Cơ sở dữ liệu ứng dụng

Giới thiệu

CRUD

Trong một hệ thống lưu trữ dữ liệu, có 4 chức năng cơ bản, được gọi tắt là CRUD:

- Create: tạo mới
- Read: đọc
- Update: thay đổi
- Delete: xóa

4 chức năng này trong T-SQL được thực hiện qua 2 nhóm ngôn ngữ là DML và DQL.

DML

DML (Data Manipulation Language) là nhóm các mệnh đề để thao tác trên dữ liệu, bao gồm các nhóm:

- INSERT
- UPDATE
- DELETE

DQL

DQL (Data Query Language) là nhóm các mệnh đề để truy vấn dữ liệu, bao gồm:

- SELECT

INSERT

Giới thiệu

Nhóm mệnh đề INSERT giúp thêm dữ liệu vào cơ sở dữ liệu, có 2 dạng nhóm nhỏ:

- INSERT
- BULK INSERT

INSERT

Dùng để nhập trực tiếp dữ liệu vào database

Cú pháp:

```
INSERT <table_name> (<column_list>) VALUES
(<value>)
```

Ví dụ

```
INSERT sinh_vien(ma_so, ten, nam_sinh) VALUES
(‘11110102’, N‘Nguyễn Văn A’, 1996)
```

Lưu ý

Các cột phải được phân cách bằng dấu phẩy, tương tự cho trường trong một dòng giá trị.

Thứ tự các trường trong dữ liệu nhập vào phải tương ứng với cột.

Số lượng các trường trong một giá trị luôn luôn bằng với số cột được khai báo trong phần INSERT.

Cách nhập giá trị

Ta có bảng sau:

Kiểu dữ liệu	Ví dụ
int	123, 456, 1212121212, …
float	12.34, 125.3467, …
char, varchar	‘abc’, ‘ght’, …
nchar, nvarchar	N‘Nguyễn Văn A’, …
datetime2	‘2019-07-27 12:30:00.123’, …
giá trị null	NULL, null, Null, …

Bổ sung

Để nhập nhiều dòng cùng lúc, ta dung cú pháp sau:

```
INSERT <table_name> (<column_list>) VALUES
(<value_1>),
(<value_2>),
…,
(<value_n>)
```

Lưu ý

Nếu có cột không nằm trong phần liệt kê thì khi insert dữ liệu cột đó sẽ lấy giá trị NULL hoặc giá trị mặc định được chỉ ra trong ràng buộc DEFAULT.

Nếu cột có ràng buộc IDENTITY thì không liệt kê khi insert, giá trị cột sẽ tự động được thêm vào.

Giá trị được thêm vào phải thỏa mãn ràng buộc cột.

SELECT

Giới thiệu

Câu lệnh SELECT dùng để truy vấn đến dữ liệu trong cơ sở dữ liệu.

Truy vấn một bảng không điều kiện

```
SELECT
    <column_list>
FROM <table_name>
```

Lưu ý

Trong <column_list>:

- Tên các cột được phân cách với nhau bằng dấu phẩy
- Thứ tự các cột không cần phải giống với thứ tự cột của bảng.
- Thứ tự cột được truyền sẽ là thứ tự cột trong kết quả trả về của dữ liệu.
- Có thể dùng ký tự * để lấy hết các cột, khi đó thứ tự của cột sẽ dùng thứ tự khi định nghĩa bảng.

Truy vấn một bảng có điều kiện

```
SELECT
    <column_list>
FROM <table_name>
WHERE
    <condition>
```

Ý nghĩa

Lấy ra những dòng dữ liệu thỏa mãn điều kiện được chỉ ra.

Lưu ý

<condition> trong T-SQL có thể là một điều kiện đơn hay nhiều điều kiện đơn được ghép lại với nhau.

Một điều kiện đơn có cấu trúc:

<biểu_thức_của_cột> <toán_tử_so_sánh> <giá_trị_so_sánh>

Biểu thức cột

- Có thể là tên cột
- Hoặc là một thao tác nào đó trên cột (tính toán số học trên cột kiểu số, thao tác chuỗi trên một cột kiểu chuỗi, ...)

