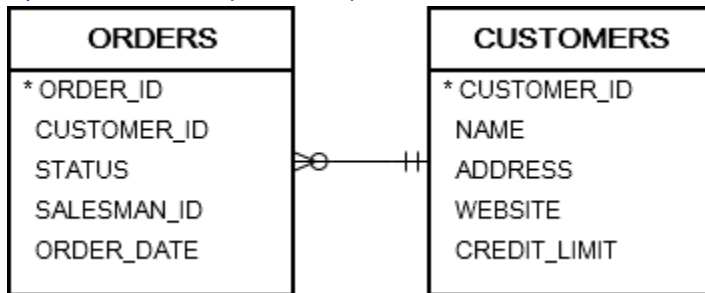
SELECT
    *
FROM
    table_name
WHERE
    EXISTS(subquery);

```

Toán tử EXISTS trả về true nếu subquery trả về bất kỳ hàng nào, nếu không, nó sẽ trả về false. Ngoài ra, EXISTS sẽ chấm dứt việc xử lý subquery sau khi subquery trả về hàng đầu tiên.

Ví dụ về EXISTS

A) EXISTS với ví dụ về câu lệnh SELECT



Ví dụ sau sử dụng EXISTS để tìm tất cả khách hàng có đơn hàng.

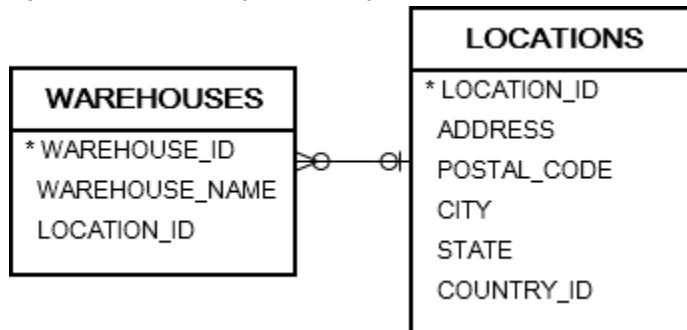
```
SELECT
    name
FROM
    customers c
WHERE
    EXISTS (
        SELECT
            1
        FROM
            orders
        WHERE
            customer_id = c.customer_id
    )
ORDER BY
    name;
```

NAME
AECOM
AbbVie
Abbott Laboratories
Aflac
Alcoa
American Electric Power
AutoNation
AutoZone
Baker Hughes
Bank of New York Mellon Corp.

Đối với mỗi khách hàng trong customers, subquery sẽ kiểm tra xem khách hàng đó có xuất hiện trên orders hay không.

Nếu có thì EXISTS trả về true và ngừng quét orders. Ngược lại, EXISTS trả về false nếu subquery không tìm thấy khách hàng trong orders.

B) EXISTS với ví dụ về câu lệnh UPDATE



```
UPDATE
  warehouses w
SET
  warehouse_name = warehouse_name || ', USA'
WHERE
  EXISTS (
    SELECT
      1
    FROM
      locations
    WHERE
      country_id = 'US'
      AND location_id = w.location_id
  );
```

EXISTS vs IN

Toán tử EXISTS dừng quét các hàng sau khi subquery trả về hàng đầu tiên vì nó có thể xác định kết quả trong khi toán tử IN phải quét tất cả các hàng được subquery trả về để kết luận kết quả.

Ngoài ra, mệnh đề IN không thể so sánh bất kỳ thứ gì với NULL giá trị, nhưng EXISTS có thể so sánh mọi thứ với giá trị NULL. Ví dụ: câu lệnh đầu tiên không trả về hàng nào trong khi câu lệnh thứ hai trả về tất cả các hàng trong bảng customers:

```
SELECT
  *
FROM
  customers
WHERE
  customer_id IN(NULL);
```

```

SELECT
    *
FROM
    customers
WHERE
    EXISTS (
        SELECT
            NULL
        FROM
            dual
    );

```

Thông thường, EXISTS nhanh hơn IN khi tập kết quả của subquery lớn. Ngược lại, IN nhanh hơn EXISTS khi tập kết quả của subquery nhỏ.

ANY

Giới thiệu về toán tử Oracle ANY

Toán tử ANY được sử dụng để so sánh một giá trị với danh sách các giá trị hoặc tập kết quả được subquery trả về. Phần sau đây minh họa cú pháp của ANY khi nó được sử dụng với danh sách hoặc subquery:

```

operator ANY ( v1, v2, v3)

operator ANY ( subquery)

```

Trong cú pháp này:

- Trước toán tử ANY phải có toán tử so sánh như =, !=, >, >=, <, <=.
- Danh sách hoặc subquery phải được bao quanh bởi dấu ngoặc đơn.

Khi bạn sử dụng toán tử ANY để so sánh một giá trị với một danh sách, Oracle sẽ mở rộng điều kiện ban đầu cho tất cả các thành phần của danh sách và sử dụng toán tử OR để kết hợp chúng như minh họa bên dưới:

```

SELECT
    *
FROM
    table_name
WHERE
    c > ANY (
        v1,
        v2,
        v3
    )

```

```
);
```

Oracle thực hiện chuyển đổi truy vấn trên thành truy vấn sau:

```
SELECT
    *
FROM
    table_name
WHERE
    c > v1
    OR c > v2
    OR c > v3;
```

Nếu bạn sử dụng toán tử ANY để so sánh một giá trị với tập kết quả được subquery trả về, Oracle sẽ sử dụng EXISTS để chuyển đổi truy vấn thành truy vấn tương đương mà không cần sử dụng ANY toán tử.

Ví dụ:

```
SELECT
    product_name,
    list_price
FROM
    products
WHERE
    list_price > ANY(
        SELECT
            list_price
        FROM
            products
        WHERE
            category_id = 1
    )
ORDER BY
    product_name;
```

Vì truy vấn sử dụng subquery với ANY nên Oracle đã thực hiện một phép biến đổi duy nhất như dưới đây:

```
SELECT
    product_name,
    list_price
FROM
    products p1
WHERE
```

```

    EXISTS(
        SELECT
            list_price
        FROM
            products p2
        WHERE
            category_id = 1
            AND p1.list_price > p2.list_price
    )
ORDER BY
    product_name;

```

1) col = ANY (list)

Biểu thức đánh giá là đúng nếu col khớp với một hoặc nhiều giá trị trong danh sách, ví dụ:

```

SELECT
    product_name,
    list_price
FROM
    products
WHERE
    list_price = ANY(
        2200,
        2259.99,
        2269.99
    )
    AND category_id = 1;

```

PRODUCT_NAME	LIST_PRICE
Intel Xeon E5-2695 V4	2269.99
Intel Xeon E5-2695 V2	2259.99
Intel Xeon E5-2643 V2 (OEM/Tray)	2200

2) col != BẤT KỲ(danh sách)

Biểu thức đánh giá là đúng nếu col không **khớp** với một hoặc nhiều giá trị trong danh sách.

```

SELECT
    product_name,
    list_price
FROM
    products
WHERE

```

```
list_price != ANY(
    2200,
    2259.99,
    2269.99
)
AND category_id = 1
ORDER BY
    list_price DESC;
```

PRODUCT_NAME	LIST_PRICE
Intel Xeon E5-2699 V3 (OEM/Tray)	3410.46
Intel Xeon E5-2697 V3	2774.98
Intel Xeon E5-2698 V3 (OEM/Tray)	2660.72
Intel Xeon E5-2697 V4	2554.99
Intel Xeon E5-2685 V3 (OEM/Tray)	2501.69
Intel Xeon E5-2695 V3 (OEM/Tray)	2431.95
Intel Xeon E5-2697 V2	2377.09
Intel Xeon E5-2695 V4	2269.99
Intel Xeon E5-2695 V2	2259.99
Intel Xeon E5-2643 V2 (OEM/Tray)	2200

ALL

Giới thiệu về toán tử ALL

Toán tử ALL được sử dụng để so sánh một giá trị với danh sách các giá trị hoặc tập kết quả được subquery trả về .

Phần sau đây trình bày cú pháp của toán tử ALL được sử dụng với danh sách hoặc subquery :

```
operator ALL ( v1, v2, v3)
```

```
operator ALL ( subquery)
```

Trong cú pháp này:

- Trước toán tử ALL phải có toán tử so sánh như =, !=, >, >=, <, <= và theo sau là danh sách hoặc subquery .
- Danh sách hoặc subquery phải được bao quanh bởi dấu ngoặc đơn.

Khi bạn sử dụng toán tử ALL để so sánh một giá trị với một danh sách, Oracle sẽ mở rộng điều kiện ban đầu cho tất cả các thành phần của danh sách và sử dụng toán AND tử để kết hợp chúng như minh họa bên dưới:

```
SELECT
```

```

*
FROM
    table_name
WHERE
    c > ALL (
        v1,
        v2,
        v3
    );

-- transform the ALL operator

SELECT
    *
FROM
    table_name
WHERE
    c > v1
    AND c > v2
    AND c > v3;

```

Nếu bạn sử dụng ALL để so sánh một giá trị với tập kết quả được subquery trả về , Oracle sẽ thực hiện chuyển đổi hai bước như dưới đây:

```

SELECT product_name,
       list_price
FROM products
WHERE list_price > ALL
      ( SELECT list_price
        FROM products
        WHERE category_id = 1 )
ORDER BY product_name;

-- 1st step: transformation that uses ANY

SELECT product_name,
       list_price
FROM products p1
WHERE NOT( p1.list_price <= ANY
          (SELECT list_price
           FROM products p2
           WHERE category_id = 1 ))
ORDER BY product_name;

-- 2nd step: transformation that eliminates ANY

```

```
SELECT product_name,
       list_price
FROM products p1
WHERE NOT EXISTS
      (SELECT p2.list_price
       FROM products p2
       WHERE p2.category_id = 1
            AND p2.list_price >= p1.list_price )
ORDER BY product_name;
```

Nếu subquery không trả về hàng nào thì điều kiện sau sẽ được đánh giá là đúng:

```
operator ALL (subquery)
```

Điều này có nghĩa là truy vấn sử dụng điều kiện trên trong mệnh đề WHERE sẽ trả về tất cả các hàng trong trường hợp subquery không trả về hàng nào.

```
SELECT
      *
FROM
      table_name
WHERE
      col operator ALL(subquery);
```

Ví dụ sau đây tìm giá niêm yết trung bình của sản phẩm trong từng danh mục sản phẩm:

```
SELECT
      ROUND( AVG( list_price ),2 ) avg_list_price
FROM
      products
GROUP BY
      category_id
ORDER BY
      avg_list price DESC;
```

AVG_LIST_PRICE
1406.1
1386.97
635.22
402.29

1) col > ALL (list)

Biểu thức đánh giá là đúng nếu giá trị col lớn hơn giá trị lớn nhất trong danh sách.

Ví dụ: truy vấn sau đây tìm tất cả các sản phẩm có giá niêm yết lớn hơn mức giá cao nhất trong bảng giá trung bình:

```
SELECT
    product_name,
    list_price
FROM
    products
WHERE
    list_price > ALL(
        SELECT
            AVG( list_price )
        FROM
            products
        GROUP BY
            category_id
    )
ORDER BY
    list_price ASC;
```

PRODUCT_NAME	LIST_PRICE
G.Skill Trident Z RGB	1418.99
G.Skill Trident Z	1431.99
PNY VCQK5200-PB	1449.98
Corsair Dominator Platinum	1449.99
Intel Xeon E5-2643 V3 (OEM/Tray)	1469.96
Intel Core i7-6950X	1499.89
Samsung MZ-75E4T0B	1499.99
AMD 100-5056062	1499.99
G.Skill TridentZ RGB	1504.99
Crucial	1620.99

2) col < ALL(list)

Biểu thức đánh giá là đúng nếu giá trị này col nhỏ hơn giá trị nhỏ nhất trong danh sách.

Ví dụ: truy vấn sau đây tìm tất cả các sản phẩm có giá niêm yết thấp hơn giá thấp nhất trong bảng giá trung bình:

```
SELECT
    product_name,
    list_price
FROM
    products
WHERE
    list_price < ALL(
```



```

SELECT
    AVG( list_price )
FROM
    products
GROUP BY
    category_id
)
ORDER BY
    list_price DESC;

```

PRODUCT_NAME	LIST_PRICE
ASRock C2750D4I	401.98
MSI X99A GODLIKE GAMING	399.99
Seagate ST10000DM0004	399.99
Gigabyte X299 AORUS Gaming 7	399.99
MSI X299 GAMING M7 ACK	397.42
Supermicro MBD-X10DRI-O	394.99
Asus MAXIMUS IX FORMULA	388.99
Asus X99-DELUXE II	383.98
ASRock Fatal1ty X299 Professional Gaming i9	382.98
ASRock EP2C612 WS	358.49

Section 8. Set Operators

UNION

Giới thiệu về toán tử Oracle

Toán tử UNION là toán tử tập hợp kết hợp các tập kết quả của hai hoặc nhiều SELECT câu lệnh thành một tập kết quả duy nhất.

Phần sau đây minh họa cú pháp của toán tử UNION kết hợp tập kết quả của hai truy vấn:

```

SELECT
    column_list_1
FROM
    T1
UNION
SELECT
    column_list_1
FROM
    T2;

```

Trong câu lệnh này, column_list_1 và column_list_2 phải có cùng số cột được trình bày theo cùng một thứ tự. Ngoài ra, kiểu dữ liệu của cột tương ứng.

Theo mặc định, UNION trả về các hàng duy nhất từ cả hai tập kết quả. Nếu bạn muốn giữ lại các hàng trùng lặp, bạn sử dụng rõ ràng UNION ALL như sau:

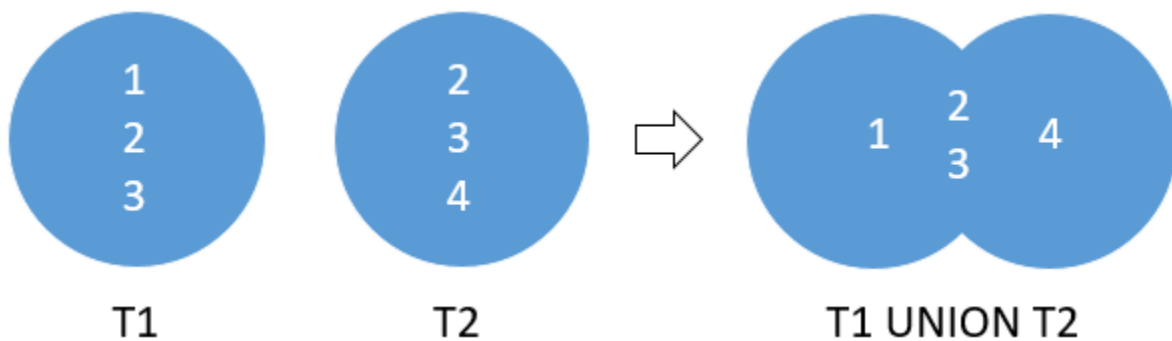
```
SELECT
    column_list
FROM
    T1
UNION ALL
SELECT
    column_list
FROM
    T2;
```

Minh họa Oracle UNION

Giả sử chúng ta có hai bảng T1 và T2:

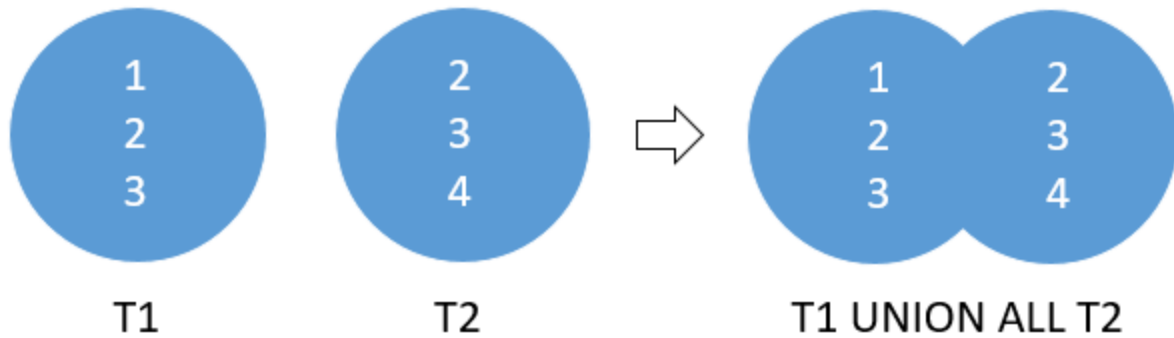
- T1 có 3 hàng 1, 2 và 3
- T2 còn có 3 hàng 2, 3 và 4

Hình ảnh sau đây minh họa UNION bảng T1 và T2:

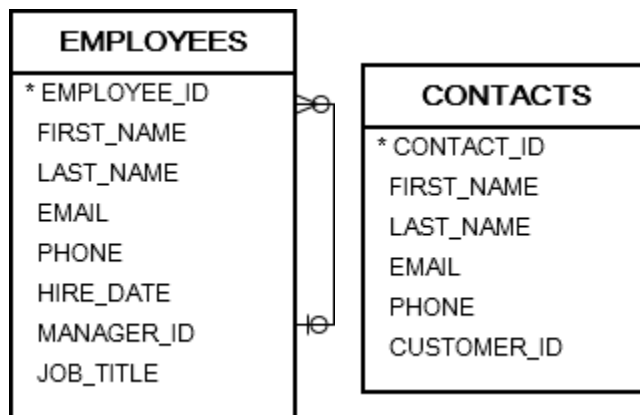


Việc UNION loại bỏ các hàng trùng lặp 2 và 3

Hình ảnh sau đây minh họa kết quả của UNION ALL bảng T1 và T2:



Như bạn có thể thấy, nó UNION ALL giữ lại hàng 2 và 3 trùng lặp.



A) Ví dụ về Oracle UNION

Giả sử bạn phải gửi email đến địa chỉ email của cả hai employees và contacts. Để thực hiện được điều này, trước tiên, bạn cần soạn danh sách địa chỉ email của nhân viên và những người liên hệ. Và sau đó gửi email vào danh sách.

Câu lệnh sau đây sử dụng UNION để xây dựng danh sách liên hệ từ bảng employees và contacts:

```
SELECT
    first_name,
    last_name,
    email,
    'contact'
FROM
    contacts
UNION SELECT
    first_name,
    last_name,
    email,
    'employee'
```

```
FROM
    employees;
```

Đây là kết quả:

FIRST_NAME	LAST_NAME	EMAIL	'CONTACT'
Aaron	Holder	aaron.holder@gilead.com	contact
Aaron	Patterson	aaron.patterson@example.com	employee
Abigail	Palmer	abigail.palmer@example.com	employee
Adah	Myers	adah.myers@dom.com	contact
Adam	Jacobs	adam.jacobs@univar.com	contact
Adrienne	Lang	adrienne.lang@qualcomm.com	contact
Agustina	Conner	agustina.conner@dollartree.com	contact
Al	Schultz	al.schultz@altria.com	contact
Albert	Watson	albert.watson@example.com	employee
Aleshia	Reese	aleshia.reese@adp.com	contact
Alessandra	Estrada	alessandra.estrada@ameriprise.com	contact

B) Ví dụ về Oracle UNION và ORDER BY

Để sắp xếp tập kết quả được toán tử trả về UNION, bạn thêm một ORDER BY vào SELECT cuối cùng như dưới đây:

```
SELECT
    first_name || ' ' || last_name name,
    email,
    'contact'
FROM
    contacts
UNION SELECT
    first_name || ' ' || last_name name,
    email,
    'employee'
FROM
    employees
ORDER BY
    name DESC;
```

Hình ảnh sau đây minh họa kết quả:

NAME	EMAIL	'CONTACT'
Aaron Holder	aaron.holder@gilead.com	contact
Aaron Patterson	aaron.patterson@example.com	employee
Abigail Palmer	abigail.palmer@example.com	employee
Adah Myers	adah.myers@dom.com	contact
Adam Jacobs	adam.jacobs@univar.com	contact
Adrienne Lang	adrienne.lang@qualcomm.com	contact
Agustina Conner	agustina.conner@dollartree.com	contact
Al Schultz	al.schultz@altria.com	contact
Albert Watson	albert.watson@example.com	employee
Aleshia Reese	aleshia.reese@adp.com	contact
Alessandra Estrada	alessandra.estrada@ameriprise.com	contact

C) UNION ALL Ví dụ về Oracle

Câu lệnh sau đây trả về họ duy nhất của nhân viên và người liên hệ:

```
SELECT
    last_name
FROM
    employees
UNION SELECT
    last_name
FROM
    contacts
ORDER BY
    last_name;
```

Truy vấn trả về 357 records.

LAST_NAME
Abbott
Alexander
Allison
Alston
Arnold
Atkinson
Avila
Bailey
Baldwin
Ball
Barnes
Barnett

Tuy nhiên, nếu bạn sử dụng UNION ALL thay vì UNION trong truy vấn như sau:

```
SELECT
    last_name
FROM
    employees
UNION ALL SELECT
    last_name
FROM
    contacts
ORDER BY
    last_name;
```

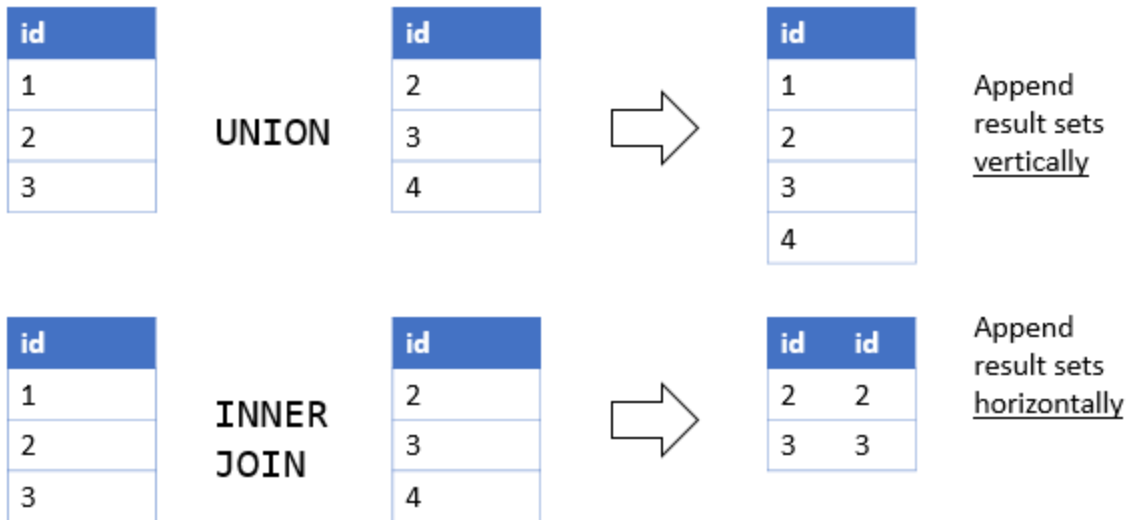
Truy vấn trả về 426 hàng. Ngoài ra, một số hàng bị trùng lặp, ví dụ: Atkinson, Barnett. Điều này là do UNION ALL toán tử không loại bỏ các hàng trùng lặp.

LAST_NAME
Abbott
Alexander
Allison
Alston
Arnold
Atkinson
Atkinson
Avila
Bailey
Baldwin
Ball
Barnes
Barnett
Barnett
Barrera

Oracle UNION vs. JOIN

Đặt một UNION tập kết quả lên trên một tập kết quả khác, nghĩa là nó join các tập kết quả theo chiều dọc. Tuy nhiên, một phép join chẳng hạn như **INNER JOIN** hoặc **LEFT JOIN** kết hợp các tập kết quả theo chiều ngang.

Hình ảnh sau đây minh họa sự khác biệt giữa kết hợp và tham gia:



INTERSECT

Giới thiệu về toán tử INTERSECT

Toán tử Oracle INTERSECT so sánh kết quả của hai truy vấn và trả về các hàng riêng biệt do cả hai truy vấn đưa ra.

Câu lệnh sau đây cho thấy cú pháp của toán tử INTERSECT:

```
SELECT
    column_list_1
FROM
    T1
INTERSECT
SELECT
    column_list_2
FROM
    T2;
```

Tương tự như UNION, bạn phải tuân theo các quy tắc sau khi sử dụng INTERSECT:

- Số lượng và thứ tự các cột trong hai truy vấn phải giống nhau.
- Kiểu dữ liệu của các cột tương ứng phải cùng nhóm kiểu dữ liệu như số hoặc ký tự.

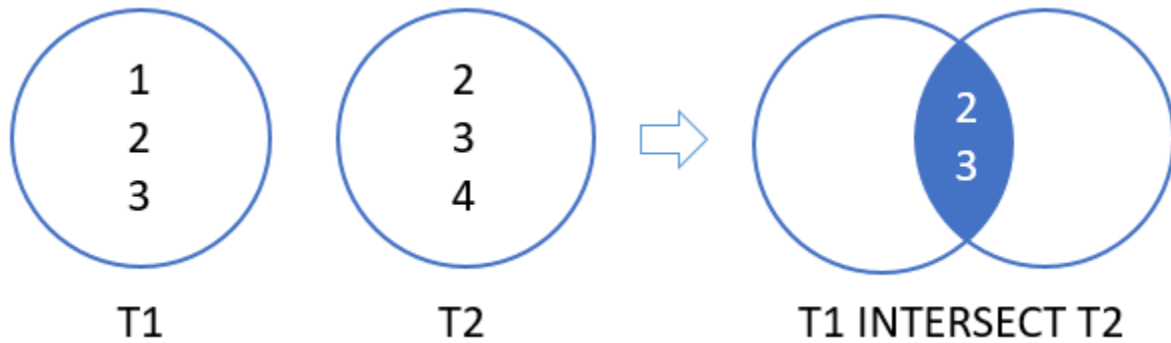
Hình minh họa INTERSECT của Oracle

Giả sử chúng ta có hai truy vấn trả về tập kết quả T1 và T2.

- Tập kết quả T1 gồm 1, 2, 3.
- Tập kết quả T2 gồm 2, 3, 4.

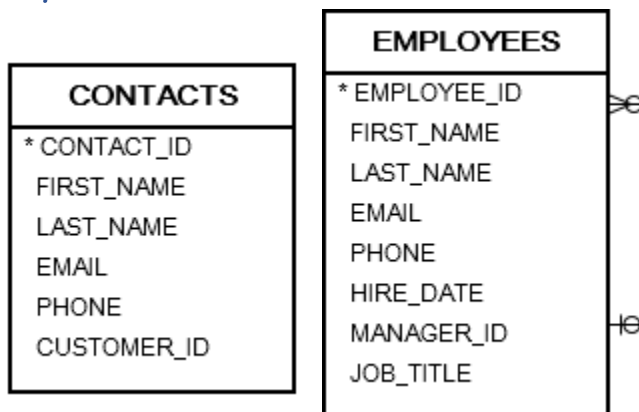
Giao điểm của kết quả T1 và T2 trả về 2 và 3. Bởi vì đây là những giá trị riêng biệt được cả hai truy vấn đưa ra.

Hình ảnh sau đây minh họa giao điểm của T1 và T2:



Hình minh họa cho thấy hàm INTERSECT trả về giao điểm của hai đường tròn (hoặc tập hợp).

Ví dụ về INTERSECT của Oracle



Câu lệnh sau đây sử dụng INTERSECT toán tử để lấy họ của mọi người trong cả hai bảng contacts và employees:

```
SELECT
    last_name
FROM
    contacts
INTERSECT
SELECT
    last_name
FROM
    employees
ORDER BY
    last_name;
```


LAST_NAME
Brooks
Bryant
Butler
Cole
Cruz
Ferguson
Flores
Ford
Grant
Hayes
Henderson
Henry
Jordan
Mason

MINUS

Giới thiệu về Toán tử MINUS

Toán tử Oracle MINUS hai truy vấn và trả về các hàng riêng biệt từ truy vấn đầu tiên mà truy vấn thứ hai không xuất ra. Nói cách khác, MINUS trừ một tập kết quả từ một tập kết quả khác.

Sau đây minh họa cú pháp của MINUS:

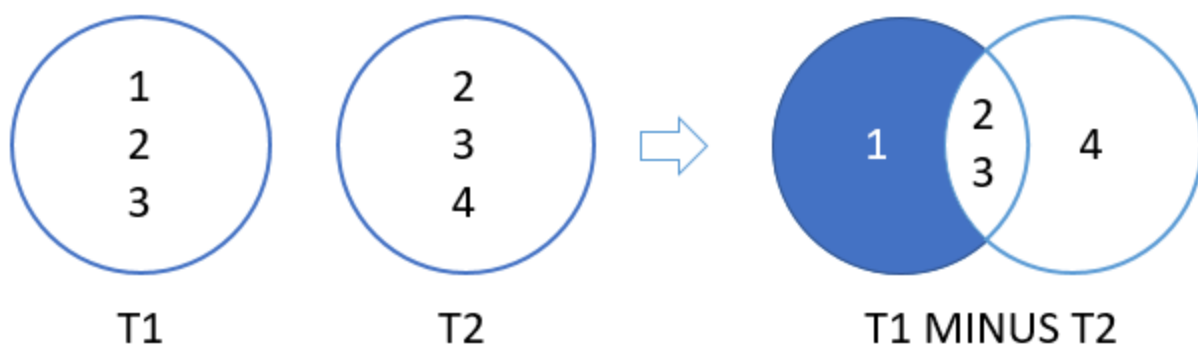
```
SELECT
    column_list_1
FROM
    T1
MINUS
SELECT
    column_list_2
FROM
    T2;
```

Tương tự như toán tử **UNION** and **INTERSECT**, các truy vấn trên phải tuân theo các quy tắc sau:

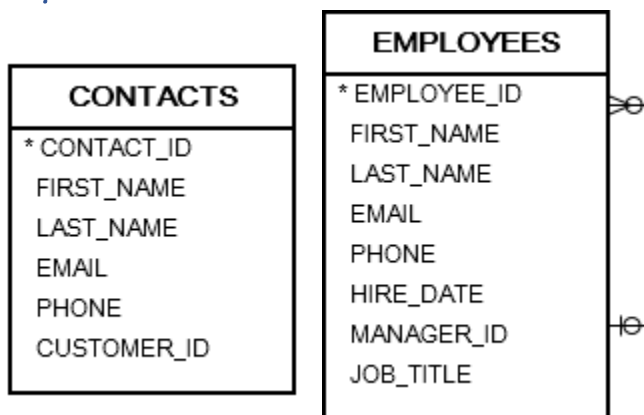
- Số lượng cột và thứ tự của chúng phải trùng nhau.
- Kiểu dữ liệu của các cột tương ứng phải cùng nhóm kiểu dữ liệu như số hoặc ký tự.

Giả sử truy vấn đầu tiên trả về tập kết quả T1 bao gồm 1, 2 và 3. Và truy vấn thứ hai trả về tập kết quả T2 bao gồm 2, 3 và 4.

Hình ảnh sau đây minh họa kết quả của MINUS của T1 và T2:



Ví dụ MINUS của Oracle



Câu lệnh sau đây trả về họ riêng biệt từ truy vấn ở bên trái của MINUS cũng không được tìm thấy trong truy vấn bên phải.

