Lập Trình PL/SQL
Dữ liệu (Data)

Người trình bày:

Đoàn Thị Kim Nhung

ITBA Team

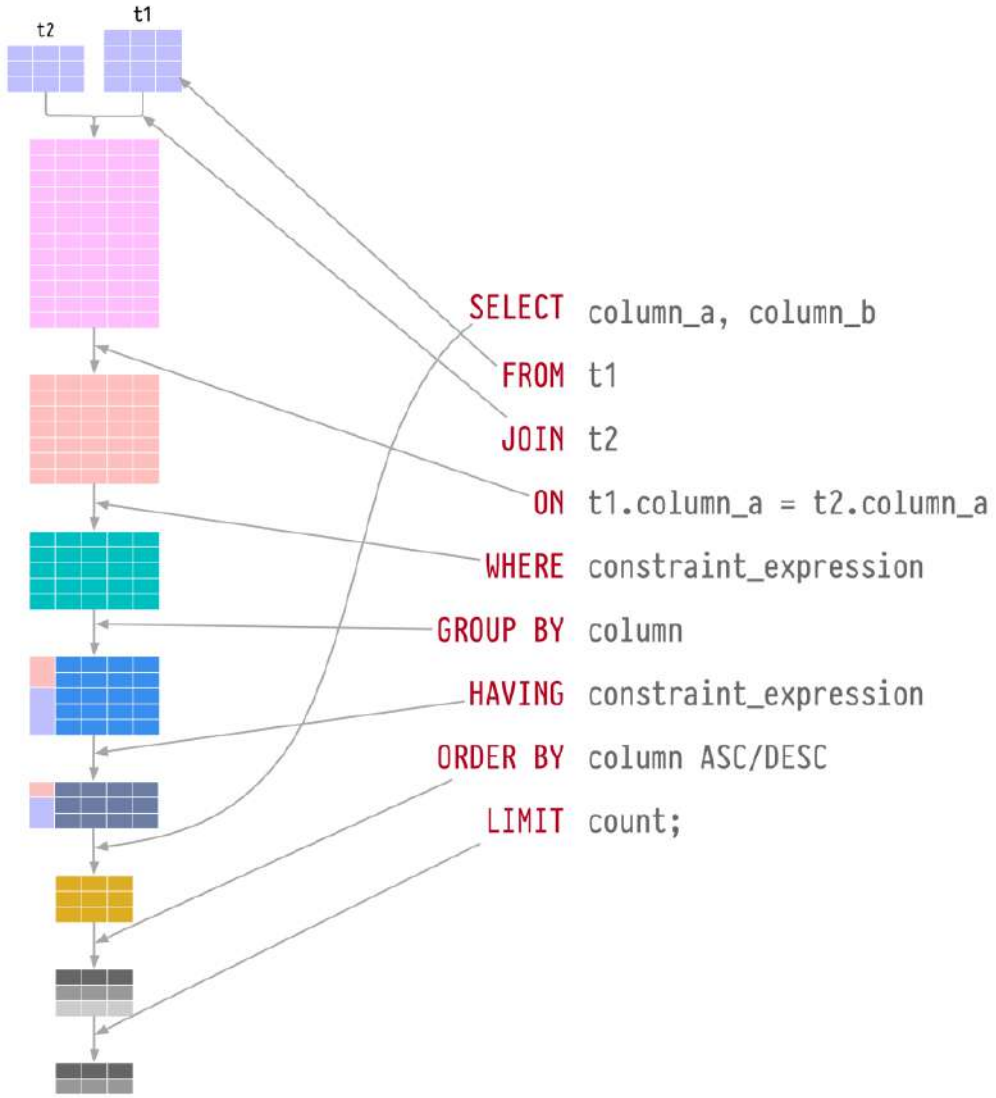
Mục Lục

BATIZENS

01

Lọc Dữ Liệu

Trình Tự Logic Lọc Dữ Liệu



1	FROM	Tên bảng (1 bảng – t1)
2	JOIN	Tên bảng (1 bảng – t2)
3	ON	Liên kết 2 bảng t1 và t2
4	WHERE	Điều kiện
5	GROUP BY	Các thuộc tính gom nhóm
6	WITH CUBE	Tính tổng phụ cho mọi thuộc tính
7	WITH ROLLUP	Tính tổng phụ cho một thuộc tính
8	HAVING	Điều kiện sau khi gom nhóm
9	SELECT	Hiển thị dữ liệu
10	ORDER BY	Sắp xếp các thuộc tính

1. FROM

/* Method 1: Không có ALIAS */

FROM <TABLE_NAME_1>

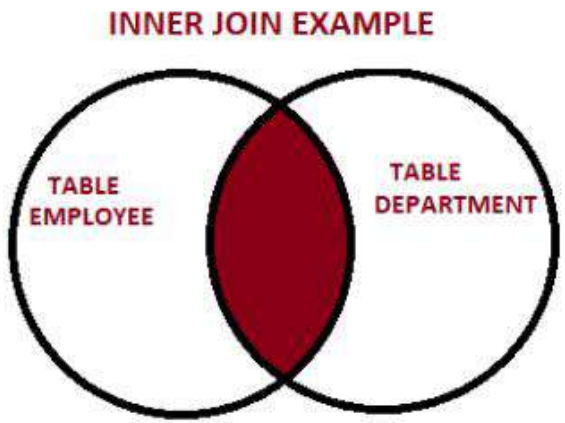
/* Method 2: Có ALIAS */

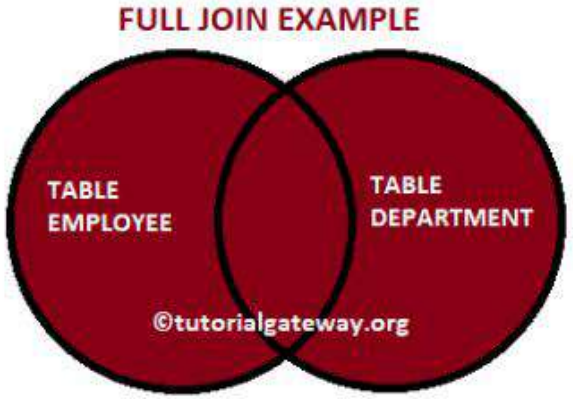
FROM <TABLE_NAME_1> <ALIAS_TABLE_NAME>

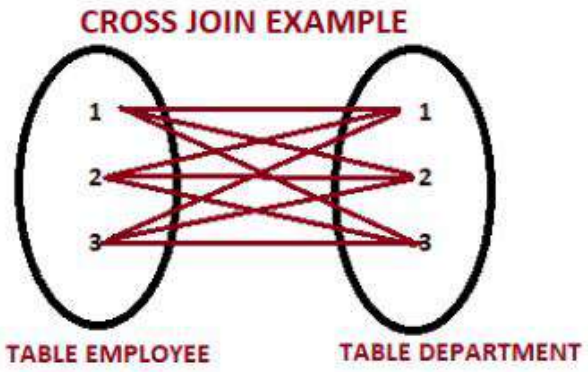
FROM	Cú pháp để lấy ra thông tin bảng cần xem thông tin (bảng 1)
ALIAS	Tên tạm thời
TABLE_NAME_1	Tên bảng đã tạo