Toán tử so sánh

Các toán tử so sánh thông dụng trong T-SQL:

- `=, <>`
- `>, >=, <, <=`
- `BETWEEN … AND …`
- `IN, NOT IN`
- `LIKE`
- `IS NULL, IS NOT NULL` (toán tử này không có giá trị so sánh liền sau)

Toán tử = và <>

Dùng để so sánh biểu thức cột có giống hay khác với giá trị so sánh.

Dùng cho các kiểu dữ liệu số, chuỗi, thời gian.

Toán tử >, >=, < và <=

Dùng để so sánh độ lớn của biểu thức cột với giá trị so sánh.

Dùng cho các kiểu dữ liệu số, thời gian.

Toán tử BETWEEN x AND y

Dùng để kiểm tra biểu thức cột có nằm trong [x, y].

Dùng cho các kiểu dữ liệu số, thời gian.

Toán tử IN

Dùng để kiểm tra biểu thức cột có nằm trong một tập hợp giá trị nào đó hay không.

Giá trị so sánh có dạng (<giá_trị_1>, <giá_trị_2>, ..., <giá_trị_n>)

Dùng cho các kiểu dữ liệu số, chuỗi, thời gian.

Toán tử NOT IN

Dùng để kiểm tra biểu thức cột có **không** nằm trong một tập hợp giá trị nào đó hay không.

Giá trị so sánh có dạng (<giá_trị_1>, <giá_trị_2>, ..., <giá_trị_n>)

Dùng cho các kiểu dữ liệu số, chuỗi, thời gian.

Toán tử LIKE

Dùng cho kiểu chuỗi.

Dùng để kiểm tra biểu thức cột có chứa một chuỗi nào đó hay không.

Dùng chung với các wild card :

- ‘_’: dùng để thay thế cho 1 ký tự
- ‘%’. dùng để thay thế cho 0 hoặc nhiều ký tự

Ví dụ toán tử LIKE

Cho cột dữ liệu sau:

col
abcdef
abcd
foo
bar

[col] LIKE ‘abc_’

col
abcdef
abcd
foo
bar

[col] LIKE ‘%f%’

col
abcdef
abcd
foo
bar

Toán tử IS NULL

Dùng để kiểm tra giá trị của biểu thức cột có phải giá trị NULL

Dùng cho các kiểu dữ liệu số, chuỗi, thời gian

Toán tử IS NOT NULL

Dùng để kiểm tra giá trị của biểu thức cột không phải giá trị NULL.

Dùng cho các kiểu dữ liệu số, chuỗi, thời gian.

Giá trị NULL

Là một giá trị đặc biệt của T-SQL, dùng để biểu diễn việc không có dữ liệu.

Ta có các bảng sau

NOT(A)

A	¬A
F	T
U	U
T	F

AND(A, B)

A ∧ B		B		
		F	U	T
A	F	F	F	F
	U	F	U	U
	T	F	U	T

OR(A, B)

A ∨ B		B		
		F	U	T
A	F	F	U	T
	U	U	U	T
	T	T	T	T

Biểu thức kép

Gồm nhiều biểu thức đơn được kết nối với nhau bằng các toán tử AND, OR và các dấu ngoặc đơn.

Đặt mới tên cột trong kết quả trả về

Có thể đổi tên cột trong kết quả trả về bằng cách như sau

```
SELECT
    <col> [AS] <alias>
FROM …
```

Việc đổi tên bảng được sử dụng trong trường hợp truy vấn nhiều bảng và cũng được thực hiện tương tự, tức là

```
<table_name> [AS] <alias>
```

Từ khóa ORDER BY

Dùng để sắp xếp kết quả trả về

```
SELECT ...
FROM ...
[WHERE ... ]
ORDER BY <column_name> [ASC|DESC]
```

Lưu ý

Mặc định của ORDER BY là ASC, tức là sắp xếp tăng dần.

Có thể sắp xếp theo nhiều cột

```
ORDER BY <column_1> [ASC|DESC], <column_2> [ASC|DESC], ...
```

Cột sắp xếp không cần phải nằm trong kết quả trả về.

Từ khóa TOP

Dùng để giới hạn kết quả trả về trong câu lệnh SELECT.