```
SELECT
    last_name
FROM
    contacts
MINUS
SELECT
    last_name
FROM
    employees
ORDER BY
    last_name;
```

Dưới đây là những last name được truy vấn đầu tiên trả về nhưng không tìm thấy trong tập kết quả của truy vấn thứ hai:

LAST_NAME
Abbott
Allison
Alston
Arnold
Atkinson
Avila
Baldwin
Ball
Barnett
Barrera

PRODUCTS	
* PRODUCT_ID	
PRODUCT_NAME	
DESCRIPTION	
STANDARD_COST	
LIST_PRICE	
CATEGORY_ID	

INVENTORIES
* PRODUCT_ID
* WAREHOUSE_ID
QUANTITY

Câu lệnh sau trả về danh sách product id từ bảng products nhưng không tồn tại trong bảng inventories:

```
SELECT
    product_id
FROM
    products
MINUS
SELECT
    product_id
FROM
    inventories;
```

PRODUCT_ID
1
10
16
28
45
48
49
51
52
53
55

Section 9. Modifying data

INSERT

Giới thiệu về câu lệnh INSERT của Oracle

Để chèn một hàng mới vào bảng, bạn sử dụng câu lệnh INSERT như sau:

```
INSERT INTO table_name (column_list)
VALUES( value_list);
```

Trong tuyên bố này:

- Đầu tiên, chỉ định tên của bảng mà bạn muốn chèn vào.
- Thứ hai, chỉ định danh sách tên cột được phân tách bằng dấu phẩy trong dấu ngoặc đơn.
- Thứ ba, chỉ định danh sách các giá trị được phân tách bằng dấu phẩy tương ứng với danh sách cột.

Nếu danh sách giá trị có cùng thứ tự với các cột trong bảng, bạn có thể bỏ qua danh sách cột mặc dù đây không được coi là một cách làm hay:

```
INSERT INTO table_name
VALUES (value_list);
```

Nếu bạn loại trừ một hoặc nhiều cột khỏi câu lệnh INSERT thì bạn phải chỉ định danh sách cột vì Oracle cần nó để khớp với các giá trị trong danh sách giá trị.

Cột mà bạn bỏ qua trong INSERT sẽ sử dụng giá trị mặc định nếu có hoặc giá trị NULL.

Ví dụ về câu lệnh INSERT của Oracle

Hãy tạo một bảng mới có tên discounts để chèn dữ liệu:

```
CREATE TABLE discounts (  
    discount_id NUMBER GENERATED BY DEFAULT AS IDENTITY,  
    discount_name VARCHAR2(255) NOT NULL,  
    amount NUMBER(3,1) NOT NULL,  
    start_date DATE NOT NULL,  
    expired_date DATE NOT NULL  
);
```

Trong table discounts, discount_id là cột identity có giá trị mặc định được hệ thống tự động tạo ra, do đó, không cần phải chỉ định cột discount_id trong câu lệnh INSERT.

Các cột khác discount_name, amount, start_date và expired_date là các cột NOT NULL nên bạn phải cung cấp giá trị cho chúng.

Câu lệnh sau chèn một hàng mới vào bảng discounts:

```
INSERT INTO discounts(discount_name, amount, start_date, expired_date)  
VALUES('Summer Promotion', 9.5, DATE '2017-05-01', DATE '2017-08-31');
```

Câu lệnh sau lấy dữ liệu từ discounts để xác minh việc insert:

```
SELECT  
    *  
FROM  
    discounts;
```

DISCOUNT_ID	DISCOUNT_NAME	AMOUNT	START_DATE	EXPIRED_DATE
1	Summer Promotion	9.5	01-MAY-17	31-AUG-17

Ví dụ sau chèn một hàng mới vào discounts:

```
INSERT INTO discounts(discount_name, amount, start_date, expired_date)  
VALUES('Winter Promotion 2017', 10.5, CURRENT_DATE, DATE '2017-12-31');
```

Trong ví dụ này, thay vì sử dụng hằng ngày, chúng tôi sử dụng kết quả của hàm CURRENT_DATE cho cột start_date.

```
SELECT  
    *  
FROM  
    discounts;
```

DISCOUNT_ID	DISCOUNT_NAME	AMOUNT	START_DATE	EXPIRED_DATE
1	Summer Promotion	9.5	01-MAY-17	31-AUG-17
2	Winter Promotion 2017	10.5	04-OCT-17	31-DEC-17

INSERT INTO SELECT

Tổng quan về câu lệnh INSERT INTO SELECT

Đôi khi muốn chọn dữ liệu từ một bảng và chèn nó vào một bảng khác. Để làm điều đó, sử dụng câu lệnh INSERT INTO SELECT như sau:

```
INSERT INTO target_table (col1, col2, col3)
SELECT col1,
       col2,
       col3
FROM source_table
WHERE condition;
```

Câu lệnh Oracle INSERT INTO SELECT yêu cầu kiểu dữ liệu của bảng nguồn và bảng đích phải khớp nhau.

A) Insert all sales data example

Hãy tạo một bảng được đặt tên sales.

```
CREATE TABLE sales (
  customer_id NUMBER,
  product_id  NUMBER,
  order_date  DATE NOT NULL,
  total       NUMBER(9,2) DEFAULT 0 NOT NULL,
  PRIMARY KEY(customer_id,
              product_id,
              order_date)
);
```

Câu lệnh sau chèn bản tóm tắt doanh số bán hàng từ bảng orders và order_items vào bảng sales:

```
INSERT INTO sales(customer_id, product_id, order_date, total)
SELECT customer_id,
       product_id,
       order_date,
       SUM(quantity * unit_price) amount
FROM orders
INNER JOIN order_items USING(order_id)
WHERE status = 'Shipped'
GROUP BY customer_id,
```

```
product_id,  
order_date;
```

Câu lệnh sau lấy dữ liệu từ sales để xác minh phần insert:

```
SELECT *  
FROM sales  
ORDER BY order_date DESC,  
total DESC;
```

CUSTOMER_ID	PRODUCT_ID	ORDER_DATE	TOTAL
6	11	01-NOV-17	213601.66
6	278	01-NOV-17	94240.61
1	4	27-OCT-17	299698.89
1	186	27-OCT-17	211697.08
1	181	27-OCT-17	142998.57
1	218	27-OCT-17	119444.54
1	80	27-OCT-17	75130.37
1	12	27-OCT-17	27224.34
1	172	27-OCT-17	13264.13
1	258	27-OCT-17	4232.54

B) Insert partial sales data example

Giả sử chỉ muốn sao chép dữ liệu tóm tắt doanh số năm 2017 sang một bảng mới. Để làm như vậy, trước tiên tạo một bảng mới có tên sales_2017 như sau:

```
CREATE TABLE sales_2017  
AS SELECT  
*  
FROM  
sales  
WHERE  
1 = 0;
```

Sử dụng INSERT INTO SELECT với WHERE mệnh đề để sao chép dữ liệu bán hàng năm 2017 vào sales_2017:

```
INSERT INTO sales_2017  
SELECT customer_id,  
product_id,  
order_date,  
SUM(quantity * unit_price) amount  
FROM orders  
INNER JOIN order_items USING(order_id)
```

```
WHERE status = 'Shipped' AND EXTRACT(year from order_date) = 2017
GROUP BY customer_id,
         product_id,
         order_date;
```

Trong ví dụ này, không chỉ định danh sách cột trong mệnh đề INSERT INTO vì kết quả của SELECT có các giá trị tương ứng với các cột của sales_2017.

```
Ngoài ra, chúng tôi đã bổ sung thêm điều kiện vào WHERE trong SELECT chỉ lấy dữ liệu bán hàng trong
SELECT *
FROM sales_2017
ORDER BY order_date DESC,
         total DESC;
```

CUSTOMER_ID	PRODUCT_ID	ORDER_DATE	TOTAL
6	11	01-NOV-17	213601.66
6	278	01-NOV-17	94240.61
1	4	27-OCT-17	299698.89
1	186	27-OCT-17	211697.08
1	181	27-OCT-17	142998.57
1	218	27-OCT-17	119444.54
1	80	27-OCT-17	75130.37
1	12	27-OCT-17	27224.34
1	172	27-OCT-17	13264.13
1	258	27-OCT-17	4232.54

C) Chèn một phần dữ liệu

```
INSERT INTO
  customer_lists(
    first_name,
    last_name,
    email,
    sent
  ) SELECT
    first_name,
    last_name,
    email,
    0
FROM
  contacts;
```


INSERT ALL

Chèn nhiều hàng vào một bảng

Để chèn nhiều hàng vào một bảng, sử dụng câu lệnh INSERT ALL:

```
INSERT ALL
  INTO table_name(col1,col2,col3) VALUES(val1,val2, val3)
  INTO table_name(col1,col2,col3) VALUES(val4,val5, val6)
  INTO table_name(col1,col2,col3) VALUES(val7,val8, val9)
Subquery;
```

Trong câu lệnh này, mỗi biểu thức giá trị val1, val2 hoặc val3 phải tham chiếu đến một cột được trả về bởi danh sách chọn của subquery .

Ví dụ sau đây minh họa cách chèn nhiều hàng vào một bảng.

Đầu tiên, tạo một bảng mới có tên fruits:

```
CREATE TABLE fruits (
  fruit_name VARCHAR2(100) PRIMARY KEY,
  color VARCHAR2(100) NOT NULL
);
```

Thứ hai, sử dụng câu lệnh Oracle INSERT ALL để chèn các hàng vào fruit sbảng:

```
INSERT ALL
  INTO fruits(fruit_name, color)
  VALUES ('Apple','Red')

  INTO fruits(fruit_name, color)
  VALUES ('Orange','Orange')

  INTO fruits(fruit_name, color)
  VALUES ('Banana','Yellow')
SELECT 1 FROM dual;
```

Thứ ba,xem dữ liệu sau khi thêm:

```
SELECT
  *
FROM
  fruits;
```

FRUIT_NAME	COLOR
Apple	Red
Orange	Orange
Banana	Yellow

Như bạn có thể thấy, ba hàng đã được chèn vào fruits thành công như mong đợi.

Chèn nhiều hàng vào nhiều bảng

Bên cạnh việc chèn nhiều hàng vào một bảng, bạn có thể sử dụng INSERT ALL để chèn nhiều hàng vào nhiều bảng như cú pháp sau:

```
INSERT ALL
  INTO table_name1(col1,col2,col3) VALUES(val1,val2, val3)
  INTO table_name2(col1,col2,col3) VALUES(val4,val5, val6)
  INTO table_name3(col1,col2,col3) VALUES(val7,val8, val9)
Subquery;
```

INSERT ALL có điều kiện

Câu lệnh INSERT ALL có điều kiện cho phép bạn chèn các hàng vào bảng dựa trên các điều kiện đã chỉ định.

Sau đây trình bày cú pháp của câu lệnh INSERT ALL có điều kiện:

```
INSERT [ ALL | FIRST ]
  WHEN condition1 THEN
    INTO table_1 (column_list ) VALUES (value_list)
  WHEN condition2 THEN
    INTO table_2(column_list ) VALUES (value_list)
  ELSE
    INTO table_3(column_list ) VALUES (value_list)
Subquery
```

Nếu bạn chỉ định từ khóa ALL thì Oracle sẽ đánh giá từng điều kiện trong WHEN. Nếu một điều kiện được đánh giá là đúng, Oracle sẽ thực thi mệnh đề INTO tương ứng.

Tuy nhiên, khi bạn chỉ định từ khóa FIRST, đối với mỗi hàng được truy vấn con trả về, Oracle sẽ đánh giá từng điều kiện trong mệnh đề WHEN từ trên xuống dưới. Nếu Oracle tìm thấy một điều kiện được đánh giá là đúng, nó sẽ thực thi mệnh đề INTO tương ứng và bỏ qua WHEN tiếp theo cho hàng đã cho.

Lưu ý rằng một câu lệnh insert nhiều bảng có điều kiện có thể có tới 127 WHEN.

Ví dụ về Oracle INSERT ALL có điều kiện

Các câu lệnh sau CREATE TABLE tạo ra ba bảng: small_orders, medium_orders, và big_orders có cùng cấu trúc:

```
CREATE TABLE small_orders (  
    order_id NUMBER(12) NOT NULL,  
    customer_id NUMBER(6) NOT NULL,  
    amount NUMBER(8,2)  
);  
  
CREATE TABLE medium_orders AS  
SELECT *  
FROM small_orders;  
  
CREATE TABLE big_orders AS  
SELECT *  
FROM small_orders;
```

Câu lệnh INSERT ALL chèn dữ liệu order vào ba bảng small_orders, medium_orders và big_orders dựa trên số lượng đơn hàng:

```
INSERT ALL  
    WHEN amount < 10000 THEN  
        INTO small_orders  
    WHEN amount >= 10000 AND amount <= 30000 THEN  
        INTO medium_orders  
    WHEN amount > 30000 THEN  
        INTO big_orders  
  
SELECT order_id,  
       customer_id,  
       (quantity * unit_price) amount  
FROM orders  
INNER JOIN order_items USING(order_id);
```

Bạn có thể đạt được kết quả tương tự bằng cách sử dụng ELSE thay cho phần insert vào big_orders như sau:

```
INSERT ALL  
    WHEN amount < 10000 THEN  
        INTO small_orders  
    WHEN amount >= 10000 AND amount <= 30000 THEN  
        INTO medium_orders  
    ELSE
```

```

        INTO big_orders
SELECT order_id,
       customer_id,
       (quantity * unit_price) amount
FROM orders
INNER JOIN order_items USING(order_id);

```

Ví dụ về Oracle INSERT FIRST có điều kiện

Hãy xem xét ví dụ sau:

```

INSERT FIRST
  WHEN amount > 30000 THEN
    INTO big_orders
  WHEN amount >= 10000 THEN
    INTO medium_orders
  WHEN amount > 0 THEN
    INTO small_orders
SELECT order_id,
       customer_id,
       (quantity * unit_price) amount
FROM orders
INNER JOIN order_items USING(order_id);

```

Các hạn chế của INSERT ALL

Câu lệnh insert nhiều bảng của Oracle phải tuân theo các hạn chế chính sau:

- Nó chỉ có thể được sử dụng để chèn dữ liệu vào bảng chứ không phải .
- Nó không thể được sử dụng để chèn dữ liệu vào các remote table.
- Số cột trong tất cả các INSERT INTO không được vượt quá 999.

UPDATE

Giới thiệu câu lệnh UPDATE

Để thay đổi các giá trị hiện có trong bảng, sử dụng câu lệnh UPDATE sau:

```

UPDATE
  table_name
SET
  column1 = value1,
  column2 = value2,
  column3 = value3,

```

```
...  
WHERE  
    condition;
```

Hãy xem xét UPDATE một cách chi tiết.

- Đầu tiên, bạn chỉ định tên của bảng mà bạn muốn cập nhật.
- Thứ hai, bạn chỉ định tên cột có giá trị cần cập nhật và giá trị mới. Nếu bạn cập nhật nhiều hơn hai cột, bạn sẽ phân tách từng biểu thức column = value bằng dấu phẩy. , , hoặc có thể là chữ hoặc truy vấn con trả về một giá trị duy nhất value1. Lưu ý rằng câu lệnh cho phép cập nhật bao nhiêu cột tùy thích.
- Thứ ba, WHERE xác định hàng nào của bảng sẽ được cập nhật. WHERE là tùy chọn. Nếu bạn bỏ qua nó, câu lệnh UPDATE sẽ cập nhật tất cả các hàng của bảng.

Ví dụ về Oracle UPDATE

Hãy tạo một bảng mới:

```
CREATE TABLE parts (  
    part_id NUMBER GENERATED BY DEFAULT AS IDENTITY,  
    part_name VARCHAR(50) NOT NULL,  
    lead_time NUMBER(2,0) NOT NULL,  
    cost NUMBER(9,2) NOT NULL,  
    status NUMBER(1,0) NOT NULL,  
    PRIMARY KEY (part_id)  
);
```

Thứ hai, INSERT các câu lệnh sau thêm dữ liệu mẫu vào parts:

```
INSERT INTO parts (part_name,lead_time,cost,status) VALUES ('sed  
dictum',5,134,0);  
INSERT INTO parts (part_name,lead_time,cost,status) VALUES ('tristique  
neque',3,62,1);  
INSERT INTO parts (part_name,lead_time,cost,status) VALUES ('dolor  
quam,',16,82,1);  
INSERT INTO parts (part_name,lead_time,cost,status) VALUES ('nec,  
diam.',41,10,1);  
INSERT INTO parts (part_name,lead_time,cost,status) VALUES ('vitae  
erat',22,116,0);  
INSERT INTO parts (part_name,lead_time,cost,status) VALUES ('parturient  
montes,',32,169,1);  
INSERT INTO parts (part_name,lead_time,cost,status) VALUES ('metus. In',45,88,1);  
INSERT INTO parts (part_name,lead_time,cost,status) VALUES ('at,  
velit.',31,182,0);
```

```

INSERT INTO parts (part_name,lead_time,cost,status) VALUES ('nonummy
ultricies',7,146,0);
INSERT INTO parts (part_name,lead_time,cost,status) VALUES ('a, dui.',38,116,0);
INSERT INTO parts (part_name,lead_time,cost,status) VALUES ('arcu et',37,72,1);
INSERT INTO parts (part_name,lead_time,cost,status) VALUES ('sapien.
Cras',40,197,1);
INSERT INTO parts (part_name,lead_time,cost,status) VALUES ('et
malesuada',24,46,0);
INSERT INTO parts (part_name,lead_time,cost,status) VALUES ('mauris id',4,153,1);
INSERT INTO parts (part_name,lead_time,cost,status) VALUES ('eleifend
egestas.',2,146,0);
INSERT INTO parts (part_name,lead_time,cost,status) VALUES ('cursus.
Nunc',9,194,1);
INSERT INTO parts (part_name,lead_time,cost,status) VALUES ('vivamus
sit',37,93,0);
INSERT INTO parts (part_name,lead_time,cost,status) VALUES ('ac orci.',35,134,0);
INSERT INTO parts (part_name,lead_time,cost,status) VALUES ('arcu.
Aliquam',36,154,0);
INSERT INTO parts (part_name,lead_time,cost,status) VALUES ('at auctor',32,56,1);
INSERT INTO parts (part_name,lead_time,cost,status) VALUES ('purus,
accumsan',33,12,1);

```

Thứ ba, chúng ta có một parts với một số dữ liệu mẫu để thực hành:

```

SELECT
    *
FROM
    parts
ORDER BY
    part_name;

```

PART_ID	PART_NAME	LEAD_TIME	COST	STATUS
1	sed dictum	5	134	0
2	tristique neque	3	62	1
3	dolor quam,	16	82	1
4	nec, diam.	41	10	1
5	vitae erat	22	116	0
6	parturient montes,	32	169	1
7	metus. In	45	88	1
8	at, velit.	31	182	0
9	nonummy ultricies	7	146	0
10	a, dui.	38	116	0
11	arcu et	37	72	1
12	sapient. Cras	40	197	1
13	et malesuada	24	46	0
14	mauris id	4	153	1
15	eleifend egestas.	2	146	0
16	cursus. Nunc	9	194	1
17	vivamus sit	37	93	0
18	ac orci.	35	134	0
19	arcu. Aliquam	36	154	0
20	at auctor	32	56	1
21	purus, accumsan	33	12	1

A) CẬP NHẬT Oracle – cập nhật một cột của một hàng

Câu lệnh sau UPDATE thay đổi chi phí của part có id 1:

```
UPDATE
  parts
SET
  cost = 130
WHERE
  part_id = 1;
```

Để xác minh bản cập nhật, bạn sử dụng truy vấn sau :

```
SELECT
  *
FROM
  parts
WHERE
  part_id = 1;
```

PART_ID	PART_NAME	LEAD_TIME	COST	STATUS
1	sed dictum	5	130	0

B) CẬP NHẬT Oracle – cập nhật nhiều cột của một hàng

Câu lệnh sau đây cập nhật thời gian thực hiện, chi phí và trạng thái của part có id là 5.

```
UPDATE
  parts
SET
  lead_time = 30,
  cost = 120,
  status = 1
WHERE
  part_id = 5;
```

PART_ID	PART_NAME	LEAD_TIME	COST	STATUS
5	vitae erat	30	120	1

C) CẬP NHẬT Oracle - ví dụ cập nhật nhiều hàng

Tuyên bố sau đây làm tăng chi phí thêm 5%:

```
UPDATE
  parts
SET
  cost = cost * 1.05;
```

Đây là kết quả:

PART_ID	PART_NAME	LEAD_TIME	COST	STATUS
1	sed dictum	5	136.5	0
2	tristique neque	3	65.1	1
3	dolor quam,	16	86.1	1
4	nec, diam.	41	10.5	1
5	vitae erat	30	126	1
6	parturient montes,	32	177.45	1
7	metus. In	45	92.4	1
8	at, velit.	31	191.1	0
9	nonummy ultricies	7	153.3	0
10	a, dui.	38	121.8	0

DELETE

Giới thiệu về câu lệnh DELETE

Để xóa một hoặc nhiều hàng khỏi bảng, bạn sử dụng DELETE như sau:

```
DELETE
FROM
    table_name
WHERE
    condition;
```

Trong câu lệnh này,

- Đầu tiên, bạn chỉ định tên của bảng mà bạn muốn xóa dữ liệu.
- Thứ hai, bạn chỉ định hàng nào sẽ bị xóa bằng cách sử dụng điều kiện trong mệnh đề WHERE. Nếu bạn bỏ qua mệnh đề WHERE này, câu lệnh DELETE sẽ xóa tất cả các hàng khỏi bảng.

Lưu ý rằng sẽ nhanh hơn và hiệu quả hơn khi sử dụng TRUNCATE TABLE câu lệnh để xóa tất cả các hàng khỏi một bảng lớn.

A) Oracle DELETE – xóa một hàng khỏi bảng

Câu lệnh sau xóa một hàng có order id là 1 and item id là 1:

```
DELETE
FROM
    orders
WHERE
    order_id = 1
    AND item_id = 1;
```

Oracle trả lại thông báo sau:

1 row deleted.

B) Oracle DELETE – xóa nhiều hàng khỏi một bảng

Câu lệnh sau xóa tất cả các hàng có order id là 1:

```
DELETE
FROM
    orders
WHERE
```

```
order_id = 1
```

Và Oracle trả lại thông báo sau:

12 rows deleted.

C) Oracle DELETE – xóa tất cả các hàng trong một bảng

Ví dụ sau xóa tất cả các hàng khỏi orders:

```
DELETE
FROM
  orders
```

D) Oracle DELETE – xóa cascade

Trong thực tế, bạn thường xóa một hàng khỏi bảng có mối quan hệ khóa ngoại (**foreign key**) với các hàng từ các bảng khác.

Ví dụ: bạn muốn xóa đơn hàng có id 1 khỏi bảng orders và cũng xóa tất cả các chi tiết đơn hàng được liên kết với order id 1 khỏi order_items. Thông thường, bạn có thể nghĩ đến việc đưa ra hai DELETE như sau:

```
DELETE
FROM
  orders
WHERE
  order_id = 1;

DELETE
FROM
  order_items
WHERE
  order_id = 1;

COMMIT WORK;
```

Lưu ý rằng COMMIT WORK đảm bảo cả hai câu lệnh DELETE đều thực thi theo cách "tất cả hoặc không có gì"..

Tuy nhiên, điều này là không cần thiết nếu bạn biết cách thiết lập ràng buộc của bảng một cách chính xác.

Trong trường hợp này, khi tạo order_items, bạn xác định ràng buộc khóa ngoại với tùy chọn như sau:

```
CREATE TABLE order_items
(
    order_id    NUMBER( 12, 0 )
    -- other columns
    -- ...
    CONSTRAINT fk_order_items_orders
    FOREIGN KEY( order_id )
    REFERENCES orders( order_id )
    ON DELETE CASCADE
);
```

Bằng cách này, bất cứ khi nào xóa một hàng khỏi orders, ví dụ:

```
DELETE
FROM
    orders
WHERE
    order_id = 1;
```

Tất cả các hàng có order id là 1 trong order_items cũng sẽ bị hệ thống cơ sở dữ liệu tự động xóa.

MERGE

Giới thiệu về câu lệnh Oracle MERGE

MERGE lấy dữ liệu từ một hoặc nhiều bảng và cập nhật hoặc thêm dữ liệu đó vào bảng đích. Câu lệnh MERGE cho phép bạn chỉ định một điều kiện để xác định xem nên cập nhật dữ liệu từ hay thêm dữ liệu vào bảng mục tiêu.

Sau đây minh họa cú pháp của câu lệnh Oracle MERGE:

```
MERGE INTO target_table
USING source_table
ON search_condition
  WHEN MATCHED THEN
    UPDATE SET col1 = value1, col2 = value2,...
    WHERE <update_condition>
    [DELETE WHERE <delete_condition>]
  WHEN NOT MATCHED THEN
    INSERT (col1,col2,...)
    values(value1,value2,...)
    WHERE <insert_condition>;
```

Trong câu lệnh MERGE này:

- Đầu tiên, chỉ định bảng mục tiêu (target_table) trong mệnh đề INTO mà bạn muốn cập nhật hoặc thêm.
- Thứ hai, chỉ định nguồn dữ liệu (source_table) sẽ được cập nhật hoặc chèn vào mệnh đề USING.
- Thứ ba, chỉ định điều kiện tìm kiếm mà thao tác cập nhật hoặc chèn vào mệnh đề ON.

Đối với mỗi hàng trong bảng mục tiêu, Oracle đánh giá điều kiện tìm kiếm:

- Nếu kết quả là đúng thì Oracle sẽ cập nhật hàng với dữ liệu tương ứng từ bảng nguồn.
- Trong trường hợp kết quả là sai đối với bất kỳ hàng nào thì Oracle sẽ chèn hàng tương ứng từ bảng nguồn vào bảng đích.

Câu lệnh MERGE trở nên thuận tiện khi bạn muốn kết hợp nhiều câu lệnh INSERT, UPDATE, và DELETE trong một thao tác duy nhất.

Vì MERGE là một câu lệnh xác định nên bạn không thể cập nhật cùng một hàng của bảng mục tiêu nhiều lần trong cùng một câu lệnh MERGE.

Bạn có thể thêm DELETE WHERE vào MATCHED để dọn dẹp sau thao tác merge. Mệnh đề DELETE chỉ xóa các hàng trong bảng đích khớp với cả hai mệnh đề ON và DELETE WHERE.

Điều kiện tiên quyết của Oracle MERGE

Để thực thi câu lệnh MERGE, bạn phải có quyền INSERT và UPDATE trên bảng nguồn. Nếu bạn sử dụng DELETE, bạn cũng phải có quyền DELETE trên bảng mục tiêu.

Ví dụ về MERGE của Oracle

Giả sử chúng ta có hai bảng: members và member_staging.

Chúng tôi chèn một hàng mới vào members bất cứ khi nào chúng tôi có thành viên mới. Sau đó, dữ liệu từ members bảng được hợp nhất với dữ liệu của member_staging.

Các câu lệnh sau đây tạo ra các bảng members và member_staging:

```
CREATE TABLE members (  
    member_id NUMBER PRIMARY KEY,  
    first_name VARCHAR2(50) NOT NULL,  
    last_name VARCHAR2(50) NOT NULL,  
    rank VARCHAR2(20)  
);  
  
CREATE TABLE member_staging AS  
SELECT * FROM members;
```

Các câu lệnh sau INSERT chèn dữ liệu mẫu vào bảng members và member_staging:

-- insert into members table

```
-- insert into members table  
INSERT INTO members(member_id, first_name, last_name, rank)  
VALUES(1,'Abel','Wolf','Gold');  
INSERT INTO members(member_id, first_name, last_name, rank)  
VALUES(2,'Clarita','Franco','Platinum');  
INSERT INTO members(member_id, first_name, last_name, rank)  
VALUES(3,'Darryl','Giles','Silver');  
INSERT INTO members(member_id, first_name, last_name, rank)  
VALUES(4,'Dorthea','Suarez','Silver');  
INSERT INTO members(member_id, first_name, last_name, rank)  
VALUES(5,'Katrina','Wheeler','Silver');  
INSERT INTO members(member_id, first_name, last_name, rank)  
VALUES(6,'Lilian','Garza','Silver');  
INSERT INTO members(member_id, first_name, last_name, rank)  
VALUES(7,'Ossie','Summers','Gold');  
INSERT INTO members(member_id, first_name, last_name, rank)  
VALUES(8,'Paige','Mcfarland','Platinum');
```

```

INSERT INTO members(member_id, first_name, last_name, rank)
VALUES(9,'Ronna','Britt','Platinum');
INSERT INTO members(member_id, first_name, last_name, rank)
VALUES(10,'Tressie','Short','Bronze');

-- insert into member_staging table
INSERT INTO member_staging(member_id, first_name, last_name, rank)
VALUES(1,'Abel','Wolf','Silver');
INSERT INTO member_staging(member_id, first_name, last_name, rank)
VALUES(2,'Clarita','Franco','Platinum');
INSERT INTO member_staging(member_id, first_name, last_name, rank)
VALUES(3,'Darryl','Giles','Bronze');
INSERT INTO member_staging(member_id, first_name, last_name, rank)
VALUES(4,'Dorthea','Gate','Gold');
INSERT INTO member_staging(member_id, first_name, last_name, rank)
VALUES(5,'Katrina','Wheeler','Silver');
INSERT INTO member_staging(member_id, first_name, last_name, rank)
VALUES(6,'Lilian','Stark','Silver');

```

Khi cập nhật dữ liệu từ bảng members này sang bảng member_staging khác, chúng ta nên thực hiện các thao tác sau:

- Chúng tôi cập nhật các hàng có id thành viên 1, 3, 4 và 6 vì cấp bậc hoặc last name của các thành viên này trong các bảng này là khác nhau.
- Chúng tôi chèn các hàng có id thành viên từ 7 đến 10 là do các hàng này tồn tại trong bảng members nhưng không có trong bảng member_staging.

Tổng cộng có 8 hàng nên được merge.

MEMBER_ID	FIRST_NAME	LAST_NAME	RANK		MEMBER_ID	FIRST_NAME	LAST_NAME	RANK
1	Abel	Wolf	Gold	→	1	Abel	Wolf	Silver
2	Clarita	Franco	Platinum		2	Clarita	Franco	Platinum
3	Darryl	Giles	Silver	→	3	Darryl	Giles	Bronze
4	Dorthea	Suarez	Silver	→	4	Dorthea	Gate	Gold
5	Katrina	Wheeler	Silver		5	Katrina	Wheeler	Silver
6	Lilian	Garza	Silver	→	6	Lilian	Stark	Silver
7	Ossie	Summers	Gold					
8	Paige	Mcfarland	Platinum					
9	Ronna	Britt	Platinum					
10	Tressie	Short	Bronze					

Sau đây là câu lệnh MERGE thực hiện tất cả các hành động này trong một lần.