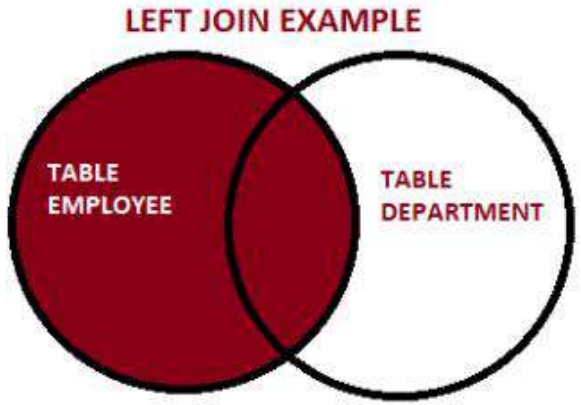
2. JOIN

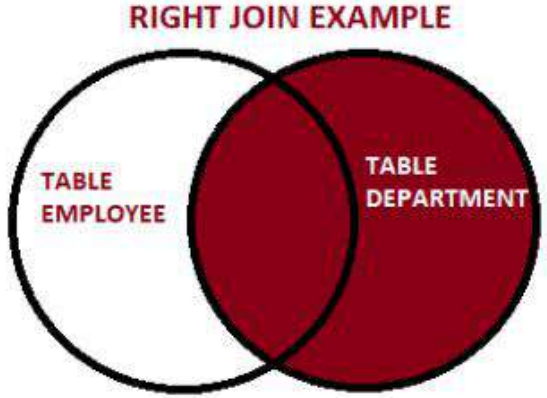
JOIN: Lấy ra bảng sẽ liên kết với bảng từ câu lệnh FROM. JOIN gồm có 6 loại dưới đây:

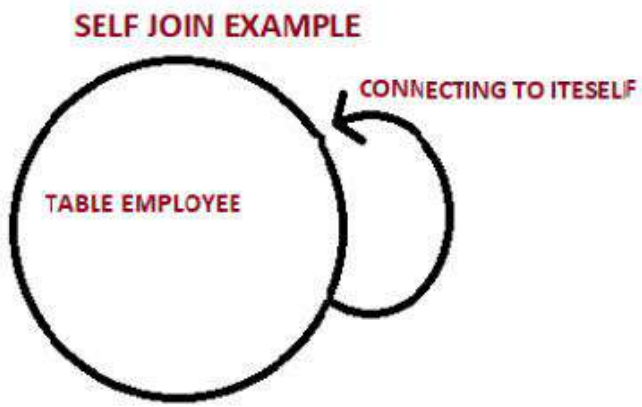








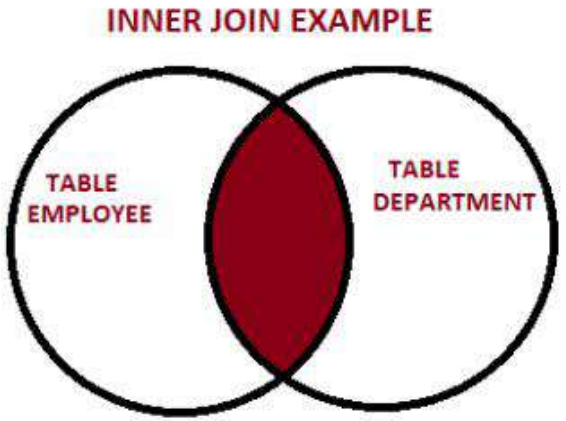




2. JOIN

1. INNER JOIN

Trả về các bản ghi có giá trị khớp trong cả hai bảng dữ liệu.




```
/* Method 1: Không có ALIAS */

INNER JOIN <TABLE_NAME_2>

/* Method 2: Có ALIAS */

INNER JOIN <TABLE_NAME_2> <ALIAS_TABLE_NAME>
```

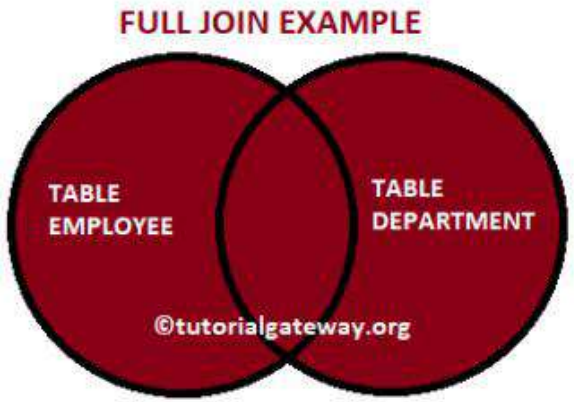
INNER JOIN	Cú pháp để lấy ra thông tin bảng cần xem thông tin (bảng 2)
ALIAS	Tên tạm thời
TABLE_NAME_2	Tên bảng đã tạo

Chú thích:
1. TABLE EMPLOYEE (bảng 1) từ câu lệnh FROM.
2. TABLE DEPARTMENT (bảng 2) từ câu lệnh INNER JOIN.

2. JOIN

2. FULL JOIN (FULL OUTER JOIN)

Trả về tất cả các bản ghi từ cả hai bảng, kết hợp cả bản ghi khớp và không khớp.



/* Method 1: Không có ALIAS */

FULL (OUTER) JOIN <TABLE_NAME_2>

/* Method 2: Có ALIAS */

FULL (OUTER) JOIN <TABLE_NAME_2> <ALIAS_TABLE_NAME>

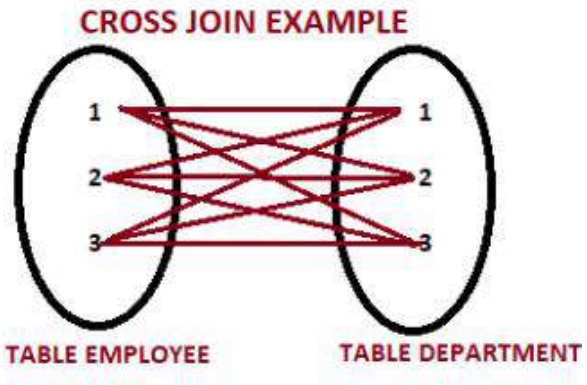
FULL (OUTER) JOIN	Cú pháp để lấy ra thông tin bảng cần xem thông tin (bảng 2)
ALIAS	Tên tạm thời
TABLE_NAME_2	Tên bảng đã tạo

Chú thích:
1. TABLE EMPLOYEE (bảng 1) từ câu lệnh FROM.
2. TABLE DEPARTMENT (bảng 2) từ câu lệnh FULL (OUTER) JOIN.

2. JOIN

3. CROSS JOIN

Tạo ra một kết hợp mọi cặp dữ liệu từ hai bảng, không cần điều kiện nối.




```
/* Method 1: Không có ALIAS */

CROSS JOIN <TABLE_NAME_2>

/* Method 2: Có ALIAS */

CROSS JOIN <TABLE_NAME_2> <ALIAS_TABLE_NAME>
```

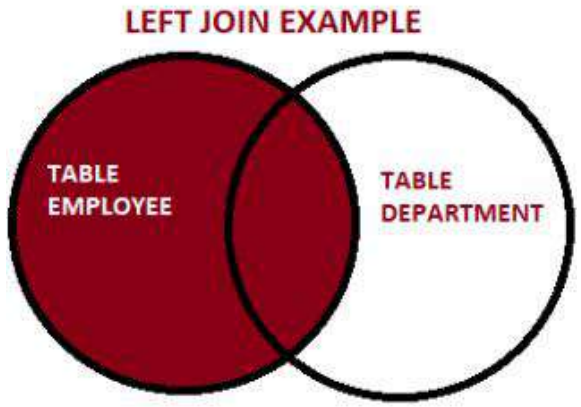
CROSS JOIN	Cú pháp để lấy ra thông tin bảng cần xem thông tin (bảng 2)
ALIAS	Tên tạm thời
TABLE_NAME_2	Tên bảng đã tạo

Chú thích:
1. TABLE EMPLOYEE (bảng 1) từ câu lệnh FROM.
2. TABLE DEPARTMENT (bảng 2) từ câu lệnh CROSS JOIN.