Có hai dạng:

- Trả về số lượng tuyệt đối.
- Trả về số lượng tương đối.

```
SELECT TOP(n) ... FROM ...
```

```
SELECT TOP(n) PERCENT ... FROM ...
```

Từ khóa DISTINCT

Dùng để trả về những kết quả khác nhau trong câu SELECT.

```
SELECT DISTINCT <col_1>, <col_2>, ..., <col_n> FROM ...
```

INSERT INTO ... SELECT ...

Dùng để thêm dữ liệu vào một bảng với dữ liệu từ bảng khác

```
INSERT INTO <insert_table> (<column_list>)
SELECT <column_list>
FROM <select_table>
[WHERE ...]
```

UPDATE

Giới thiệu

Mệnh đề UPDATE dùng để thay đổi dữ liệu của một hoặc nhiều dòng trong cơ sở dữ liệu.

Cú pháp

```
UPDATE <table_name>
SET
    <column_1> = <value_1>,
    <column_2> = <value_2>,
    ...,
    <column_n> = <value_n>
WHERE <condition>
```

Lưu ý

Nên có mệnh đề WHERE để giới hạn dữ liệu cần thay đổi.

Từ khóa TOP

Tương tự với câu SELECT, từ khóa TOP trong câu UPDATE dùng để giới hạn số lượng dữ liệu cần cập nhật.

DELETE

Công dụng

Dùng để xóa dữ liệu trong cơ sở dữ liệu

Cú pháp

```
DELETE FROM <table_name>
WHERE <condition>
```

Lưu ý

Nếu trong bảng có cột IDENTITY thì dù có xóa hết bảng bằng DELETE thì giá trị cột IDENTITY vẫn tiếp tục từ số lớn nhất trước khi xóa, không được đặt lại từ giá trị khởi tạo.

Từ khóa TOP

Tương tự với câu SELECT, từ khóa TOP trong câu DELETE dùng để giới hạn số lượng dữ liệu cần xóa.

TRUNCATE TABLE

Dùng để xóa toàn bộ dữ liệu trong một bảng.

Tương tự với câu DELETE không có điều kiện.

Nếu trong bảng có cột IDENTITY thì số sẽ được đánh lại từ đầu.

Cú pháp

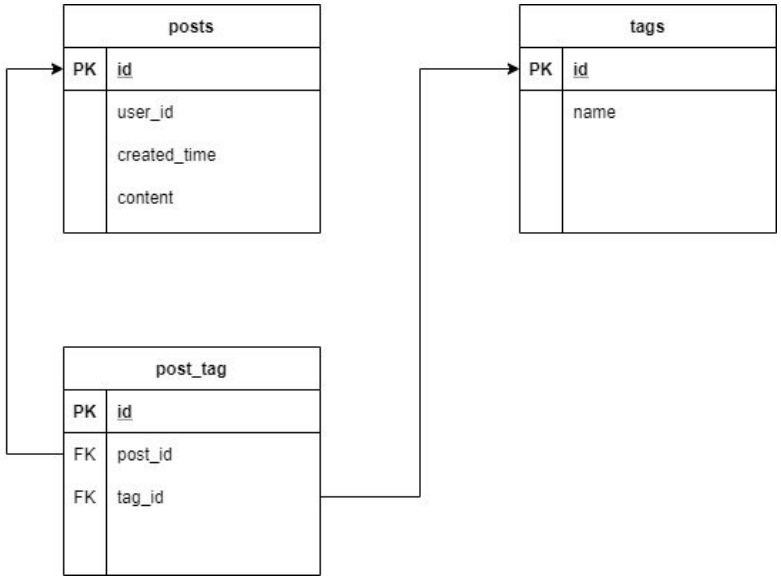
```
TRUNCATE TABLE <table_name>
```

JOIN

Giới thiệu

Cho mô hình như hình bên.

Làm thế nào để truy vấn tên những tag của một post nào đó.



Ý nghĩa

Từ khóa JOIN dùng để liên kết thông tin từ hai hay nhiều bảng khác nhau dựa trên điều kiện nào đó.

Cú pháp JOIN 2 bảng

```
SELECT
    <column_list>
FROM          <table_1> AS <alias_1>
<join_type>   <table_2> AS <alias_2>
ON            <alias_1>.<fk_column> = <alias_2>.<pk_column>
```

Chú thích

<table_1> được gọi là bảng trái.