```
MERGE INTO member_staging x
USING (SELECT member_id, first_name, last_name, rank FROM members) y
ON (x.member_id = y.member_id)
WHEN MATCHED THEN
    UPDATE SET x.first_name = y.first_name,
               x.last_name = y.last_name,
               x.rank = y.rank
    WHERE x.first_name <> y.first_name OR
          x.last_name <> y.last_name OR
          x.rank <> y.rank
WHEN NOT MATCHED THEN
    INSERT(x.member_id, x.first_name, x.last_name, x.rank)
    VALUES(y.member_id, y.first_name, y.last_name, y.rank);
```

Câu lệnh merge so sánh từng hàng trong bảng members với mỗi hàng trong bảng member_staging dựa trên các giá trị trong các cột member_id.

Section 10. Data definition

CREATE TABLE

Giới thiệu câu lệnh CREATE TABLE

Để tạo một bảng mới trong Cơ sở dữ liệu Oracle, bạn sử dụng câu lệnh TABLE CREATE. Cú pháp cơ bản của CREATE TABLE:

```
CREATE TABLE schema_name.table_name (  
    column_1 data_type column_constraint,  
    column_2 data_type column_constraint,  
    ...  
    table_constraint  
);
```

Trong cú pháp này:

- Đầu tiên, chỉ định tên bảng và tên lược đồ chứa bảng mới trong mệnh đề CREATE TABLE.
- Thứ hai, liệt kê tất cả các cột của bảng trong dấu ngoặc đơn. Trường hợp bảng có nhiều cột thì các bạn cần ngăn cách chúng bằng dấu phẩy (.). Định nghĩa cột bao gồm tên cột theo sau là kiểu dữ liệu của nó, ví dụ NUMBER: VARCHAR2, và một ràng buộc cột như NOT NULL, primary key, check.
- Thứ ba, thêm các ràng buộc về bảng nếu có, ví dụ: primary key, foreign key, check..

Ví dụ về câu lệnh CREATE TABLE

Ví dụ sau đây cho thấy cách tạo một bảng mới có tên persons:

```
CREATE TABLE ot.persons(  
    person_id NUMBER GENERATED BY DEFAULT AS IDENTITY,  
    first_name VARCHAR2(50) NOT NULL,  
    last_name VARCHAR2(50) NOT NULL,  
    PRIMARY KEY(person_id)  
);
```

Trong ví dụ này, bảng persons có ba cột: person_id, first_name, và last_name.

Đây person_id là identity column xác định các hàng duy nhất trong bảng. Kiểu dữ liệu của cột person_id là NUMBER. Mệnh đề GENERATED BY DEFAULT AS IDENTITY tạo một số nguyên mới cho cột bất cứ khi nào một hàng mới được insert vào bảng.

Cột first_name có kiểu dữ liệu VARCHAR2 có độ dài tối đa là 50. Điều đó có nghĩa là không thể chèn tên có độ dài lớn hơn 50 vào cột first_name. Ngoài ra, ràng buộc cột NOT NULL ngăn cột first_name có giá trị NULL.

Mệnh đề PRIMARY KEY chỉ định person_id là cột khóa chính được sử dụng để xác định hàng duy nhất trong bảng persons.

Identity Column

Giới thiệu về Oracle identity column

Oracle 12c đã giới thiệu một cách mới cho phép bạn xác định **identity column** cho một bảng, tương tự như cột **AUTO_INCREMENT** trong MySQL hoặc **IDENTITY** trong SQL Server.

Identity column rất hữu ích cho primary key. Khi bạn insert một hàng mới vào identity column, Oracle sẽ tự động tạo và thêm một giá trị tuần tự vào column.

Để xác định một identity column, sử dụng mệnh đề **identity** như dưới đây:

```
GENERATED [ ALWAYS | BY DEFAULT [ ON NULL ] ]  
AS IDENTITY [ ( identity_options ) ]
```

Đầu tiên, từ khóa GENERATED là bắt buộc.

Thứ hai, có thể chỉ định một tùy chọn để tạo các giá trị identity:

- **GENERATED ALWAYS:** Oracle luôn tạo một giá trị cho identity column. Cố gắng chèn một giá trị vào identity column sẽ gây ra lỗi.
- **GENERATED BY DEFAULT:** Oracle tạo giá trị cho identity column nếu bạn không cung cấp giá trị. Nếu bạn cung cấp một giá trị, Oracle sẽ chèn giá trị đó vào identity column. Đối với tùy chọn này, Oracle sẽ báo lỗi nếu bạn thêm giá trị NULL vào identity column.
- **GENERATED BY DEFAULT ON NULL:** Oracle tạo giá trị cho identity column nếu bạn cung cấp giá trị NULL hoặc không có giá trị nào cả.

Thứ ba, bạn có thể có một số tùy chọn cho identity column.

- **START WITH initial_value** kiểm soát giá trị ban đầu để sử dụng cho identity column. Giá trị ban đầu mặc định là 1.
- **INCREMENT BY interval_value** xác định khoảng cách giữa các giá trị được tạo ra. Theo mặc định, giá trị khoảng là 1.
- **CACHE** xác định một số giá trị mà Oracle nên tạo trước để cải thiện hiệu suất.

A) Ví dụ GENERATED ALWAYS

```
CREATE TABLE identity_demo (  
    id NUMBER GENERATED ALWAYS AS IDENTITY,  
    description VARCHAR2(100) NOT NULL  
);
```

Câu lệnh sau chèn một hàng mới vào identity_demo:

```
INSERT INTO identity_demo(description)
VALUES('Oracle identity column demo with GENERATED ALWAYS');
```

Vì không chỉ định giá trị cho id cột nên Oracle tự động tạo ra một giá trị tuần tự bắt đầu từ 1.

```
SELECT
    *
FROM
    identity_demo;
```

ID	DESCRIPTION
1	Oracle identity column demo with GENERATED ALWAYS

Câu lệnh sau cố gắng chèn một giá trị vào identity column:

```
INSERT INTO identity_demo(id,description)
VALUES(2,
    'Oracle identity column example with GENERATED ALWAYS ');
```

Oracle đưa ra lỗi:

```
SQL Error: ORA-32795: cannot insert into a generated always identity column
```

Vì cột id được xác định là GENERATED ALWAYS, nên cột này không thể chấp nhận bất kỳ giá trị nào được cung cấp.

B) Ví dụ *GENERATED BY DEFAULT*

```
DROP TABLE identity_demo;
```

```
DROP TABLE identity_demo;
```

```
CREATE TABLE identity_demo (
    id NUMBER GENERATED BY DEFAULT AS IDENTITY,
    description VARCHAR2(100) not null
);
```

Câu lệnh sau chèn một hàng mới vào identity_demo:

```
INSERT INTO identity_demo(description)
VALUES('Oracle identity column demo with GENERATED BY DEFAULT');
```

ID	DESCRIPTION
1	Oracle identity column demo with GENERATED BY DEFAULT

Câu lệnh sau chèn một hàng mới vào identity_demo với giá trị được cung cấp cho cột id:

```
INSERT INTO identity_demo(id,description)
VALUES(2, 'Oracle identity column example with GENERATED BY DEFAULT');
```

Trong ví dụ này, Oracle đã sử dụng giá trị được cung cấp và chèn nó vào bảng.

```
SELECT
    *
FROM
    identity_demo;
```

ID	DESCRIPTION
1	Oracle identity column demo with GENERATED BY DEFAULT
2	Oracle identity column example with GENERATED BY DEFAULT

Ví dụ sau cố gắng chèn một giá trị null vào cột id:

```
INSERT INTO identity_demo(id,description)
VALUES(NULL,
        'Oracle identity column demo with GENERATED BY DEFAULT, NULL value');
```

Oracle đã đưa ra lỗi:

```
SQL Error: ORA-01400: cannot insert NULL into ("OT"."IDENTITY_DEMO"."ID")
```

C) Ví dụ *GENERATED BY DEFAULT ON NULL*

```
DROP TABLE identity_demo;

CREATE TABLE identity_demo (
    id NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY,
    description VARCHAR2(100) not null
);
```

Câu lệnh sau không cung cấp giá trị cho cột id, Oracle sẽ tự động tạo giá trị để chèn:

```
INSERT INTO identity_demo(description)
VALUES('Oracle identity column demo with no value');
```

Câu lệnh sau chèn một giá trị NULL vào cột id vì cột id đã được xác định là GENERATED BY DEFAULT ON NULL, Oracle tạo ra một giá trị tuần tự và sử dụng nó để thêm:

```
INSERT INTO identity_demo(description)
VALUES('Oracle identity column demo with null');
```

ID	DESCRIPTION
1	Oracle identity column demo with no value
2	Oracle identity column demo with null

D) Ví dụ về START WITH

```
DROP TABLE identity_demo;

CREATE TABLE identity_demo (
    id NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY START WITH 100,
    description VARCHAR2(100) not null
);
```

Thứ hai, chèn một hàng vào identity_demo:

```
INSERT INTO identity_demo(description)
VALUES('Oracle identity column demo with START WITH option');
```

Thứ ba, truy vấn dữ liệu từ identity_demo:

```
SELECT
    *
FROM
    identity_demo;
```

ID	DESCRIPTION
100	Oracle identity column demo with START WITH option

E) Ví dụ về INCREMENT BY

Đầu tiên, thay đổi cột id của identity_demo bảng bao gồm cả hai START WITH và INCREMENT BY chọn.

```
DROP TABLE identity_demo;

CREATE TABLE identity_demo (
    id NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY START WITH 10 INCREMENT BY 10,
    description VARCHAR2(100) not null
);
```

Thứ hai, chèn hai hàng vào identity_demo:

```
INSERT INTO identity_demo(description)
VALUES('Oracle identity column demo 1 with INCREMENT BY option');

INSERT INTO identity_demo(description)
VALUES('Oracle identity column demo 2 with INCREMENT BY option');
```

Thứ ba, truy vấn dữ liệu từ bảng để xác minh các phần insert:

```
SELECT
```

```
*
FROM
  identity_demo;
```

ID	DESCRIPTION
10	Oracle identity column demo 1 with INCREMENT BY option
20	Oracle identity column demo 2 with INCREMENT BY option

ALTER TABLE

Để sửa đổi cấu trúc của một bảng hiện có, bạn sử dụng câu lệnh ALTER TABLE. Sau đây minh họa cú pháp:

```
ALTER TABLE table_name action;
```

Trong câu lệnh này:

- Đầu tiên, chỉ định tên bảng mà bạn muốn sửa đổi.
- Thứ hai, cho biết hành động bạn muốn thực hiện sau tên bảng.

ALTER TABLE cho phép bạn:

- Thêm một hoặc nhiều cột
- Sửa đổi định nghĩa cột
- Bỏ một hoặc nhiều cột
- Đổi tên cột
- Đổi tên bảng

Ví dụ về ALTER TABLE ADD column

Để thêm một cột mới vào bảng, sử dụng cú pháp sau:

```
ALTER TABLE table_name
ADD column_name type constraint;
```

Ví dụ: câu lệnh sau thêm một cột mới có tên birthdate vào persons:

```
ALTER TABLE persons
ADD birthdate DATE NOT NULL;
```

Nếu bạn xem persons, sẽ thấy birthdate được thêm vào cuối danh sách cột:

```
DESC persons;
```

Name	Null	Type
------	------	------

```

-----
PERSON_ID  NOT NULL  NUMBER
FIRST_NAME NOT NULL  VARCHAR2(50)
LAST_NAME  NOT NULL  VARCHAR2(50)
BIRTHDATE  NOT NULL  DATE

```

Để thêm nhiều cột vào bảng cùng lúc, đặt các cột mới vào trong dấu ngoặc đơn như sau:

```

ALTER TABLE table_name
ADD (
    column_name type constraint,
    column_name type constraint,
    ...
);

```

Xem ví dụ sau:

```

ALTER TABLE persons
ADD (
    phone VARCHAR(20),
    email VARCHAR(100)
);

```

Trong ví dụ này, câu lệnh đã thêm hai cột mới có tên phone và email vào persons.

```

DESC persons

Name          Null          Type
-----
PERSON_ID     NOT NULL      NUMBER
FIRST_NAME    NOT NULL      VARCHAR2(50)
LAST_NAME     NOT NULL      VARCHAR2(50)
BIRTHDATE     NOT NULL      DATE
PHONE                     VARCHAR2(20)
EMAIL                     VARCHAR2(100)

```

Ví dụ về ALTER TABLE MODIFY column

Để sửa đổi các thuộc tính của một cột, bạn sử dụng cú pháp sau:

```

ALTER TABLE table_name
    MODIFY column_name type constraint;

```

Ví dụ: câu lệnh sau thay đổi birthdate thành column có thể null:

```
ALTER TABLE persons MODIFY birthdate DATE NULL;
```

Hãy xác minh lại cấu trúc bảng persons:

```
DESC persons
```

Name	Null	Type
-----	-----	-----
PERSON_ID	NOT NULL	NUMBER
FIRST_NAME	NOT NULL	VARCHAR2(50)
LAST_NAME	NOT NULL	VARCHAR2(50)
BIRTHDATE		DATE
PHONE		VARCHAR2(20)
EMAIL		VARCHAR2(100)

Như bạn có thể thấy, birthdate đã trở thành null-able.

Để sửa đổi nhiều cột, sử dụng cú pháp sau:

```
ALTER TABLE table_name
MODIFY ( column_1 type constraint,
        column_1 type constraint,
        ...);
```

Ví dụ: câu lệnh sau thay đổi cột phone và email thành NOT NULL và kéo dài độ dài của email thành 255 ký tự:

```
ALTER TABLE persons MODIFY(
    phone VARCHAR2(20) NOT NULL,
    email VARCHAR2(255) NOT NULL
);
```

Xác minh lại cấu trúc bảng persons:

```
DESC persons;
```

Name	Null	Type
-----	-----	-----
PERSON_ID	NOT NULL	NUMBER
FIRST_NAME	NOT NULL	VARCHAR2(50)
LAST_NAME	NOT NULL	VARCHAR2(50)
BIRTHDATE		DATE
PHONE	NOT NULL	VARCHAR2(20)
EMAIL	NOT NULL	VARCHAR2(255)

Ví dụ về ALTER TABLE DROP COLUMN

Để xóa một cột hiện có khỏi bảng, bạn sử dụng cú pháp sau:

```
ALTER TABLE table_name  
DROP COLUMN column_name;
```

Câu lệnh này xóa cột khỏi cấu trúc bảng và cả dữ liệu được lưu trữ trong cột đó.

Ví dụ sau xóa birthdate khỏi bảng persons:

```
ALTER TABLE persons  
DROP  
COLUMN birthdate;
```

Xem lại cấu trúc bảng persons:

```
DESC persons;
```

Name	Null	Type
-----	-----	-----
PERSON_ID	NOT NULL	NUMBER
FIRST_NAME	NOT NULL	VARCHAR2(50)
LAST_NAME	NOT NULL	VARCHAR2(50)
PHONE	NOT NULL	VARCHAR2(20)
EMAIL	NOT NULL	VARCHAR2(255)

Để xóa nhiều cột cùng lúc, sử dụng cú pháp dưới đây:

```
ALTER TABLE table_name  
DROP (column_1,column_2,...);
```

Ví dụ: câu lệnh sau sẽ xóa các cột phone và email khỏi bảng persons:

```
ALTER TABLE persons  
DROP  
( email, phone );
```

Hãy kiểm tra lại bảng persons:

```
DESC persons;
```

Name	Null	Type
-----	-----	-----
PERSON_ID	NOT NULL	NUMBER