2. JOIN

4. LEFT JOIN (LEFT OUTER JOIN)

Trả về tất cả các bản ghi từ bảng bên trái (t1), và các bản ghi từ bảng bên phải (t2) mà khớp với điều kiện, nếu không có bản ghi khớp thì các cột từ bảng bên phải sẽ có giá trị NULL.



/* Method 1: Không có ALIAS */

LEFT (OUTER) JOIN <TABLE_NAME_2>

/* Method 2: Có ALIAS */

LEFT (OUTER) JOIN <TABLE_NAME_2> <ALIAS_TABLE_NAME>

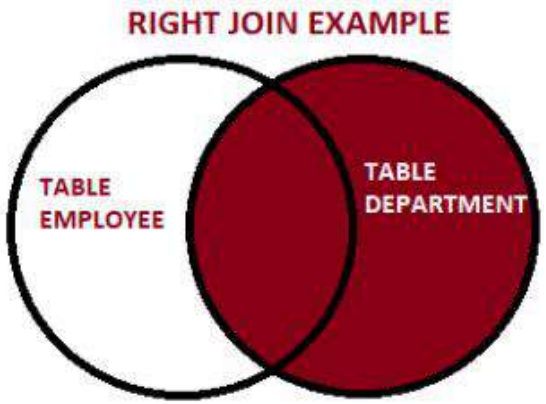
LEFT (OUTER) JOIN	Cú pháp để lấy ra thông tin bảng cần xem thông tin (bảng 2)
ALIAS	Tên tạm thời
TABLE_NAME_2	Tên bảng đã tạo

Chú thích:
1. TABLE EMPLOYEE (bảng 1) từ câu lệnh FROM.
2. TABLE DEPARTMENT (bảng 2) từ câu lệnh LEFT (OUTER) JOIN.

2. JOIN

5. RIGHT JOIN (RIGHT OUTER JOIN)

Trả về tất cả các bản ghi từ bảng bên phải (t2), và các bản ghi từ bảng bên trái (t1) mà khớp với điều kiện, nếu không có bản ghi khớp thì các cột từ bảng bên phải sẽ có giá trị NULL.




```
/* Method 1: Không có ALIAS */

RIGHT (OUTER) JOIN <TABLE_NAME_2>

/* Method 2: Có ALIAS */

RIGHT (OUTER) JOIN <TABLE_NAME_2> <ALIAS_TABLE_NAME>
```

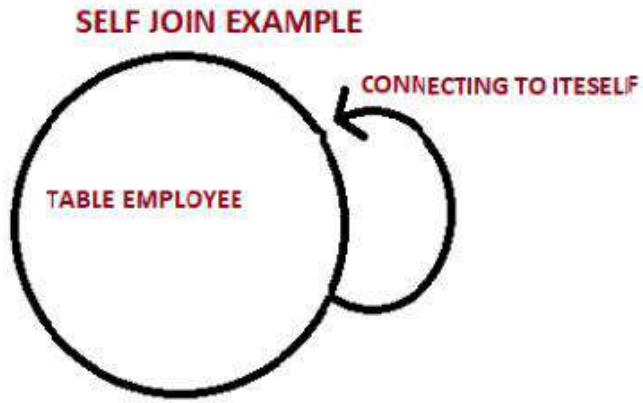
RIGHT (OUTER) JOIN	Cú pháp để lấy ra thông tin bảng cần xem thông tin (bảng 2)
ALIAS	Tên tạm thời
TABLE_NAME_2	Tên bảng đã tạo

Chú thích:
1. TABLE EMPLOYEE (bảng 1) từ câu lệnh FROM.
2. TABLE DEPARTMENT (bảng 2) từ câu lệnh RIGHT (OUTER) JOIN.

2. JOIN

6. JOIN (SELF JOIN)

thực hiện kết nối giữa hai phiên bản (hay còn gọi là hai "bản thể") của cùng được sử dụng khi bạn muốn so sánh các dòng trong cùng một bảng dữ liệu.



/* Method 1: Không có ALIAS */

(SELF) JOIN <TABLE_NAME_2>

/* Method 2: Có ALIAS */

(SELF) JOIN <TABLE_NAME_2> <ALIAS_TABLE_NAME>

(SELF) JOIN	Cú pháp để lấy ra thông tin bảng cần xem thông tin (bảng 2)
ALIAS	Tên tạm thời
TABLE_NAME_2	Tên bảng đã tạo

Chú thích:
1. TABLE EMPLOYEE (bảng 1) từ câu lệnh FROM.
2. TABLE EMPLOYEE (bảng 2) từ câu lệnh (SELF) JOIN.