<table_2> được gọi là bảng phải.

<table_1>.<fk_column> = <table_2>.<pk_column> được gọi là điều kiện join.

Diễn giải

Với mỗi dòng ở bảng trái, dựa vào điều kiện join và loại join sẽ trả về những dữ liệu tương ứng ở bảng phải.

Các loại JOIN

Có các loại JOIN:

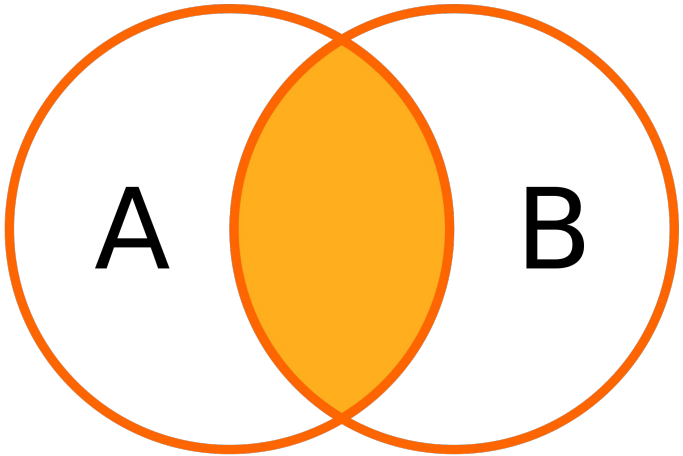
- JOIN/INNER JOIN
- LEFT JOIN/LEFT OUTER JOIN
- RIGHT JOIN/RIGHT OUTER JOIN
- CROSS JOIN
- FULL OUTER JOIN

Trong đó, thường dùng nhất là INNER JOIN và LEFT JOIN.

INNER JOIN/JOIN

Chỉ trả về những dòng có điều kiện join là TRUE.

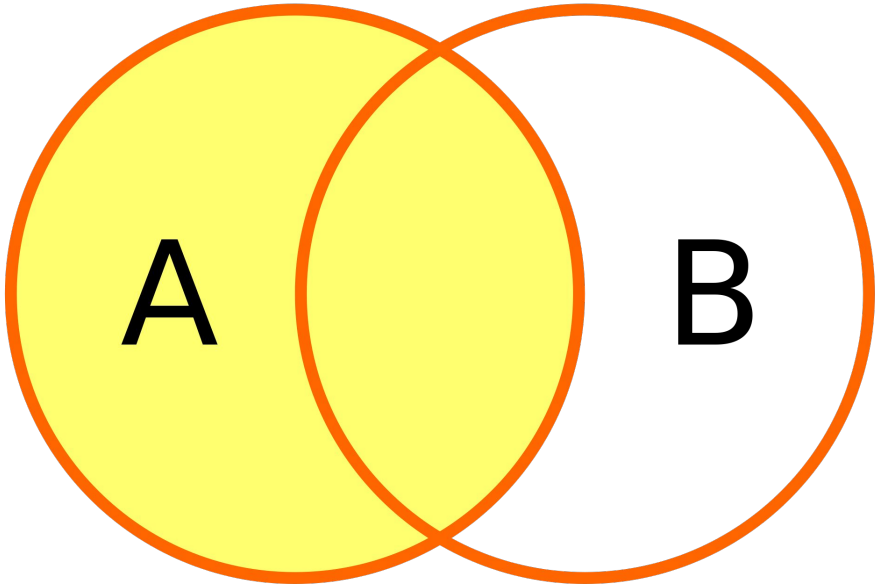
Nói cách khác, nếu không có dòng tương ứng ở bảng phải, sẽ không trả về dòng ở bảng trái.