```
FIRST_NAME NOT NULL VARCHAR2(50)
LAST_NAME NOT NULL VARCHAR2(50)
```

Ví dụ về ALTER TABLE RENAME column

```
ALTER TABLE table_name
RENAME COLUMN column_name TO new_name;
```

Ví dụ: câu lệnh sau đổi tên cột first_name thành cột forename:

```
ALTER TABLE persons
RENAME COLUMN first_name TO forename;
```

Câu lệnh sau kiểm tra kết quả:

```
DESC persons;

Name          Null          Type
-----
PERSON_ID     NOT NULL      NUMBER
FORENAME      NOT NULL      VARCHAR2(50)
LAST_NAME     NOT NULL      VARCHAR2(50)
```

Ví dụ về ALTER TABLE RENAME table

Để đặt tên mới cho bảng, sử dụng cú pháp sau:

```
ALTER TABLE table_name
RENAME TO new_table_name;
```

Ví dụ: câu lệnh dưới đây đổi tên bảng persons thành people:

```
ALTER TABLE persons
RENAME TO people;
```

DROP TABLE

Giới thiệu về câu lệnh Oracle DROP TABLE

Để di chuyển một bảng vào recycle bin hoặc xóa nó hoàn toàn khỏi cơ sở dữ liệu, bạn sử dụng câu lệnh DROP TABLE:

```
DROP TABLE schema_name.table_name  
[CASCADE CONSTRAINTS | PURGE];
```

Trong tuyên bố này:

- Đầu tiên, chỉ ra bảng và schema của nó mà bạn muốn xóa sau mệnh đề DROP TABLE. Nếu bạn không chỉ định rõ ràng tên schema, câu lệnh sẽ giả định rằng bạn đang xóa bảng khỏi schema của chính mình.
- Thứ hai, chỉ định CASCADE CONSTRAINTS mệnh đề để loại bỏ tất cả các ràng buộc toàn vẹn tham chiếu đến primary và unique keys trong bảng. Trong trường hợp các ràng buộc toàn vẹn tham chiếu như vậy tồn tại và bạn không sử dụng mệnh đề này, Oracle sẽ trả về lỗi và ngừng xóa bảng.
- Thứ ba, chỉ định mệnh đề PURGE nếu bạn muốn loại bỏ bảng và giải phóng khoảng trống liên kết với nó cùng một lúc. Bằng cách sử dụng mệnh đề PURGE này, Oracle sẽ không đặt bảng và các đối tượng phụ thuộc của nó vào recycle bin.

Lưu ý rằng PURGE mệnh đề không cho phép bạn rollback hoặc khôi phục bảng mà bạn đã xóa. Vì vậy, sẽ rất hữu ích nếu bạn không muốn dữ liệu nhạy cảm xuất hiện trong recycle bin.

Ví dụ cơ bản về DROP TABLE

Câu lệnh CREATE TABLE tạo bảng persons:

```
CREATE TABLE persons (  
    person_id NUMBER,  
    first_name VARCHAR2(50) NOT NULL,  
    last_name VARCHAR2(50) NOT NULL,  
    PRIMARY KEY(person_id)  
);
```

Ví dụ sau loại bỏ bảng persons khỏi cơ sở dữ liệu:

```
DROP TABLE persons;
```

Ví dụ về DROP TABLE CASCADE CONSTRAINTS

Các câu lệnh sau đây tạo ra hai bảng mới có tên là brands và cars:

```
CREATE TABLE brands(  
    brand_id NUMBER PRIMARY KEY,  
    brand_name varchar2(50)  
);  
  
CREATE TABLE cars(  
    car_id NUMBER PRIMARY KEY,  
    make VARCHAR(50) NOT NULL,  
    model VARCHAR(50) NOT NULL,  
    year NUMBER NOT NULL,  
    plate_number VARCHAR(25),  
    brand_id NUMBER NOT NULL,  
  
    CONSTRAINT fk_brand  
    FOREIGN KEY (brand_id)  
    REFERENCES brands(brand_id) ON DELETE CASCADE  
);
```

Trong các bảng này, mỗi hãng có từ 1 xe trở lên trong khi mỗi xe chỉ thuộc một hãng.

Câu lệnh sau đây cố gắng loại bỏ bảng brands:

```
DROP TABLE brands;
```

Oracle đã đưa ra lỗi sau:

```
ORA-02449: unique/primary keys in table referenced by foreign keys
```

Điều này là do khóa chính của bảng brands hiện được tham chiếu bởi cột brand_id trong bảng cars.

Phải sử dụng mệnh đề CASCADE CONSTRAINTS như sau:

```
DROP TABLE brands CASCADE CONSTRAINTS;
```

Câu lệnh này không chỉ loại bỏ brands mà còn loại bỏ ràng buộc foreign key fk_brand khỏi bảng cars.

TRUNCATE TABLE

Giới thiệu câu lệnh TRUNCATE TABLE của Oracle

Khi muốn xóa toàn bộ dữ liệu trong một bảng, sử dụng câu lệnh DELETE không có mệnh đề WHERE như sau:

```
DELETE FROM table_name;
```

Đối với một bảng có số lượng hàng nhỏ, câu lệnh DELETE sẽ hoạt động tốt. Tuy nhiên, khi có một bảng có số lượng hàng lớn thì việc sử dụng DELETE để xóa toàn bộ dữ liệu sẽ không hiệu quả.

Oracle đã giới thiệu câu lệnh TRUNCATE TABLE cho phép xóa tất cả các hàng khỏi một bảng lớn.

Sau đây minh họa cú pháp của TRUNCATE TABLE:

```
TRUNCATE TABLE schema_name.table_name  
[CASCADE]  
[[ PRESERVE | PURGE] MATERIALIZED VIEW LOG ]]  
[[ DROP | REUSE]] STORAGE ]
```

Theo mặc định, để xóa tất cả các hàng khỏi một bảng, chỉ định tên của bảng mà muốn truncate trong TRUNCATE TABLE:

```
TRUNCATE TABLE table_name;
```

Trong trường hợp này, vì chúng ta không chỉ định rõ ràng tên schema nên Oracle giả định rằng đã TRUNCATE bảng khỏi schema hiện tại.

Nếu một bảng có mối quan hệ với các bảng khác thông qua ràng buộc foreign key, cần sử dụng mệnh đề CASCADE:

```
TRUNCATE TABLE table_name  
CASCADE;
```

Trong trường hợp này, TRUNCATE TABLE CASCADE sẽ xóa tất cả các hàng khỏi table_name, và truncate các bảng liên quan trong chuỗi một cách đệ quy.

A) Ví dụ đơn giản về TRUNCATE TABLE

Câu lệnh sau tạo một bảng có tên customers_copy và sao chép dữ liệu từ:

```
CREATE TABLE customers_copy  
AS  
SELECT  
*
```

```
FROM  
customers;
```

Để xóa tất cả các hàng khỏi bảng customers_copy, bạn sử dụng câu lệnh sau TRUNCATE TABLE:

```
TRUNCATE TABLE customers_copy;
```

B) Ví dụ về TRUNCATE TABLE CASCADE

Đầu tiên, hãy tạo bảng quotations và quotation_items:

```
CREATE TABLE quotations (  
    quotation_no NUMERIC GENERATED BY DEFAULT AS IDENTITY,  
    customer_id NUMERIC NOT NULL,  
    valid_from DATE NOT NULL,  
    valid_to DATE NOT NULL,  
    PRIMARY KEY(quotation_no)  
);  
  
CREATE TABLE quotation_items (  
    quotation_no NUMERIC,  
    item_no NUMERIC ,  
    product_id NUMERIC NOT NULL,  
    qty NUMERIC NOT NULL,  
    price NUMERIC(9 , 2 ) NOT NULL,  
    PRIMARY KEY (quotation_no , item_no),  
    CONSTRAINT fk_quotation FOREIGN KEY (quotation_no)  
        REFERENCES quotations  
        ON DELETE CASCADE  
);
```

Tiếp theo, insert một số hàng vào hai bảng này:

```
INSERT INTO quotations(customer_id, valid_from, valid_to)  
VALUES(100, DATE '2017-09-01', DATE '2017-12-01');  
  
INSERT INTO quotation_items(quotation_no, item_no, product_id, qty, price)  
VALUES(1,1,1001,10,90.5);  
  
INSERT INTO quotation_items(quotation_no, item_no, product_id, qty, price)  
VALUES(1,2,1002,20,200.5);  
  
INSERT INTO quotation_items(quotation_no, item_no, product_id, qty, price)  
VALUES(1,3,1003,30, 150.5);
```

Sau đó, truncate bảng quotation :

```
TRUNCATE TABLE quotations;
```

Câu lệnh không thành công và Oracle trả về lỗi sau:

```
SQL Error: ORA-02266: unique/primary keys in table referenced by enabled foreign keys
```

Để khắc phục điều này, thêm mệnh đề CASCADE vào câu lệnh TRUNCATE TABLE trên:

```
TRUNCATE TABLE quotations CASCADE;
```

Câu lệnh này đã xóa dữ liệu khỏi không chỉ quotations mà còn cả quotation_items.

Section 11. Oracle data types

Oracle data types

Giới thiệu về các kiểu dữ liệu của Oracle

Trong Oracle, mọi giá trị đều có một kiểu dữ liệu xác định một tập hợp các đặc điểm cho giá trị đó. Những đặc điểm này khiến Oracle xử lý các giá trị của một kiểu dữ liệu khác với các giá trị của kiểu dữ liệu khác.

Khi tạo một bảng mới, chỉ định loại dữ liệu cho mỗi cột của nó. Tương tự, khi tạo một procedure mới, chỉ định kiểu dữ liệu cho mỗi đối số của nó.

Kiểu dữ liệu xác định các giá trị được phép mà mỗi cột hoặc đối số có thể lưu trữ. Ví dụ: DATE không thể lưu trữ giá trị ngày 30 tháng 2 vì đây không phải là ngày hợp lệ.

Oracle có một số kiểu dữ liệu tích hợp được minh họa trong bảng sau:

Code	Data Type
1	VARCHAR2(size [BYTE CHAR])
1	NVARCHAR2(size)
2	NUMBER[(precision [, scale])]
8	LONG
12	DATE
21	BINARY_FLOAT
22	BINARY_DOUBLE
23	RAW(size)
24	LONG RAW
69	ROWID
96	CHAR [(size [BYTE CHAR])]
96	NCHAR[(size)]
112	CLOB
112	NCLOB
113	BLOB
114	BFILE
180	TIMESTAMP [(fractional_seconds)]
181	TIMESTAMP [(fractional_seconds)] WITH TIME ZONE
182	INTERVAL YEAR [(year_precision)] TO MONTH
183	INTERVAL DAY [(day_precision)] TO SECOND[(fractional_seconds)]
208	UROWID [(size)]
231	TIMESTAMP [(fractional_seconds)] WITH LOCAL TIMEZONE

Mỗi loại dữ liệu có một code do Oracle quản lý nội bộ. Để tìm code kiểu dữ liệu của một giá trị trong một cột, bạn sử dụng DUMP().

Kiểu dữ liệu ký tự

- Các kiểu dữ liệu ký tự bao gồm **CHAR** , **NCHAR**, **VARCHAR2** , **NVARCHAR2** và **VARCHAR**.
- Kiểu dữ liệu **NCHAR** và **NVARCHAR2** dùng để lưu trữ chuỗi ký tự Unicode.
- Các kiểu dữ liệu ký tự có độ dài cố định là **CHAR**, **NCHAR** và các kiểu dữ liệu ký tự có độ dài thay đổi là **VARCHAR2**, **NVARCHAR2**.
- **VARCHAR** là đồng nghĩa của **VARCHAR2**. Tuy nhiên, bạn không nên sử dụng **VARCHAR** vì Oracle có thể thay đổi ngữ nghĩa của nó trong tương lai.
- Đối với các kiểu dữ liệu ký tự, bạn có thể chỉ định kích thước của chúng theo byte hoặc ký tự.

Kiểu dữ liệu số

- Kiểu dữ liệu NUMBER có độ chính xác **p** và tỷ lệ **s** . Độ chính xác nằm trong khoảng từ 1 đến 38 trong khi thang đo nằm trong khoảng từ -84 đến 127.
- Nếu bạn không chỉ định độ chính xác, cột có thể lưu trữ các giá trị bao gồm số dấu phẩy động. Giá trị mặc định cho thang đo là 0.

Kiểu dữ liệu ngày giờ và khoảng thời gian

- Các kiểu dữ liệu ngày giờ là **DATE** , **TIMESTAMP** , **TIMESTAMP WITH TIME ZONE** và **TIMESTAMP WITH TIME TIME ZONE**. Các giá trị của kiểu dữ liệu datetime là datetimes.
- Các kiểu dữ liệu khoảng thời gian là **INTERVAL YEAR TO MONTH** và **INTERVAL DAY TO SECOND** . Các giá trị của kiểu dữ liệu interval là **intervals**.

Các kiểu dữ liệu: Oracle và ANSI

Khi sử dụng các kiểu dữ liệu ANSI cho các định nghĩa cột, Oracle sẽ chuyển đổi chúng thành các kiểu dữ liệu tương ứng trong Oracle dựa trên bảng ánh xạ sau:

ANSI SQL Datatype	Oracle Data Type
CHARACTER(n)	CHAR(n)
CHAR(n)	
CHARACTER VARYING(n)	VARCHAR2(n)
CHAR VARYING(n)	
NATIONAL CHARACTER(n)	NCHAR(n)
NATIONAL CHAR(n)	
NCHAR(n)	
NATIONAL CHARACTER VARYING(n)	NVARCHAR2(n)
NATIONAL CHAR VARYING(n)	
NCHAR VARYING(n)	

NUMERIC(p,s)	NUMBER(p,s)
DECIMAL(p,s) (a)	
INTEGER	NUMBER(38)
INT	
SMALLINT	
FLOAT (b)	NUMBER
DOUBLE PRECISION (c)	
REAL (d)	

NUMBER

Giới thiệu về kiểu dữ liệu NUMBER

Kiểu dữ liệu NUMBER được sử dụng để lưu trữ các giá trị số có thể âm hoặc dương.

```
NUMBER[(precision [, scale])]
```

Kiểu dữ liệu Oracle NUMBER có độ **chính xác(precision)** và **tỷ lệ (scale)**.

- Độ chính xác là số chữ số trong một số. Nó dao động từ 1 đến 38.
- Tỷ lệ là số chữ số ở bên phải dấu thập phân trong một số. Nó dao động từ -84 đến 127.

Ví dụ: số 1234,56 có độ chính xác là 6 và tỷ lệ là 2. Vì vậy, để lưu trữ số này, bạn cần NUMBER(6,2).

Cả độ chính xác và tỷ lệ đều ở dạng chữ số thập phân và tùy chọn. Nếu bạn bỏ qua độ chính xác và tỷ lệ, Oracle sẽ sử dụng phạm vi và độ chính xác tối đa cho số đó.

Ví dụ: biểu mẫu sau xác định một số có thể lưu trữ các giá trị số với phạm vi và độ chính xác tối đa:

```
NUMBER
```

Cú pháp sau đây định nghĩa một số cố định:

```
NUMBER(p,s)
```

Để xác định một số nguyên, bạn sử dụng mẫu sau:

```
NUMBER(p)
```

Oracle cho phép thang đo âm, ví dụ: số sau sẽ làm tròn hàng trăm.