3. ON

/* Method 1: Không có ALIAS */

ON <COLUMN_NAME_ID_T1> = <COLUMN_NAME_ID_T2>

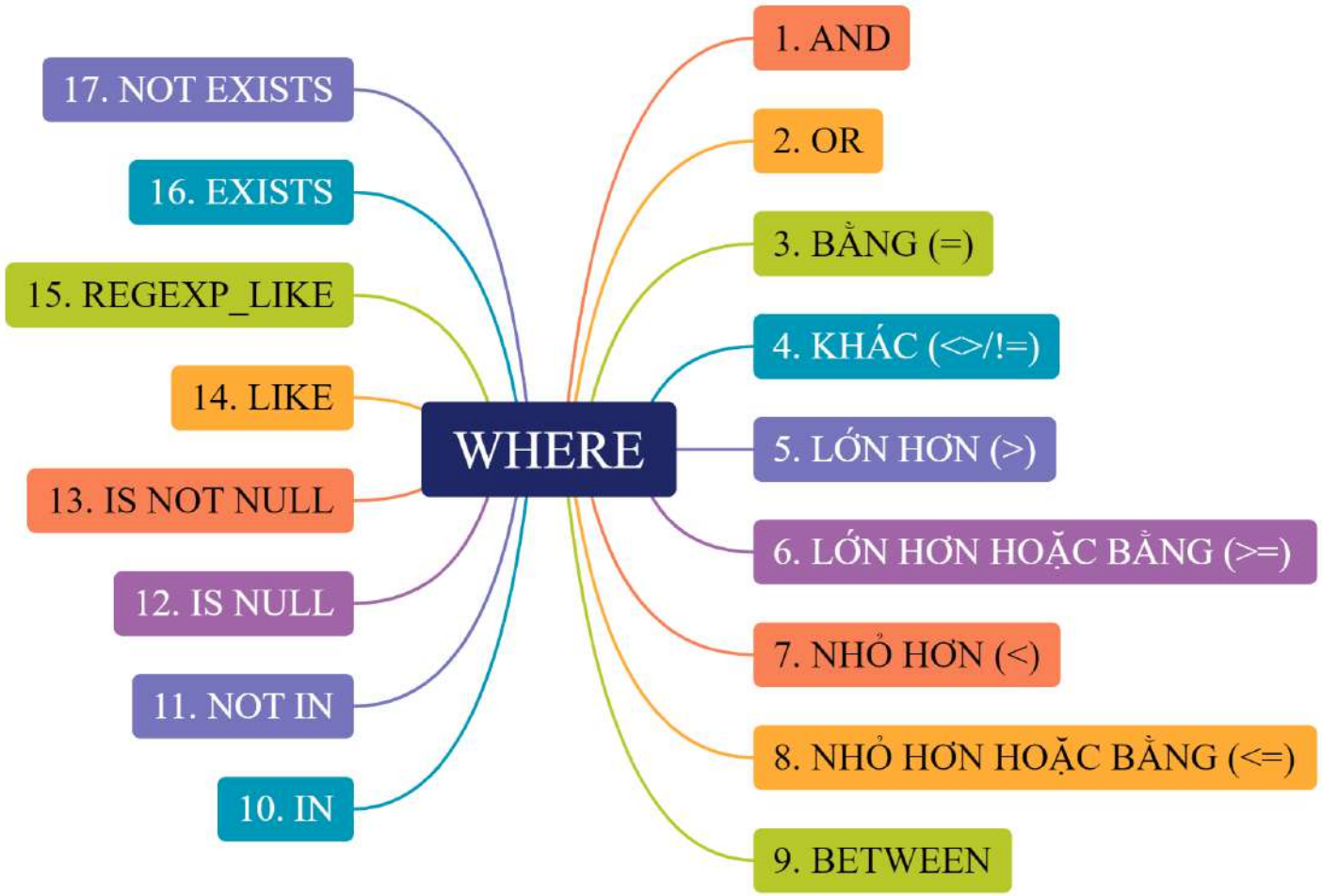
/* Method 2: Có ALIAS */

ON <ALIAS_T1>.<COLUMN_NAME_ID_T1> = <ALIAS_T2>.<COLUMN_NAME_ID_T2>

ON	Cú pháp để liên kết bảng 1 (t1) và bảng 2 (t2) thành 1 bảng
ALIAS	Tên tạm thời
COLUMN_NAME_ID_T1 COLUMN_NAME_ID_T2	Giá trị tại 2 trường thông tin này sẽ trùng nhau (khóa ngoại).

4. WHERE

WHERE: Truy vấn dữ liệu theo các tiêu chí nhất định và thực hiện các thao tác chỉnh sửa dữ liệu dựa trên các điều kiện này. Các toán tử được sử dụng trong câu lệnh WHERE gồm:



4. WHERE

WHERE: Truy vấn dữ liệu theo các tiêu chí nhất định và thực hiện các thao tác chỉnh sửa dữ liệu dựa trên các điều kiện này. Các toán tử được sử dụng trong câu lệnh WHERE gồm:

```
/* Method 1: Không có ALIAS */

WHERE <WHERE_CONDITION>

/* Method 2: Có ALIAS */

WHERE <WHERE_CONDITION>
```

WHERE	Cú pháp để thực hiện các điều kiện so sánh
ALIAS	Tên tạm thời

4. WHERE

STT	Toán tử	Mô tả
1	AND	Toán tử logic được sử dụng để kết hợp các điều kiện trong câu lệnh WHERE, UPDATE, DELETE, và các lệnh khác. Toán tử AND yêu cầu tất cả điều kiện đều phải đúng để một hàng hoặc một giá trị được chấp nhận >> Cùng TRUE hoặc cùng FALSE.
2	OR	Toán tử logic được sử dụng để kết hợp các điều kiện trong câu lệnh WHERE, UPDATE, DELETE, và các lệnh khác. Toán tử OR yêu cầu một trong các điều kiện phải đúng để một hàng hoặc một giá trị được chấp nhận >> Một trong các điều kiện TRUE thì TRUE, một trong các điều kiện FALSE thì FALSE.
3	=	Toán tử so sánh, dùng để kiểm tra xem giá trị bên trái có bằng bên phải hay không. Kết quả của phép so sánh này là một giá trị Boolean (TRUE hoặc FALSE).
4	<> / !=	Toán tử so sánh, dùng để kiểm tra xem giá trị bên trái có khác bên phải hay không. Kết quả của phép so sánh này là một giá trị Boolean (TRUE hoặc FALSE).
5	>	Toán tử so sánh, dùng để kiểm tra xem giá trị bên trái có lớn hơn giá trị bên phải hay không. Kết quả của phép so sánh này là một giá trị Boolean (TRUE hoặc FALSE).
6	>=	Toán tử so sánh, dùng để kiểm tra xem giá trị bên trái có lớn hơn hoặc bằng giá trị bên phải hay không. Kết quả của phép so sánh này là một giá trị Boolean (TRUE hoặc FALSE).

4. WHERE

STT	Toán tử	Mô tả
7	<	Toán tử so sánh, dùng để kiểm tra xem giá trị bên trái có nhỏ hơn giá trị bên phải hay không. Kết quả của phép so sánh này là một giá trị Boolean (TRUE hoặc FALSE).
8	<=	Toán tử so sánh, dùng để kiểm tra xem giá trị bên trái có nhỏ hơn hoặc bằng giá trị bên phải hay không. Kết quả của phép so sánh này là một giá trị Boolean (TRUE hoặc FALSE).
9	BETWEEN … AND	Toán tử BETWEEN được sử dụng để kiểm tra xem một giá trị có nằm trong một đoạn giá trị xác định hay không.
10	IN ()	Toán tử IN kiểm tra xem một giá trị có tồn tại trong một tập hợp giá trị nào đó hay không. Nếu giá trị đó nằm trong tập hợp, biểu thức sẽ là TRUE, ngược lại là FALSE.
11	NOT IN ()	Toán tử NOT IN kiểm tra xem một giá trị có nằm ngoài một tập hợp giá trị nào đó hay không. Nếu giá trị đó không nằm trong tập hợp, biểu thức sẽ là TRUE, ngược lại là FALSE.
12	IS NULL	Toán tử IS NULL được sử dụng để kiểm tra xem một giá trị có phải là NULL hay không. Nếu giá trị đó là NULL, biểu thức sẽ là TRUE, ngược lại là FALSE.
13	IS NOT NULL	Toán tử IS NOT NULL được sử dụng để kiểm tra xem một giá trị có phải là không NULL hay không. Nếu giá trị đó không phải là NULL, biểu thức sẽ là TRUE, ngược lại là FALSE.

4. WHERE

STT	Toán tử	Mô tả
14	LIKE	Toán tử LIKE được sử dụng để so sánh một giá trị chuỗi với một mẫu chuỗi có thể chứa các ký tự đại diện (‘%’ và ‘_’). ‘%’ đại diện cho bất kỳ chuỗi ký tự nào (bao gồm cả chuỗi rỗng). ‘_’ đại diện cho 1 ký tự đơn. UPPER (<COLUMN_NAME>) LIKE <‘%’>: Chuyển đổi sang chữ hoa. LOWER (<COLUMN_NAME>) LIKE <‘%’>: Chuyển đổi sang chữ thường. Trường hợp muốn tìm kiếm ký tự đặc biệt như: %, _, \ <COLUMN_NAME> LIKE ‘%\%%’ ESCAPE ‘\’: Tìm kiếm chuỗi có ký tự %. <COLUMN_NAME> LIKE ‘%_%’ ESCAPE ‘\’: Tìm kiếm chuỗi có ký tự _. <COLUMN_NAME> LIKE ‘%\\%’ ESCAPE ‘\’: Tìm kiếm chuỗi có ký tự \.
15	REGEXP_LIKE	Nó hỗ trợ các biểu thức chính quy, cho phép tạo ra các mẫu phức tạp hơn LIKE. Tuy nhiên, REGEXP_LIKE có thể chậm hơn LIKE đối với các mẫu đơn giản do sự phức tạp của biểu thức chính quy.
16	EXISTS (subquery)	Được sử dụng để kiểm tra sự tồn tại của các hàng trong một truy vấn con (Subquery). Toán tử EXISTS trả về TRUE nếu truy vấn con trả về ít nhất một hàng. Ngược lại, nó trả về FALSE nếu truy vấn con không trả về hàng nào.
17	NOT EXISTS (subquery)	Được sử dụng để kiểm tra sự tồn tại của các hàng trong một truy vấn con (Subquery). Toán tử NOT EXISTS trả về TRUE nếu truy vấn con không trả về hàng nào. Ngược lại, nó trả về FALSE nếu truy vấn con trả về ít nhất một hàng.