LEFT OUTER JOIN/LEFT JOIN

Với mỗi dòng ở bảng trái:

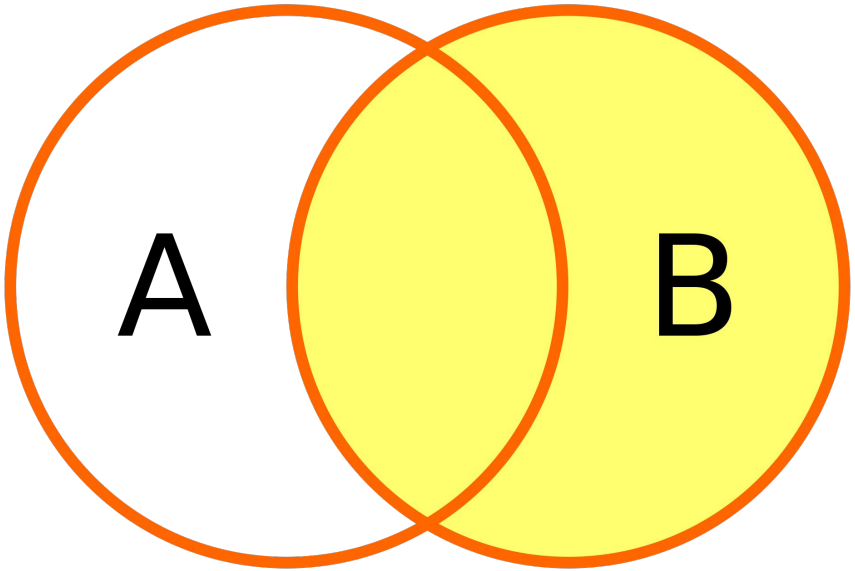
- Nếu có dữ liệu tương ứng ở bảng phải, trả về dữ liệu đó.
- Nếu không có dữ liệu tương ứng ở bảng phải, trả về NULL.



RIGHT OUTER JOIN/RIGHT JOIN

Ta có:

```
A RIGHT JOIN B = B LEFT JOIN A
```



Phân tích dữ liệu

Giới thiệu

Ngoài việc truy vấn dữ liệu, T-SQL còn cung cấp nhiều công cụ cho việc phân tích dữ liệu. Các công cụ này bao gồm:

- Các hàm định nghĩa sẵn.
- Lập trình T-SQL

Các hàm dựng sẵn

Là các hàm trong T-SQL được viết sẵn.

Các hàm này được phân thành các nhóm:

- Hàm chuyển đổi kiểu dữ liệu.
- Hàm logic
- Hàm toán học
- Hàm xử lý chuỗi
- Hàm xử lý thời gian
- Hàm tổng hợp thông tin

Lưu ý khi tính toán

Trong T-SQL:

- Phép chia hai số nguyên là phép chia nguyên, tức là 3 / 2 sẽ trả về 1. Nếu muốn trả về float, thực hiện 3 * 1.0 / 2.

Hàm toán học

Category	Function	Brief Description
Scientific and Trig Functions	ACOS	Arc Cosine (inverse cosine)
	ASIN	Arc Sine (inverse sine)
	ATAN	Arc Tangent (inverse tangent)
	ATN2	Arc Tangent (inverse tangent)
	COS	Cosine
	COT	Cotangent
	DEGREES	Degrees from Radians
	EXP	Exponent
	LOG	Natural logarithm (ln)
	LOG10	Log base 10
	PI	Apple pie? I think not!
	POWER	Power function
	RADIANS	Radians from Degrees
	SIN	Sine
	SQRT	Square Root
	SQUARE	Square
	TAN	Tangent
Rounding Functions	CEILING	Ceiling
	FLOOR	Floor
	ROUND	Round Number
Signs	ABS	Absolute Value
	SIGN	Determine Sign of Value
Random Numbers	RAND	Generate Random Number

Hàm xử lý chuỗi

Category	Function	Description
Position	CHARINDEX	Find position of one or more characters in another value
	LEN	Return the number of characters
	PATINDEX	CHARINDEX on super vitamins...
Transformation	LEFT	Return beginning portion of value
	LOWER	Return value as all lower case characters
	LTRIM	Remove any beginning spaces
	QUOTENAME	Make the value legal for SQL code generation
	REPLACE	Replace one set of characters with another
	REPLICATE	Repeat characters
	REVERSE	Flip the value end to end
	RIGHT	Return the last portion of the value
	RTRIM	Remove any trailing spaces
	SPACE	Create a value of repeated spaces
	STR	Convert a number to a text value.
	STUFF	Insert characters inside another value
	SUBSTRING	Return a portion of a value, such as the middle.
	UPPER	Return value as all UPPER CASE characters
Character set	ASCII	Return the ASCII code for a character
	CHAR	Return the Character for the corresponding ASCII code
	NCHAR	Like CHAR but for UNICODE.
	UNICODE	Like ASCII but for UNICODE.
Soundex	DIFFERENCE	An interesting way to compare differences in strings.
	SOUNDEX	An interesting way to compare strings.