```
NUMBER(5,-2)
```

Lưu ý rằng nếu bạn thêm một số vào một cột NUMBER(p,s) và số đó vượt quá độ chính xác p, Oracle sẽ báo lỗi.

Ví dụ về kiểu dữ liệu NUMBER

Câu lệnh sau đây tạo một bảng có tên number_demo bao gồm một cột số:

```
CREATE TABLE number_demo (  
    number_value NUMERIC(6, 2)  
);
```

Các câu lệnh sau INSERT ba số vào number_demo:

```
INSERT INTO number_demo  
VALUES(100.99);  
  
INSERT INTO number_demo  
VALUES(90.551);  
  
INSERT INTO number_demo  
VALUES(87.556);
```

Trong ví dụ này:

- Giá trị đầu tiên được chèn thành công vì số này nằm trong phạm vi được xác định cho cột.
- Giá trị thứ hai được làm tròn xuống và giá trị thứ ba được làm tròn lên vì cột chỉ chấp nhận các số có hai chữ số thập phân.

Ví dụ sau chèn giá trị maximum và minimum mà number_value thể chấp nhận:

```
INSERT INTO number_demo  
VALUES(9999.99);  
  
INSERT INTO number_demo  
VALUES(-9999.99);
```

Ví dụ sau gây ra lỗi vì giá trị được chèn vượt quá độ chính xác được xác định cho cột.

```
INSERT INTO number_demo  
VALUES(-10000);
```

FLOAT

Giới thiệu về kiểu dữ liệu FLOAT

Kiểu dữ liệu Oracle FLOAT là kiểu con của kiểu dữ liệu NUMBER. Mục đích chính của nó là tạo điều kiện tương thích với các kiểu dữ liệu FLOAT trong ANSI SQL.

Sau đây cho thấy cú pháp của kiểu dữ liệu FLOAT:

```
FLOAT(p)
```

Bạn chỉ có thể chỉ định độ chính xác cho kiểu dữ liệu FLOAT. Bạn không thể chỉ định thang đo. Độ chính xác tối đa FLOAT là 126.

Ví dụ về FLOAT

Đầu tiên, tạo một bảng mới được đặt tên float_demo:

```
CREATE TABLE float_demo (  
    f1 FLOAT(1),  
    f2 FLOAT(4),  
    f3 FLOAT(7)  
);
```

Thứ hai, chèn một hàng mới vào float_demo:

```
INSERT  
    INTO  
        float_demo(  
            f1,  
            f2,  
            f3  
        )  
VALUES(  
    1 / 3, 1 / 3, 1 / 3  
);
```

Thứ ba, truy vấn dữ liệu từ float_demo:

```
SELECT  
    *  
FROM  
    float_demo;
```

F1	F2	F3
0.3	0.33	0.333

CHAR

Giới thiệu về kiểu dữ liệu Oracle CHAR

Kiểu dữ liệu CHAR cho phép bạn lưu trữ các chuỗi ký tự có độ dài cố định. Kiểu dữ liệu CHAR có thể lưu trữ một chuỗi ký tự có kích thước từ 1 đến 2000 byte.

Để xác định một cột CHAR, cần chỉ định độ dài chuỗi tính bằng byte hoặc ký tự như sau:

```
CHAR(length BYTE)
CHAR(length CHAR)
```

Nếu bạn không chỉ định rõ ràng BYTE hoặc CHAR tuân theo length, thì Oracle sẽ sử dụng BYTE theo mặc định.

Giá trị mặc định của length là 1 nếu bạn bỏ qua nó như ví dụ sau:

```
column_name CHAR
```

- Khi bạn chèn hoặc cập nhật cột chuỗi ký tự có độ dài cố định, Oracle sẽ lưu trữ các ký tự dưới dạng dữ liệu có độ dài cố định.
- Điều đó có nghĩa là nếu lưu trữ một giá trị có độ dài nhỏ hơn độ dài tối đa được xác định trong cột, Oracle sẽ đệm các khoảng trống vào chuỗi ký tự đến độ dài tối đa.
- Trong trường hợp bạn chèn một giá trị có độ dài lớn hơn cột, Oracle sẽ trả về lỗi.

NCHAR

Tổng quan về NCHAR

Kiểu dữ liệu NCHAR được sử dụng để lưu trữ dữ liệu ký tự Unicode có độ dài cố định. Bộ ký tự của NCHAR chỉ có thể là AL16UTF16 hoặc UTF8.

Khi bạn tạo bảng có một cột NCHAR thì kích thước tối đa của cột NCHAR luôn nằm trong độ dài ký tự, ví dụ:

```
CREATE TABLE nchar_demo (  
    description NCHAR(10)  
);
```

Trong ví dụ này, độ dài tối đa của cột description là 10 ký tự. Không thể sử dụng độ dài byte cho kích thước tối đa của các cột NCHAR như sau:

```
description NCHAR(10 BYTE) -- not possible
```

Oracle **NCHAR** vs. **CHAR**

Đầu tiên, kích thước tối đa của NCHAR chỉ có trong độ dài ký tự trong khi kích thước tối đa CHAR có thể ở độ dài ký tự hoặc byte.

Thứ hai, NCHAR lưu trữ các ký tự trong bộ ký tự mặc định quốc gia trong khi CHAR các ký tự lưu trữ trong bộ ký tự mặc định.

VARCHAR2

*Giới thiệu về kiểu dữ liệu **VARCHAR2***

Để lưu trữ các chuỗi ký tự có độ dài thay đổi, bạn sử dụng kiểu dữ liệu VARCHAR2. Một cột VARCHAR2 có thể lưu trữ một giá trị nằm trong khoảng từ 1 đến 4000 byte. Điều đó có nghĩa là đối với bộ ký tự một byte, bạn có thể lưu trữ tối đa 4000 ký tự trong một cột VARCHAR2.

Khi bạn tạo một bảng có một cột VARCHAR2, phải chỉ định độ dài chuỗi tối đa, tính bằng byte:

```
VARCHAR2(max_size BYTE)
```

hoặc bằng ký tự

```
VARCHAR2(max_size CHAR)
```

- Theo mặc định, Oracle sử dụng BYTE nếu không chỉ định rõ ràng BYTE hoặc CHAR sau max_size. Nói cách khác, một cột VARCHAR2(N) có thể chứa tối đa N byte ký tự.
- Nếu lưu trữ chuỗi ký tự có kích thước vượt quá kích thước tối đa của VARCHAR2, Oracle sẽ báo lỗi.

*Độ dài tối đa của **VARCHAR2***

Kể từ Cơ sở dữ liệu Oracle 12c, có thể chỉ định kích thước tối đa là 32767 cho kiểu dữ liệu VARCHAR2. Oracle sử dụng tham số MAX_STRING_SIZE để kiểm soát kích thước tối đa. Nếu MAX_STRING_SIZE là STANDARD, thì kích thước tối đa VARCHAR2 là 4000 byte. Trong trường hợp MAX_STRING_SIZE là EXTENDED, giới hạn kích thước VARCHAR2 là 32767.

*Ví dụ về Oracle **VARCHAR2***

Câu lệnh sau đây tạo một bảng mới có tên econtacts để lưu trữ các liên hệ khẩn cấp của nhân viên.

```
CREATE TABLE econtacts (  
    econtact_id NUMBER generated BY DEFAULT AS identity PRIMARY KEY,  
    employee_id NUMBER NOT NULL,  
    first_name VARCHAR2( 20 ) NOT NULL,  
    last_name VARCHAR2( 20 ) NOT NULL,  
    phone VARCHAR2( 12 ) NOT NULL,
```

```
FOREIGN KEY( employee_id ) REFERENCES employees( employee_id )
ON DELETE CASCADE
);
```

Bảng econtacts có ba cột VARCHAR2: first_name, last_name, và phone.

```
INSERT
  INTO
    econtacts(
      employee_id,
      first_name,
      last_name,
      phone
    )
  VALUES(
    1,
    'Branden',
    'Wiley',
    '202-555-0193'
  );
```

Nó hoạt động như mong đợi vì dữ liệu đầu vào không vượt quá kích thước tối đa của VARCHAR2.

Tuy nhiên, câu lệnh sau không chèn được:

```
INSERT
  INTO
    econtacts(
      employee_id,
      first_name,
      last_name,
      phone
    )
  VALUES(
    10,
    'Pablo Diego Jose Francisco',
    'Gray',
    '202-555-0195'
  );
```

Vì tên đầu vào vượt quá độ dài tối đa của first_name nên Oracle đã đưa ra lỗi sau:

```
SQL Error: ORA-12899: value too large for column "OT"."ECONTACTS"."FIRST_NAME"
(actual: 26, maximum: 20)
```

NVARCHAR2

Giới thiệu về kiểu dữ liệu NVARCHAR2

NVARCHAR2 là kiểu dữ liệu Unicode có thể lưu trữ các ký tự Unicode. Bộ ký tự của NVARCHAR2 là bộ ký tự quốc gia được chỉ định tại thời điểm tạo cơ sở dữ liệu.

Ví dụ về NVARCHAR2

Câu lệnh sau đây tạo một bảng có một cột NVARCHAR2 có độ dài tối đa là 50 ký tự.

```
CREATE TABLE nvarchar2_demo (  
    description NVARCHAR2(50)  
);
```

Câu lệnh sau chèn một hàng vào bảng nvarchar2_demo:

```
INSERT INTO nvarchar2_demo  
VALUES ('ABCDE');
```

VARCHAR2 vs NVARCHAR2

- Đầu tiên, kích thước tối đa của VARCHAR2 có thể bằng byte hoặc ký tự, trong khi kích thước tối đa NVARCHAR2 chỉ bằng ký tự. Ngoài ra, độ dài byte tối đa của một NVARCHAR2 phụ thuộc vào bộ ký tự quốc gia được định cấu hình.
 - Thứ hai, một cột VARCHAR2 chỉ có thể lưu trữ các ký tự trong bộ ký tự mặc định trong khi cột NVARCHAR2 có thể lưu trữ hầu như bất kỳ ký tự nào
-

DATE

Giới thiệu về kiểu dữ liệu DATE

- Kiểu dữ liệu DATE cho phép bạn lưu trữ các giá trị tại thời điểm bao gồm cả ngày và giờ với độ chính xác đến một giây.
- Kiểu dữ liệu DATE lưu trữ năm (bao gồm thế kỷ), tháng, ngày, giờ, phút và giây. Nó có phạm vi từ ngày 1 tháng 1 năm 4712 trước Công nguyên đến ngày 31 tháng 12 năm 9999 CN (sau Công nguyên hoặc 'AD').
- Cơ sở dữ liệu Oracle có định dạng riêng để lưu trữ dữ liệu ngày tháng. Nó sử dụng các trường có độ dài cố định là 7 byte, mỗi byte tương ứng với thế kỷ, năm, tháng, ngày, giờ, phút và giây để lưu trữ dữ liệu ngày tháng.

Định dạng ngày của Oracle

Định dạng ngày tiêu chuẩn cho đầu vào và đầu ra là DD-MON-YY ví dụ, 01-JAN-17 được điều khiển bởi giá trị của tham số NLS_DATE_FORMAT.

Câu lệnh sau đây hiển thị giá trị hiện tại của tham số NLS_DATE_FORMAT:

```
SELECT
  value
FROM
  V$NLS_PARAMETERS
WHERE
  parameter = 'NLS_DATE_FORMAT';
```

Trong hệ thống Cơ sở dữ liệu Oracle, giá trị của NLS_DATE_FORMAT là:

```
DD-MON-RR
```

Câu lệnh sau trả về ngày hiện tại với định dạng ngày tiêu chuẩn bằng cách sử dụng hàm SYSDATE.

```
SELECT
  sysdate
FROM
  dual;
```

Kết quả là:

```
01-NOV-23
```

Giả sử, các bạn muốn thay đổi định dạng ngày chuẩn thành YYYY-MM-DD, các bạn sử dụng câu lệnh ALTER SESSION để thay đổi giá trị của NLS_DATE_FORMAT như sau:

```
ALTER SESSION SET NLS_DATE_FORMAT = 'YYYY-MM-DD';
```


Định dạng ngày tháng bằng hàm TO_CHAR()

Hàm TO_CHAR() nhận một giá trị DATE, định dạng giá trị đó dựa trên định dạng đã chỉ định và trả về một chuỗi ngày tháng.

Ví dụ: để hiển thị ngày hệ thống hiện tại theo một định dạng cụ thể, bạn sử dụng TO_CHAR():

```
SELECT
  TO_CHAR( SYSDATE, 'FMMonth DD, YYYY' )
FROM
  dual;
```

Đầu ra là:

```
November 1, 2023
```

Chuyển đổi string thành date

Để chuyển đổi các giá trị ngày không ở định dạng chuẩn, bạn sử dụng hàm TO_DATE() có chuỗi định dạng.

Ví dụ sau chuyển đổi chuỗi 'August 01, 2017' thành ngày tương ứng bằng TO_DATE():

```
SELECT
  TO_DATE( 'August 01, 2017', 'MONTH DD, YYYY' )
FROM
  dual;
```

Giá trị đầu ra là:

```
01-AUG-17
```

Ví dụ về kiểu dữ liệu Oracle DATE

Câu lệnh sau đây tạo một bảng có tên my_events:

```
CREATE TABLE my_events (
  event_id NUMBER GENERATED BY DEFAULT AS IDENTITY,
  event_name VARCHAR2 ( 255 ) NOT NULL,
  location VARCHAR2 ( 255 ) NOT NULL,
  start_date DATE NOT NULL,
  end_date DATE NOT NULL,
  PRIMARY KEY ( event_id )
);
```

Trong bảng này, chúng ta có hai cột có kiểu dữ liệu DATE là start_date và end_date.

Để chèn một hàng mới vào my_events, sử dụng câu lệnh sau:

```
INSERT INTO my_events
    (event_name,
     location,
     start_date,
     end_date)
VALUES    ( 'TechEd Europe',
           'Barcelona, Spain',
           DATE '2017-11-14',
           DATE '2017-11-16' );
```

Trong ví dụ này, đã sử dụng hai ký tự ngày tháng trong INSERT.

Có thể sử dụng TO_DATE() để chuyển đổi một chuỗi thành một ngày trước khi chèn như trong ví dụ sau:

```
INSERT INTO my_events
    (event_name,
     location,
     start_date,
     end_date)
VALUES    ( 'Oracle OpenWorld',
           'San Francisco, CA, USA',
           TO_DATE( 'October 01, 2017', 'MONTH DD, YYYY' ),
           TO_DATE( 'October 05, 2017', 'MONTH DD, YYYY'));
```

TIMESTAMP

Giới thiệu về kiểu dữ liệu TIMESTAMP

- Kiểu dữ liệu **TIMESTAMP** cho phép bạn lưu trữ dữ liệu ngày và giờ bao gồm năm, tháng, ngày, giờ, phút và giây.
- Ngoài ra, nó còn lưu trữ các phân số giây, không được lưu trữ theo kiểu dữ liệu DATE.

Để xác định một cột TIMESTAMP, sử dụng cú pháp sau:

```
column_name TIMESTAMP[(fractional_seconds_precision)]
```

- Việc fractional_seconds_precision chỉ định số chữ số trong phần phân số của trường SECOND. Nó nằm trong khoảng từ 0 đến 9, nghĩa là bạn có thể sử dụng kiểu dữ liệu TIMESTAMP để lưu trữ tối đa nano giây.
- Nếu bạn bỏ qua fractional_seconds_precision, nó mặc định là 6.

Biểu thức sau đây minh họa cách xác định một cột TIMESTAMP:

```
...  
started_at TIMESTAMP(2),  
...
```

Trong ví dụ này, started_at là một TIMESTAMP có độ chính xác theo phân số giây được đặt thành micro giây.

TIMESTAMP literals

Để chỉ định TIMESTAMP literals, bạn sử dụng định dạng sau:

```
TIMESTAMP 'YYYY-MM-DD HH24:MI:SS.FF'
```

Ví dụ sau minh họa một TIMESTAMP:

```
TIMESTAMP '1999-12-31 23:59:59.10'
```

Ví dụ về TIMESTAMP

Đầu tiên, tạo một bảng mới có tên logs có một TIMESTAMP để minh họa.