5. GROUP BY

/* Method 1: Không có ALIAS */

GROUP BY
<COLUMN_NAME_1>, <COLUMN_NAME_2>, ..., <COLUMN_NAME_N>

/* Method 2: Có ALIAS */

GROUP BY
<ALIAS_1>.<COLUMN_NAME_1>, <ALIAS_2>.<COLUMN_NAME_2>, ..., <ALIAS_N>.<COLUMN_NAME_N>

GROUP BY	Được sử dụng để nhóm các hàng có cùng giá trị trong một hoặc nhiều cột. Nó cho phép áp dụng các hàm tổng hợp như SUM (DISTINCT <COLUMN_NAME>), COUNT (DISTINCT <COLUMN_NAME>), AVG (DISTINCT <COLUMN_NAME>), MAX (DISTINCT <COLUMN_NAME>), MIN (DISTINCT <COLUMN_NAME>) vào từng nhóm, thay vì cho toàn bộ bảng.
ALIAS	Tên tạm thời.
COLUMN_NAME_1 COLUMN_NAME_2 COLUMN_NAME_N	Trường thông tin tại các bảng được lấy ra. Các trường thông tin xuất hiện tại SELECT thì cũng sẽ xuất hiện tương tự tại GROUP BY.

6. WITH CUBE

/* Method 1: Không có ALIAS */

```
WITH CUBE
```

/* Method 2: Có ALIAS */

```
WITH CUBE
```

WITH CUBE	Phần mở rộng của mệnh đề GROUP BY. Cú pháp được sử dụng để tạo các kết quả tổng hợp cho tất cả các tổ hợp có thể của các cột được liệt kê trong mệnh đề GROUP BY. Nó tạo ra các tổ hợp tổng hợp cho **từng cột** và **tất cả các tổ hợp của các cột đó**.
ALIAS	Tên tạm thời.

7. WITH ROLLUP

/* Method 1: Không có ALIAS */

WITH ROLLUP

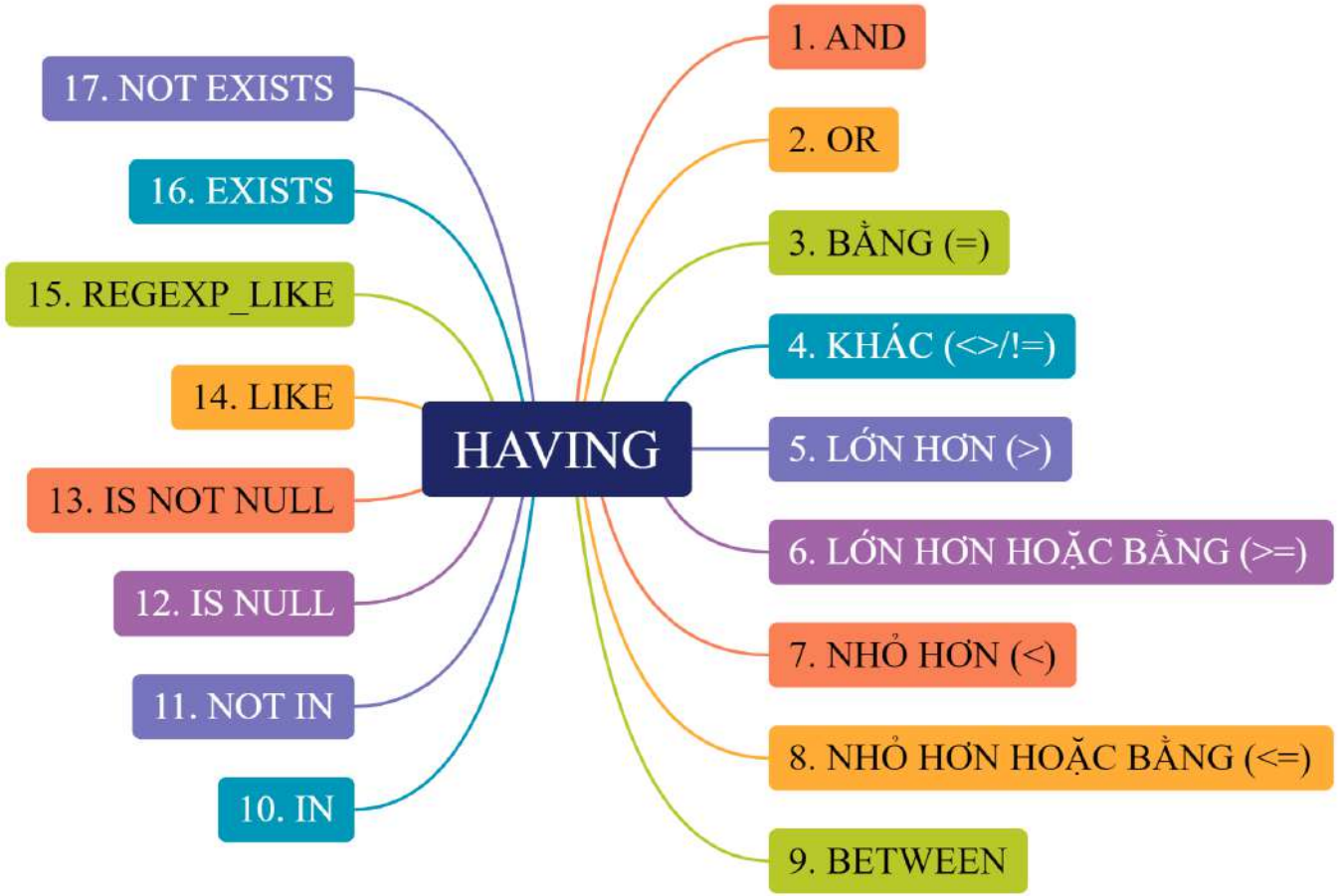
/* Method 2: Có ALIAS */

WITH ROLLUP

WITH ROLLUP	Phần mở rộng của mệnh đề GROUP BY. Cú pháp tạo ra các kết quả tổng hợp với các mức tổng hợp của các cột được liệt kê trong GROUP BY. Nó cung cấp số liệu được **tổng hợp của các cột**.
ALIAS	Tên tạm thời.

8. HAVING

HAVING: Tương tự như WHERE, truy vấn dữ liệu theo các tiêu chí được tạo bởi mệnh đề GROUP BY và thực hiện các thao tác chỉnh sửa dữ liệu dựa trên các điều kiện này. Các toán tử được sử dụng trong câu lệnh HAVING gồm:



8. HAVING

/* Method 1: Không có ALIAS */

HAVING <HAVING_CONDITION>

/* Method 2: Có ALIAS */

HAVING <HAVING_CONDITION>

HAVING	Cú pháp được sử dụng để áp dụng điều kiện lọc cho các nhóm được tạo bởi mệnh đề GROUP BY. Nó cho phép lọc các nhóm dựa trên các điều kiện tổng hợp (aggregate conditions) như SUM (DISTINCT <COLUMN_NAME>), COUNT (DISTINCT <COLUMN_NAME>), AVG (DISTINCT <COLUMN_NAME>), MAX (DISTINCT<COLUMN_NAME>), MIN (DISTINCT <COLUMN_NAME>).
ALIAS	Tên tạm thời.