Hàm xử lý thời gian

Category	Function
Current Date and Time	SYSDATETIME
	SYSDATETIMEOFFSET
	SYSUTDATETIME
	CURRENT_TIMESTAMP
	GETDATE
	GETUTCDATE
Validate	ISDATE
Date Parts	DATENAME
	DATEPART
	DAY
	MONTH
	YEAR
Difference and Modification	DATEDIFF
	DATEADD
	EOMONTH
	SWITCHOFFSET
	TODATETIMEOFFSET
Build a Date	DATEFROMPARTS
	DATETIME2FROMPARTS
	DATETIMEFROMPARTS
	DATETIMEOFFSETFROMPARTS
	SMALLDATETIMEFROMPARTS
	TIMEFROMPARTS

Hàm logic

Bao gồm các hàm COALESCE(), IIF().

Các hàm này có thể sử dụng trong các mệnh đề SELECT, UPDATE, WHERE, ...

COALESCE

Cú pháp:

```
COALESCE(<value_1>, <value_2>, …, <value_n>)
```

Công dụng:

Dùng để trả về giá trị đầu tiên khác NULL trong danh sách các giá trị truyền vào.

IIF

Cú pháp

`IIF(<condition>, <true_value>, <false_value>)`

Công dụng:

Dựa vào giá trị đúng sai của `<condition>` mà trả về `<true_value>` hoặc `<false_value>`

Cấu trúc CASE

```
-- Cú pháp
CASE
    WHEN <case_1> THEN <expression_1>
    WHEN <case_2> THEN <expression_2>
    ...
    WHEN <case_n> THEN <expression_n>
    [ELSE <else_expression>]
END
```

Hàm tổng hợp thông tin

Bao gồm các hàm sau:

- COUNT(*): trả về số dòng trong kết quả truy vấn
- MIN(<column>): trả về giá trị nhỏ nhất.
- MAX(<column>): trả về giá trị lớn nhất.
- SUM(<column>): trả về tổng của cột
- AVG(<column>): trả về trung bình của cột

Các hàm này thường được dùng chung với GROUP BY

GROUP BY

Dùng để sắp xếp các dòng của kết quả truy vấn vào các nhóm khác nhau.

Thường dùng chung với các hàm tổng hợp COUNT, MIN, MAX, SUM, AVG.

GROUP BY

```
-- cú pháp
SELECT
    [<column_1>, ..., <column_n>],
    <hàm_tổng_hợp_1>, ..., <hàm_tổng_hợp_n>
FROM <table>
[WHERE <condition>]
GROUP BY <column_1>, ..., <column_n>
```

GROUP BY

Các cột trong phần SELECT:

- Hoặc là phải nằm trong GROUP BY
- Hoặc là phải nằm trong một hàm tổng hợp nào đó.

HAVING

Từ khóa HAVING dùng để kiểm tra điều kiện của các hàm tổng hợp.

Từ khóa HAVING được thêm vào vì WHERE không thể dùng với các hàm tổng hợp.

Từ khóa HAVING luôn dùng chung với GROUP BY

HAVING

```
-- cú pháp
SELECT
    [<column_1>, ..., <column_n>],
    <hàm_tổng_hợp_1>, ..., <hàm_tổng_hợp_n>
FROM <table>
[WHERE <condition>]
GROUP BY <column_1>, ..., <column_n>
HAVING <điều_kiện_với_hàm_tổng_hợp>
```

Truy vấn lồng nhau (Subquery)

Khái niệm

Truy vấn lồng nhau là một câu truy vấn có thể chứa câu truy vấn khác bên trong nó.

Câu truy vấn bên trong gọi là truy vấn phụ.

Câu truy vấn bên ngoài gọi là truy vấn chính.