```
CREATE TABLE logs (  
    log_id NUMBER GENERATED BY DEFAULT AS IDENTITY,  
    message VARCHAR(2) NOT NULL,  
    logged_at TIMESTAMP (2) NOT NULL,  
    PRIMARY KEY (log_id)
```

```
);
```

Thứ hai, chèn hàng mới vào bảng logs:

```
INSERT INTO logs (  
    message,  
    logged_at  
)  
VALUES (  
    'Invalid username/password for root user',  
    LOCALTIMESTAMP(2)  
);  
  
INSERT INTO logs (  
    message,  
    logged_at  
)  
VALUES (  
    'User root logged in successfully',  
    LOCALTIMESTAMP(2)  
);
```

Trong ví dụ này, chúng tôi lấy dấu thời gian cục bộ hiện tại với độ chính xác phân số giây lên đến micro giây từ hàm LOCALTIMESTAMP(2) và chèn giá trị đó vào logged_at của bảng logs.

Thứ ba, truy vấn dữ liệu TIMESTAMP từ logs bảng:

```
SELECT log_id,  
       message,  
       logged_at  
FROM logs;
```

	LOG_ID	MESSAGE	LOGGED_AT
1	1	Invalid username/password for root user	03-AUG-17 10.39.09.730000000 AM
2	2	User root logged in successfully	03-AUG-17 10.39.11.240000000 AM

Định dạng giá trị TIMESTAMP

Để thay đổi kết quả đầu ra của một giá trị TIMESTAMP, sử dụng TO_CHAR() bằng cách chuyển tên của giá trị TIMESTAMP hoặc cột làm đối số đầu tiên và chuỗi định dạng làm đối số thứ hai.

```
SELECT message,  
       TO_CHAR(logged_at, 'MONTH DD, YYYY "at" HH24:MI')  
FROM logs;
```

Hình ảnh dưới đây minh họa đầu ra:

	MESSAGE	TO_CHAR(LOGGED_AT,'MONTHDD,YYYY"AT"HH24:MI')
1	Invalid username/password for root user	AUGUST 03, 2017 at 10:39
2	User root logged in successfully	AUGUST 03, 2017 at 10:39

Trích xuất các thành phần TIMESTAMP

Để trích xuất các thành phần TIMESTAMP như năm, tháng, ngày, giờ, phút, giây sử dụng hàm EXTRACT():

```
EXTRACT( component FROM timestamp);
```

Xem ví dụ:

```
SELECT
    message,
    EXTRACT(year FROM logged_at) year,
    EXTRACT(month FROM logged_at) month,
    EXTRACT(day FROM logged_at) day,
    EXTRACT(hour FROM logged_at) hour,
    EXTRACT(minute FROM logged_at) minute,
    EXTRACT(second FROM logged_at) second
FROM
    logs;
```

Section 12. Constraints

Primary key

Giới thiệu về khóa chính - Primary key

Khóa chính là một cột gồm nhiều cột trong bảng xác định duy nhất một hàng trong bảng.

Sau đây là các quy tắc đặt một cột làm khóa chính:

- Cột khóa chính không thể chứa giá trị NULL hoặc chuỗi trống.
- Giá trị khóa chính phải là duy nhất trong toàn bộ bảng.
- Giá trị khóa chính không được thay đổi theo thời gian.

Để tạo primary key trong bảng, bạn sử dụng PRIMARY KEY.

Thông thường, bạn primary key cho một bảng khi tạo bảng đó. Ngoài ra, bạn có thể thêm primary key vào bảng sau bằng cách sử dụng ALTER TABLE.

Tạo primary key bao gồm một cột

Câu lệnh sau đây sử dụng CREATE TABLE tạo bảng purchase_orders:

```
CREATE TABLE purchase_orders (  
    po_nr NUMBER PRIMARY KEY,  
    vendor_id NUMBER NOT NULL,  
    po_status NUMBER(1,0) NOT NULL,  
    created_at TIMESTAMP WITH TIME ZONE NOT NULL  
);
```

Bảng purchase_orders có bốn cột số đơn đặt hàng (po_nr), id nhà cung cấp (vendor_id), trạng thái đơn đặt hàng (po_status) và thời gian (created_at) mà đơn đặt hàng được tạo.

Trong bảng này, xác định cột po_nr là primary key bằng cách sử dụng mệnh đề PRIMARY KEY.

Hãy xem xét câu lệnh sau đây:

```
CREATE TABLE purchase_orders (  
    po_nr NUMBER,  
    vendor_id NUMBER NOT NULL,  
    po_status NUMBER(1,0) NOT NULL,  
    created_at TIMESTAMP WITH TIME ZONE NOT NULL,  
    CONSTRAINT pk_purchase_orders PRIMARY KEY(po_nr)  
);
```

Ngoài ra, chúng tôi đã gán rõ ràng PRIMARY KEY cho một tên là pk_purchase_orders.

Tạo primary key bao gồm nhiều cột

Câu lệnh sau đây tạo bảng chi tiết đơn hàng của đơn đặt hàng:

```
CREATE TABLE purchase_order_items (  
    po_nr NUMBER NOT NULL,  
    item_nr NUMBER NOT NULL,  
    product_id NUMBER NOT NULL,  
    quantity NUMBER NOT NULL,  
    purchase_unit NUMBER NOT NULL,  
    buy_price NUMBER (9,2) NOT NULL,  
    delivery_date DATE,  
    PRIMARY KEY (po_nr, item_nr)  
);
```

Trong ví dụ này, primary key của purchase_order_items bao gồm hai cột: po_nr và item_nr. Điều đó có nghĩa là sự kết hợp các giá trị của các cột này xác định duy nhất một chi tiết đơn hàng của đơn đặt hàng.

Thêm primary key vào bảng

Đôi khi, bạn có thể muốn thêm ràng buộc primary key vào bảng hiện có. Để làm điều đó, bạn sử dụng ALTER TABLE như sau:

```
ALTER TABLE table_name  
ADD CONSTRAINT constraint_name  
PRIMARY KEY (column1, column2, ...);
```

Ví dụ sau tạo bảng vendors trước rồi thêm ràng buộc primary key vào bảng:

```
CREATE TABLE vendors (  
    vendor_id NUMBER,  
    vendor_name VARCHAR2(255) NOT NULL,  
    address VARCHAR2(255) NOT NULL  
);  
  
ALTER TABLE vendors  
ADD CONSTRAINT pk_vendors PRIMARY KEY (vendor_id);
```

Loại bỏ PRIMARY KEY

```
ALTER TABLE table_name  
DROP CONSTRAINT primary_key_constraint_name;
```

Ví dụ: bạn có thể loại bỏ primary key của bảng vendors như sau:

```
ALTER TABLE vendors  
DROP CONSTRAINT pk_vendors;
```

Có thể sử dụng câu lệnh sau để loại bỏ primary key của bảng:

```
ALTER TABLE vendors  
DROP PRIMARY KEY;
```

Enable / Disable an Oracle PRIMARY KEY constraint

Để cải thiện hiệu suất khi tải một lượng lớn dữ liệu vào bảng hoặc cập nhật dữ liệu hàng loạt, có thể tạm thời vô hiệu hóa PRIMARY KEY.

Để vô hiệu hóa một PRIMARY KEY của bảng, bạn sử dụng ALTER TABLE:

```
ALTER TABLE table_name  
DISABLE CONSTRAINT primary_key_constraint_name;
```

hoặc

```
ALTER TABLE table_name  
DISABLE PRIMARY KEY;
```

Ví dụ: để vô hiệu hóa ràng buộc primary key của bảng purchase_orders, bạn sử dụng câu lệnh sau:

```
ALTER TABLE purchase_orders  
DISABLE CONSTRAINT pk_purchase_orders;
```

hoặc

```
ALTER TABLE purchase_orders  
DISABLE PRIMARY KEY;
```

⇒ **Enable thì làm ngược lại**

Foreign key

Giới thiệu về foreign key

Giả sử chúng ta có hai bảng `supplier_groups` và `suppliers`:

```
CREATE TABLE supplier_groups(  
    group_id NUMBER GENERATED BY DEFAULT AS IDENTITY,  
    group_name VARCHAR2(255) NOT NULL,  
    PRIMARY KEY (group_id)  
);  
  
CREATE TABLE suppliers (  
    supplier_id NUMBER GENERATED BY DEFAULT AS IDENTITY,  
    supplier_name VARCHAR2(255) NOT NULL,  
    group_id NUMBER NOT NULL,  
    PRIMARY KEY(supplier_id)  
);
```

- Bảng `supplier_groups` lưu trữ các nhóm nhà cung cấp, ví dụ: nhà cung cấp một lần, nhà cung cấp bên thứ ba và nhà cung cấp liên doanh. Mỗi nhóm nhà cung cấp có thể không có, một hoặc nhiều nhà cung cấp.
- Bảng `suppliers` lưu trữ thông tin nhà cung cấp. Mỗi nhà cung cấp phải thuộc về một nhóm nhà cung cấp.
- Mỗi quan hệ giữa `supplier_groups` và `suppliers` là một-nhiều. Nói cách khác, một nhóm nhà cung cấp có nhiều nhà cung cấp trong đó mỗi nhà cung cấp phải thuộc một nhóm nhà cung cấp.

Trong `group_id` bảng `suppliers` được sử dụng để thiết lập mối quan hệ giữa các hàng trong bảng `suppliers` và `supplier_groups`.

Trước khi chèn một hàng vào bảng `suppliers`, bạn phải tra cứu một hàng hiện có `group_id` trong bảng `supplier_groups` và sử dụng giá trị này để insert.

Create a foreign key

Câu lệnh sau minh họa cú pháp tạo foreign key:

```
CREATE TABLE child_table (  
    ...  
    CONSTRAINT fk_name  
    FOREIGN KEY(col1, col2,...) REFERENCES parent_table(col1,col2)  
    ON DELETE [ CASCADE | SET NULL ]  
);
```

Hãy xem xét câu lệnh một cách chi tiết.

- Đầu tiên, để gán tên rõ ràng cho **foreign key**, sử dụng mệnh đề CONSTRAINT theo sau là tên. CONSTRAINT là tùy chọn.
- Thứ hai, chỉ định mệnh đề FOREIGN KEY để xác định một hoặc nhiều cột làm **foreign key** và bảng cha với các cột mà cột **foreign key** tham chiếu đến.
- Thứ ba, sử dụng mệnh đề ON DELETE này để xác định hậu quả khi các hàng trong bảng cha bị xóa.
 - ON DELETE CASCADE: nếu một hàng trong bảng cha bị xóa thì tất cả các hàng trong bảng con tham chiếu đến hàng bị xóa cũng sẽ bị xóa.
 - ON DELETE SET NULL: nếu một hàng trong bảng cha bị xóa thì tất cả các hàng trong bảng con tham chiếu đến hàng bị xóa sẽ được đặt thành NULL cho các cột **foreign key**.

Không giống như primary key, một bảng có thể có nhiều foreign key.

Thêm foreign key vào bảng

Nếu bạn muốn thêm foreign key vào bảng hiện có, bạn sử dụng ALTER TABLE như sau:

```
ALTER TABLE child_table
ADD CONSTRAINT fk_name
FOREIGN KEY (col1,col2) REFERENCES parent_table(col1,col2);
```

Bỏ ràng buộc khóa ngoại

Để loại bỏ foreign key, sử dụng câu lệnh ALTER TABLE bên dưới:

```
ALTER TABLE child_table
DROP CONSTRAINT fk_name;
```

Vô hiệu hóa foreign key

Để tạm thời vô hiệu hóa foreign key, sử dụng ALTER TABLE câu lệnh sau:

```
ALTER TABLE child_table
DISABLE CONSTRAINT fk_name;
```

Kích hoạt một foreign key

Tương tự, cũng sử dụng câu lệnh ALTER TABLE để kích hoạt foreign key bị vô hiệu hóa:

```
ALTER TABLE child_table
ENABLE CONSTRAINT fk_name;
```

UNIQUE constraint

Cú pháp UNIQUE

UNIQUE là ràng buộc toàn vẹn nhằm đảm bảo dữ liệu được lưu trữ trong một cột hoặc một nhóm cột là duy nhất giữa các hàng trong bảng.

Thông thường, bạn áp dụng các UNIQUE cho các cột khi tạo bảng bằng cú pháp như sau:

```
CREATE TABLE table_name (  
    ...  
    column_name data_type UNIQUE  
    ...  
);
```

UNIQUE này xác định rằng các giá trị trong cột column_name toàn bộ bảng là duy nhất.

Cũng có thể sử dụng như sau:

```
CREATE TABLE table_name (  
    ...,  
    UNIQUE(column_name)  
);
```

Hoặc

```
CREATE TABLE table_name (  
    ...  
    column_name data_type CONSTRAINT unique_constraint_name UNIQUE  
    ...  
);  
  
CREATE TABLE table_name (  
    ...  
    column_name data_type,  
    ...,  
    CONSTRAINT unique_constraint_name UNIQUE(column_name)  
);
```

Để xác định một UNIQUE cho nhiều cột, bạn sử dụng cú pháp ràng buộc ngoài dòng:

```
CREATE TABLE table_name (  
    ...  
    column_name1 data_type,  
    column_name2 data_type,  
    ...,
```

```
CONSTRAINT unique_constraint_name UNIQUE(column_name1, column_name2)
);
```

Nếu bạn muốn thêm một UNIQUE vào một bảng hiện có, bạn sử dụng ALTER TABLE:

```
ALTER TABLE table_name
ADD CONSTRAINT unique_constraint_name UNIQUE(column_name1, column_name2);
```

Đôi khi, bạn có thể muốn tạm thời vô hiệu hóa một UNIQUE:

```
ALTER TABLE table_name
DISABLE CONSTRAINT unique_constraint_name;
```

Và sau đó kích hoạt nó:

```
ALTER TABLE table_name
ENABLE CONSTRAINT unique_constraint_name;
```

Hoặc thậm chí loại bỏ một UNIQUE:

```
ALTER TABLE table_name
DROP CONSTRAINT unique_constraint_name;
```

Ví dụ UNIQUE của Oracle

```
CREATE TABLE clients (
  client_id NUMBER GENERATED BY DEFAULT AS IDENTITY,
  first_name VARCHAR2(50) NOT NULL,
  last_name VARCHAR2(50) NOT NULL,
  company_name VARCHAR2(255) NOT NULL,
  email VARCHAR2(255) NOT NULL UNIQUE,
  phone VARCHAR(25)
);
```

Cột email có một UNIQUE để đảm bảo sẽ không có email trùng lặp.

Câu lệnh sau insert một hàng vào bảng clients:

```
INSERT INTO clients(first_name,last_name, email, company_name, phone)
VALUES('Christene','Snider','christene.snider@abc.com', 'ABC Inc', '408-875-6075');
```

Bây giờ, chúng tôi cố gắng thêm một hàng mới có giá trị email đã tồn tại trong cột email:

```
INSERT INTO clients(first_name,last_name, email, company_name, phone)
VALUES('Sherly','Snider','christene.snider@abc.com', 'ABC Inc', '408-875-6076');
```

Oracle đã đưa ra thông báo lỗi sau cho biết UNIQUE đã bị vi phạm:

```
SQL Error: ORA-00001: unique constraint (OT.SYS_C0010726) violated
```