9. SELECT

/* Method 1: Không có ALIAS */

SELECT (DISTINCT)
<COLUMN_NAME_1>, <COLUMN_NAME_2>, ..., <COLUMN_NAME_N> (<AGGREGATE CONDITIONS>)

/* Method 2: Có ALIAS */

SELECT (DISTINCT)
<ALIAS_1>.<COLUMN_NAME_1>, <ALIAS_2>.<COLUMN_NAME_2>, ..., <ALIAS_N>.<COLUMN_NAME_N> (<AGGREGATE CONDITIONS>)

SELECT	Cú pháp cơ bản và quan trọng nhất, cho phép lấy thông tin từ bảng hoặc các đối tượng dữ liệu khác như view, procedure, function và kết quả các phép toán.
DISTINCT; ALIAS	Cú pháp để loại bỏ các giá trị trùng lặp; Tên tạm thời.
COLUMN_NAME_1 COLUMN_NAME_2 COLUMN_NAME_N	Trường thông tin tại các bảng được lấy ra.
AGGREGATE CONDITIONS	SUM (<COLUMN_NAME>), COUNT (<COLUMN_NAME>), AVG (<COLUMN_NAME>), MAX (<COLUMN_NAME>), MIN (<COLUMN_NAME>).

10. ORDER BY

/* Method 1: Không có ALIAS */

ORDER BY <COLUMN_NAME> **<ASC/DESC>**

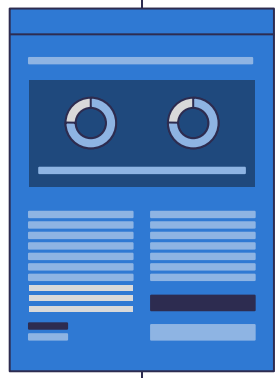
/* Method 2: Có ALIAS */

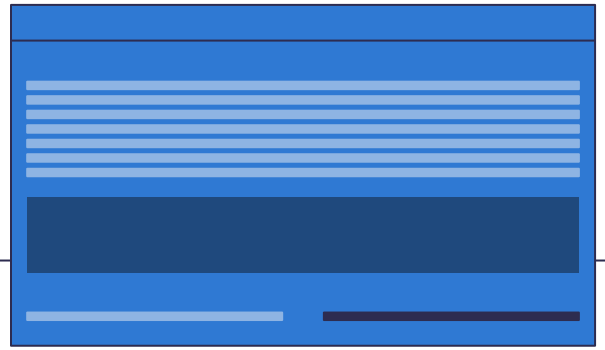
ORDER BY <ALIAS>.<COLUMN_NAME> **<ASC/DESC>**

ORDER BY ASC/DESC	Cú pháp để sắp xếp các kết quả của câu truy vấn theo một hoặc nhiều cột theo thứ tự tăng dần (ASC, mặc định) hoặc giảm dần (DESC). ORDER BY là một trong những câu lệnh quan trọng để điều chỉnh thứ tự hiển thị của dữ liệu trả về từ câu truy vấn. Áp dụng sau khi dữ liệu đã được lọc (WHERE) và nhóm hóa (GROUP BY và HAVING).
ALIAS	Tên tạm thời.
COLUMN_NAME	Trường thông tin sẽ sắp xếp

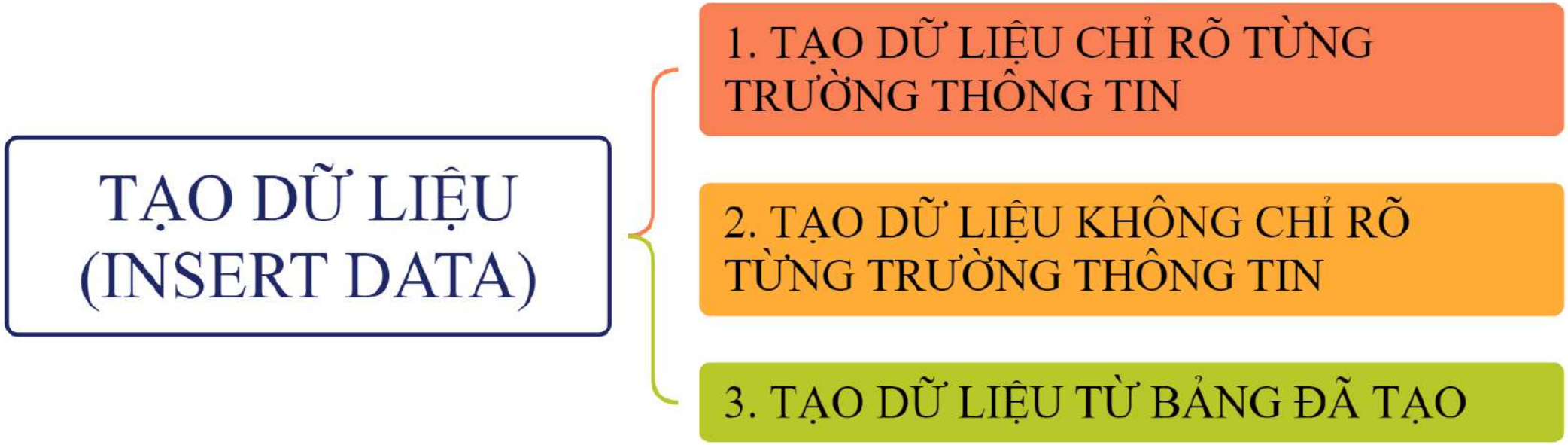
02

Tạo Dữ Liệu





Tạo Dữ Liệu


TẠO DỮ LIỆU
(INSERT DATA)
1. TẠO DỮ LIỆU CHỈ RÕ TỪNG TRƯỜNG THÔNG TIN
2. TẠO DỮ LIỆU KHÔNG CHỈ RÕ TỪNG TRƯỜNG THÔNG TIN
3. TẠO DỮ LIỆU TỪ BẢNG ĐÃ TẠO

Tạo Dữ Liệu

/* Method 1: Tạo dữ liệu chỉ rõ từng trường thông tin */

INSERT INTO <TABLE_NAME> (COLUMN_NAME_1, COLUMN_NAME_2, ..., COLUMN_NAME_N)

VALUES (VALUE_1, VALUE_2, …, VALUE_N);

<table>
<tr><td>INSERT INTO … VALUES</td><td colspan="2">Cú pháp tạo dữ liệu cho bảng (trong Oracle chỉ tạo 1 lần 1 bản ghi)</td></tr>
<tr><td>TABLE_NAME</td><td colspan="2">Tên bảng</td></tr>
<tr><td>COLUMN_NAME</td><td colspan="2">Tên cột trong bảng, thêm dữ liệu vào cột nào thì liệt kê cột đó.</td></tr>
<tr><td rowspan="4">VALUE</td><td colspan="2">Giá trị tương ứng với cột khai báo bên trên</td></tr>
<tr><td>Số</td><td>VALUE = 12345</td></tr>
<tr><td>Chuỗi ký tự</td><td>VALUE = ‘String’ hoặc VALUE = N‘String’</td></tr>
<tr><td rowspan="2">Ngày giờ</td><td>VALUE = TO_DATE (‘16/05/2024’, ‘DD/MM/YYYY’)</td></tr>
<tr><td>VALUE = SYSDATE</td></tr>
</table>

Tạo Dữ Liệu

VÍ DỤ: Tạo dữ liệu chỉ rõ từng trường thông tin

```
CREATE TABLE table_example ( -- Tạo bảng thông thường bảng ví dụ
   id NUMBER PRIMARY KEY, -- Mã là số, khóa chính
   description NVARCHAR2(100), -- Mô tả là chuỗi quốc gia tối đa 100 ký tự
   created_date DATE DEFAULT SYSDATE -- Thời điểm tạo là ngày tháng
);

INSERT INTO table_example (id, description, created_date) -- Tạo dữ liệu cho 3 trường thông tin trong bảng ví dụ
VALUES (1, N'Bản ghi đầu tiên', SYSDATE); -- Thông tin dữ liệu cho 3 trường thông tin tương ứng khai báo lần lượt bên trên

INSERT INTO table_example (id, description, created_date) -- Tạo dữ liệu cho 3 trường thông tin trong bảng ví dụ
VALUES (2, N'Bản ghi thứ hai', SYSDATE); -- Thông tin dữ liệu cho 3 trường thông tin tương ứng khai báo lần lượt bên trên
```