Ví dụ

```sql
--truy vấn các bàn thắng trong trận chung kết
-- câu truy vấn bên trong dấu ngoặc là truy vấn phụ
SELECT
    team_id, player, goal_time
FROM goals
WHERE
    game_id = (SELECT id FROM games WHERE [round] = 'FI')
```

Ví dụ

```
-- truy vấn các cầu thủ đã ghi bàn và số bàn thắng của họ
-- trong vòng loại trực tiếp
SELECT
    player, count(*)
FROM goals
WHERE game_id IN (SELECT id FROM games WHERE [round] <> 'RO')
GROUP BY player
```

Cách hoạt động

Khi thực hiện một câu truy vấn lồng nhau, câu truy vấn phụ sẽ được thực hiện trước, sau đó mới thực hiện câu truy vấn chính.

Ví dụ

```
-- với câu truy vấn
SELECT
    team_id, player, goal_time
FROM goals
WHERE game_id = (SELECT id FROM games WHERE [round] = 'FI')
-- câu truy vấn phụ được thực hiện trước, kết quả là 1031
-- kế tiếp sẽ thực hiện câu truy vấn
SELECT
    team_id, player, goal_time
FROM goals
WHERE game_id = 1031
```

Phạm vi sử dụng

Câu truy vấn phụ có thể sử dụng trong các mệnh đề INSERT, SELECT, UPDATE và DELETE:

- SELECT: truy vấn phụ có thể dùng như một cột hoặc dùng trong phần WHERE.
- INSERT: dùng truy vấn trong cấu trúc INSERT INTO ... SELECT ...
- UPDATE, DELETE: thường dùng trong phần WHERE

Ví dụ

```sql
-- truy vấn các trận đấu tứ kết và tên sân vận động tổ chức
SELECT
    id, [date], home_team_id, away_team_id,
    (SELECT
        stadium_name
        FROM stadiums
    WHERE stadiums.id = games.stadium_id) as stadium_name
FROM games
WHERE [round] = 'QF'
```

Lưu ý

- Chỉ có thể truy vấn một cột hoặc một biểu thức cột trong câu truy vấn phụ, trừ trường hợp dùng với EXISTS, NOT EXISTS. Tức là, kết quả của câu truy vấn phụ chỉ có thể là một giá trị duy nhất hoặc một cột các giá trị.
- Câu truy vấn phụ phải nằm trong dấu ngoặc đơn.
- Câu truy vấn phụ có thể dùng bảng của câu truy vấn chính trong phần WHERE của mình (như trong ví dụ liền trước)

Truy vấn phụ trong phần WHERE

Câu truy vấn phụ trong phần WHERE thường dùng với các toán tử:

- Các toán tử so sánh =, <>, >, >=, <, <=
- Toán tử IN, NOT IN
- Toán tử EXISTS, NOT EXISTS

Truy vấn phụ và các toán tử so sánh

Kết quả truy vấn phụ phải là một giá trị duy nhất.

```
-- ví dụ như câu dưới đây
-- chúng ta biết là chỉ có 1 trận chung kết
-- nên có thể dùng toán tử '='
SELECT
    team_id, player, goal_time
FROM goals
WHERE
    game_id = (SELECT id FROM games WHERE [round] = 'FI')
```

Truy vấn phụ với IN, NOT IN

Kết quả trả về có thể là một giá trị hoặc một cột giá trị

Truy vấn phụ và EXISTS, NOT EXISTS

Cú pháp

```
WHERE EXISTS (<truy_vấn_phụ>)
```

Từ khóa `EXISTS` dùng để kiểm tra `<truy_vấn_phụ>` có dữ liệu hay không. Vì chỉ quan tâm đến việc tồn tại của dữ liệu, `<truy_vấn_phụ>` bắt đầu bằng `SELECT *`

```sql
-- truy vấn những trận đấu không có bàn thắng
SELECT
    *
FROM games
WHERE NOT EXISTS (SELECT * FROM goals WHERE games.id =
goals.game_id)
-- câu truy vấn phụ truy vấn những bàn thắng trong các trận đấu
-- nếu không có dữ liệu tức là trận đấu không có bàn thắng
```

Cấu trúc CASE

Cấu trúc CASE dùng để trả về giá trị thích hợp dựa trên một tập các giá trị cho trước.

Cấu trúc CASE có thể dùng trong các mệnh đề SELECT, UPDATE, DELETE, WHERE, ORDER BY