Tạo Dữ Liệu

/* Method 2: Tạo dữ liệu không chỉ rõ từng trường thông tin */

INSERT INTO <TABLE_NAME>

VALUES (VALUE_1, VALUE_2, …, VALUE_N);

<table>
<tr><td>INSERT INTO …VALUES</td><td colspan="2">Cú pháp tạo dữ liệu cho bảng (trong Oracle chỉ tạo 1 lần 1 bản ghi)</td></tr>
<tr><td>TABLE_NAME</td><td colspan="2">Tên bảng</td></tr>
<tr><td rowspan="5">VALUE</td><td colspan="2">Giá trị tương ứng với cột được tạo trong bảng</td></tr>
<tr><td>Số</td><td>VALUE = 12345</td></tr>
<tr><td>Chuỗi ký tự</td><td>VALUE = ‘String’ hoặc VALUE = N‘String’</td></tr>
<tr><td rowspan="2">Ngày giờ</td><td>VALUE = TO_DATE (‘16/05/2024’, ‘DD/MM/YYYY’)</td></tr>
<tr><td>VALUE = SYSDATE</td></tr>
</table>

Tạo Dữ Liệu

VÍ DỤ: Tạo dữ liệu không chỉ rõ từng trường thông tin

```
CREATE TABLE table_example ( -- Tạo bảng thông thường bảng ví dụ
  id NUMBER PRIMARY KEY, -- Mã là số, khóa chính
  description NVARCHAR2(100), -- Mô tả là chuỗi quốc gia tối đa 100 ký tự
  created_date DATE DEFAULT SYSDATE -- Thời điểm tạo là ngày tháng
);

INSERT INTO table_example -- Tạo dữ liệu cho tất cả trường thông tin trong bảng ví dụ
VALUES (1, N'Bản ghi đầu tiên', SYSDATE); -- Thông tin dữ liệu cho 3 trường thông tin tương ứng khai báo lần lượt trong bảng từ trên xuống

INSERT INTO table_example -- Tạo dữ liệu cho 3 trường thông tin trong bảng ví dụ
VALUES (2, N'Bản ghi thứ hai', SYSDATE); -- Thông tin dữ liệu cho 3 trường thông tin tương ứng khai báo lần lượt trong bảng từ trên xuống
```

Tạo Dữ Liệu

/* Method 3: Tạo dữ liệu từ bảng có sẵn */

INSERT INTO <TABLE_NAME>

VALUES (VALUE_1, VALUE_2, …, VALUE_N);

SELECT <COLUMN_NAME_1>, <COLUMN_NAME_2>, …, <COLUMN_NAME_N>,

FROM <EXISTED_TABLE_NAME>

WHERE <CONDITION>

INSERT INTO … VALUES	Cú pháp tạo dữ liệu cho bảng (trong Oracle chỉ tạo 1 lần 1 bản ghi)
SELECT … FROM	Chọn bảng và các cột trong bảng
WHERE	Điều kiện lấy dữ liệu

Tạo Dữ Liệu

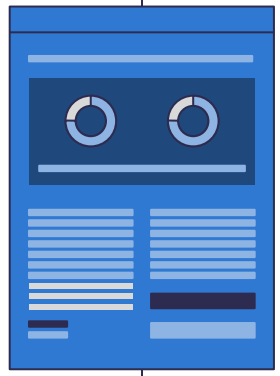
VÍ DỤ: Tạo dữ liệu từ bảng có sẵn

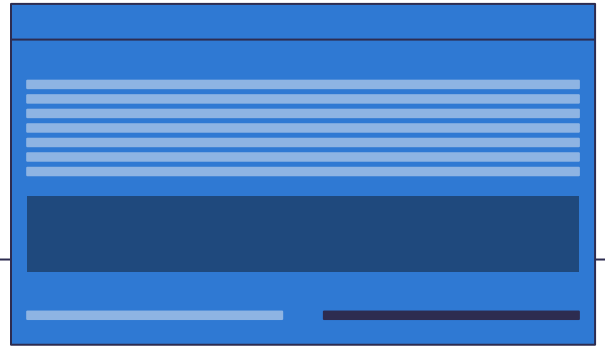
```
CREATE TABLE table_example ( -- Tạo bảng thông thường bảng ví dụ
   id NUMBER PRIMARY KEY, -- Mã là số, khóa chính
   description NVARCHAR2(100), -- Mô tả là chuỗi quốc gia tối đa 100 ký tự
   created_date DATE DEFAULT SYSDATE -- Thời điểm tạo là ngày tháng
);


INSERT INTO new_table_example (id, description, created_date) -- Tạo dữ liệu cho 3 trường thông tin trong bảng ví dụ
SELECT id, description, created_date -- Chọn tất các trường thông tin tại bảng ví dụ
FROM table_example -- Từ bảng ví dụ
WHERE id = 100; -- Với điều kiện chỉ lấy mã = 100
```

03

Sửa Dữ Liệu





Sửa Dữ Liệu

UPDATE <TABLE_NAME> <ALIAS_T>

(JOIN … ON)

SET

<ALIAS_T>.<COLUMN_NAME_1> = <VALUE_1>,
<ALIAS_T>.<COLUMN_NAME_2> = <VALUE_2>,
…
<ALIAS_T>.<COLUMN_NAME_N> = <VALUE_N>

(SUBQUERY)

WHERE <CONDITION>;

Sửa Dữ Liệu

UPDATE … SET	Cú pháp sửa/cập nhật dữ liệu cho bảng
JOIN … ON	Cú pháp kết nối với một bảng khác nếu liên quan đến việc cập nhật dữ liệu
SUBQUERY	Câu truy vấn con nếu liên quan đến việc cập nhật dữ liệu
WHERE	Cú pháp điều kiện lấy dữ liệu nếu liên quan đến việc cập nhật dữ liệu
CONDITION	Các toán tử so sánh sẽ thực hiện tại đây
ALIAS_T	Tên tạm thời của bảng cần cập nhật dữ liệu
COLUMN_NAME	Tên trường thông tin cần cập nhật dữ liệu
VALUE	Giá trị cần cập nhật

Sửa Dữ Liệu

VÍ DỤ: Sửa dữ liệu của 1 cột cho tất cả các bản ghi

```
CREATE TABLE employees ( -- Tạo bảng thông thường nhân viên
  employee_id NUMBER(6) CONSTRAINT emp_pk PRIMARY KEY, -- Mã nhân viên là số tối đa 6 ký tự, khóa chính
  first_name VARCHAR2(20) NOT NULL, -- Tên là chuỗi tối đa 20 ký tự, không được bỏ trống
  last_name VARCHAR2(25) NOT NULL, -- Họ là chuỗi tối đa 25 ký tự, không được bỏ trống
  email VARCHAR2(50) CONSTRAINT emp_email_uk UNIQUE, -- Email là chuỗi tối đa 50 ký tự, duy nhất
  hire_date DATE DEFAULT SYSDATE NOT NULL, -- Ngày thuê là ngày tháng, mặc định nếu không xác định được thì lấy thời điểm hiện tại, không bỏ trống
  job_id VARCHAR2(10) NOT NULL, -- Mã công việc là chuỗi tối đa 10 ký tự, không được bỏ trống
  salary NUMBER(8, 2) CHECK (salary >= 0), -- Tiền lương là số tối đa 8 ký tự bao gồm 6 ký tự nguyên và 2 ký tự thập phân, kiểm tra lương phải lớn hơn hoặc bằng 0
  is_manager CHAR(1) DEFAULT 'N' CHECK (is_manager IN ('Y', 'N')), -- Mã quản lý là chuỗi tối đa 1 ký tự, mặc định nếu không xác định được thì lấy 'N', kiểm tra mã quản lý phải có ký tự là 'Y' hoặc 'N'
  department_id NUMBER(4) CONSTRAINT emp_dept_fk REFERENCES departments(department_id) -- Mã phòng ban là số tối đa 4 ký tự nguyên, khóa ngoại liên kết với khóa chính mã phòng ban của bảng phòng ban
);

UPDATE employees -- Cập nhật bảng nhân viên
SET salary = salary * 1.1; -- Tăng lương của tất cả nhân viên lên 10%
```

Sửa Dữ Liệu

VÍ DỤ: Sửa dữ liệu của 1 cột với điều kiện

```
CREATE TABLE employees ( -- Tạo bảng thông thường nhân viên
   employee_id NUMBER(6) CONSTRAINT emp_pk PRIMARY KEY, -- Mã nhân viên là số tối đa 6 ký tự, khóa chính
   first_name VARCHAR2(20) NOT NULL, -- Tên là chuỗi tối đa 20 ký tự, không được bỏ trống
   last_name VARCHAR2(25) NOT NULL, -- Họ là chuỗi tối đa 25 ký tự, không được bỏ trống
   email VARCHAR2(50) CONSTRAINT emp_email_uk UNIQUE, -- Email là chuỗi tối đa 50 ký tự, duy nhất
   hire_date DATE DEFAULT SYSDATE NOT NULL, -- Ngày thuê là ngày tháng, mặc định nếu không xác định được thì lấy thời điểm hiện tại, không bỏ trống
   job_id VARCHAR2(10) NOT NULL, -- Mã công việc là chuỗi tối đa 10 ký tự, không được bỏ trống
   salary NUMBER(8, 2) CHECK (salary >= 0), -- Tiên lương là số tối đa 8 ký tự bao gồm 6 ký tự nguyên và 2 ký tự thập phân, kiểm tra lương phải lớn hơn hoặc bằng 0
   is_manager CHAR(1) DEFAULT 'N' CHECK (is_manager IN ('Y', 'N')), -- Mã quản lý là chuỗi tối đa 1 ký tự, mặc định nếu không xác định được thì lấy 'N', kiểm tra mã quản lý phải có ký tự là 'Y' hoặc 'N'
   department_id NUMBER(4) CONSTRAINT emp_dept_fk REFERENCES departments(department_id) -- Mã phòng ban là số tối đa 4 ký tự nguyên, khóa ngoại liên kết với khóa chính mã phòng ban của bảng phòng ban
);

UPDATE employees -- Cập nhật bảng nhân viên
SET salary = salary * 1.05 -- Tăng lương của những nhân viên thuộc phòng ban IT lên 5%
WHERE department_id = 1; -- Với điều kiện mã phòng ban = 1
```

Sửa Dữ Liệu

VÍ DỤ: Sửa dữ liệu của nhiều cột với điều kiện

```
CREATE TABLE employees ( -- Tạo bảng thông thường nhân viên
   employee_id NUMBER(6) CONSTRAINT emp_pk PRIMARY KEY, -- Mã nhân viên là số tối đa 6 ký tự, khóa chính
   first_name VARCHAR2(20) NOT NULL, -- Tên là chuỗi tối đa 20 ký tự, không được bỏ trống
   last_name VARCHAR2(25) NOT NULL, -- Họ là chuỗi tối đa 25 ký tự, không được bỏ trống
   email VARCHAR2(50) CONSTRAINT emp_email_uk UNIQUE, -- Email là chuỗi tối đa 50 ký tự, duy nhất
   hire_date DATE DEFAULT SYSDATE NOT NULL, -- Ngày thuê là ngày tháng, mặc định nếu không xác định được thì lấy thời điểm hiện tại, không bỏ trống
   job_id VARCHAR2(10) NOT NULL, -- Mã công việc là chuỗi tối đa 10 ký tự, không được bỏ trống
   salary NUMBER(8, 2) CHECK (salary >= 0), -- Tiên lương là số tối đa 8 ký tự bao gồm 6 ký tự nguyên và 2 ký tự thập phân, kiểm tra lương phải lớn hơn hoặc bằng 0
   bonus NUMBER(8,2), -- Tiền bổ sung thêm là số tối đa 8 ký tự bao gồm 6 ký tự nguyên và 2 ký tự thập phân
   is_manager CHAR(1) DEFAULT 'N' CHECK (is_manager IN ('Y', 'N')), -- Mã quản lý là chuỗi tối đa 1 ký tự, mặc định nếu không xác định được thì lấy 'N', kiểm tra mã quản lý phải có ký tự là 'Y' hoặc 'N'
   department_id NUMBER(4) CONSTRAINT emp_dept_fk REFERENCES departments(department_id) -- Mã phòng ban là số tối đa 4 ký tự nguyên, khóa ngoại liên kết với khóa chính mã phòng ban của bảng phòng ban
);


UPDATE employees -- Cập nhật bảng nhân viên
SET
   salary = salary * 1.03, -- Tăng lương lên 3%
   bonus = 2000 -- Đặt mức thưởng là 2000 đồng
WHERE salary < 50000; -- Với điều kiện lương < 50000
```

Sửa Dữ Liệu

VÍ DỤ: Sửa dữ liệu từ nhiều bảng

```
UPDATE employees e -- Cập nhật bảng nhân viên - đặt bí danh bảng là e
INNER JOIN salary_changes sc ON e.employee_id = sc.employee_id -- JOIN với bảng thay đổi lương - đặt bí danh là sc, lấy ra những bản ghi chung của cả 2 bảng
SET e.salary = e.salary + sc.amount; -- Tăng lương của nhân viên theo thông tin từ bảng salary_changes
```

VÍ DỤ: Sửa dữ liệu bằng cách sử dụng Subquery

```
UPDATE employees -- Cập nhật bảng nhân viên
SET manager_id = ( -- Cập nhật cột mã quản lý
  SELECT manager_id -- Chọn mã quản lý
  FROM employees -- Từ bảng nhân viên
  WHERE employee_id = 2 -- Giả sử manager của nhân viên có employee_id = 2 là manager mới
)
WHERE department_id = 1; -- Cập nhật manager cho những nhân viên thuộc phòng ban IT
```

04

Xóa Dữ Liệu

Xóa Dữ Liệu

DELETE FROM <TABLE_NAME> <ALIAS_T>

WHERE <CONDITION>

(COMMIT/ROLLBACK);

DELETE FROM	Cú pháp thực hiện xóa dữ liệu
WHERE	Cú pháp điều kiện lấy dữ liệu nếu liên quan đến việc xóa dữ liệu
CONDITION	Các toán tử so sánh hoặc Subquery sẽ thực hiện tại đây
COMMIT	Cú pháp xác nhận thay đổi
ROLLBACK	Cú pháp hoàn tác thay đổi

Xóa Dữ Liệu

VÍ DỤ: Xóa hết dữ liệu xác nhận thay đổi

```
DELETE FROM employees; -- Xóa hết bản ghi từ bảng nhân viên
COMMIT; -- Xác nhận các thay đổi (commit)
```

VÍ DỤ: Xóa hết dữ liệu hoàn tác thay đổi

```
DELETE FROM employees; -- Xóa hết bản ghi từ bảng nhân viên
ROLLBACK; -- Hoàn tác các thay đổi (rollback)
```

VÍ DỤ: Xóa hết dữ liệu với điều kiện

```
DELETE FROM employees -- Xóa hết bản ghi từ bảng nhân viên
WHERE department_id = 1 AND salary < 60000; -- Với điều kiện mã phòng ban = 1 và lương < 60000
```

BATIZENS

Trân trọng cảm ơn!