THIẾT KẾ HỆ CSDL QUAN HỆ - XÂY DỰNG CSDL VẬT LÝ

Nguyễn Đình Hóa

dinhhoa@gmail.com

094-280-7711

Ngôn ngữ thao tác dữ liệu

- Ngôn ngữ thao tác dữ liệu của SQL có thể được chia ra làm hai phần tách riêng nhưng vẫn chung nhau ở một số phạm vi nào đó: các câu lệnh DML không truy vấn và các câu lệnh truy vấn dữ liệu.
- Ngôn ngữ thao tác không truy vấn cho phép bạn thêm dữ liệu vào bảng, sửa đổi dữ liệu, xoá dữ liệu từ các bảng.
- Các câu lệnh truy vấn DML bao gồm câu lệnh đơn SELECT với rất nhiều các mệnh đề lựa chọn khác nhau.

Tóm tắt các câu lệnh DML của SQL

Câu lệnh hoặc lựa chọn	Mô tả
INSERT	Chèn thêm một (các) hàng vào trong một bảng
SELECT	Lựa chọn các thuộc tính từ các hàng trong một hoặc nhiều bảng hoặc khung nhìn
WHERE	Giới hạn việc lựa chọn các hàng dựa trên một biểu thức điều kiện
GROUP BY	Gộp nhóm các hàng đã được chọn ra dựa trên một hoặc nhiều thuộc tính
HAVING	Hạn chế sự lựa chọn các hàng để gộp nhóm dựa trên một điều kiện
ORDER BY	Xếp thứ tự các hàng được chọn
UPDATE	Sửa đổi giá trị thuộc tính của một hoặc nhiều hàng của một bảng
DELETE	Xoá một hoặc nhiều hàng từ một bảng
COMMIT	Lưu trữ vĩnh viễn những thay đổi về dữ liệu
ROLLBACK	Phục hồi dữ liệu về những giá trị ban đầu của chúng
Các phép toán so sánh	
=, <, >, <=, >=, <>	Được sử dụng trong các biểu thức điều kiện
Các phép toán logic	
AND, OR, NOT	Được sử dụng trong các biểu thức điều kiện

Tóm tắt các câu lệnh DML của SQL

Câu lệnh hoặc lựa chọn	Mô tả
Các phép toán đặc biệt	*được sử dụng trong các biểu thức điều kiện*
BETWEEN	Kiểm tra xem các giá trị của một thuộc tính có nằm trong một khoảng xác định
IS NULL	Kiểm tra xem giá trị của một thuộc tính có là trống / hoặc có giá trị không
LIKE	Kiểm tra xem giá trị của một thuộc tính có giống với một kiểu chuỗi ký tự cho trước
IN / NOT IN	Kiểm tra xem giá trị của một thuộc tính có nằm trong / hoặc không nằm trong một danh sách các giá trị nào đó
EXISTS / NOT EXISTS	Kiểm tra xem một truy vấn con có trả về hàng dữ liệu nào không
DISTINCT	Hạn chế các giá trị tới những giá trị duy nhất, hay loại bỏ những giá trị trùng lặp
Các hàm thống kê	*được sử dụng với SELECT để trả về những giá trị tổng hợp trên các cột*
COUNT	Trả về số lượng các hàng với các giá trị không rỗng cho một cột nào đó
MIN	Trả về giá trị nhỏ nhất của một thuộc tính được tìm thấy trong một cột nào đó
MAX	Trả về giá trị lớn nhất của một thuộc tính được tìm thấy trong một cột nào đó
SUM	Trả về tổng của tất cả các giá trị của một cột nào đó
AVG	Trả về giá trị trung bình của tất cả c ác giá trị của một cột nào đó

Thêm bản ghi vào bảng

```
INSERT INTO tablename
        VALUES (value1, value 2, ...value n);
```

- Ví dụ

```
INSERT INTO PRODUCT
VALUES ('23114-AA',  'Sledge hammer, 12 lb.', '02-Jan-02', 8, 5, 14.40, 0.05, NULL);
```

```
INSERT INTO PRODUCT
VALUES ('23114-AA',  'Sledge hammer, 12 lb.', NULL, NULL, NULL, NULL, NULL, NULL);
-or-
INSERT INTO PRODUCT(P_CODE, P_DESCRIPT)
VALUES('23114-AA', 'Sledge hammer, 12 lb.');
```

Thêm bản ghi trùng giá trị ở khoá

```
INSERT INTO `table_name` (`index_field`, `other_field_1`, `other_field_2`)
VALUES ('index_value', 'insert_value', 'other_value')
ON DUPLICATE KEY UPDATE
`other_field_1` = 'update_value',
`other_field_2` = VALUES(`other_field_2`);
```

Trường hợp này xảy ra khi giá trị của khoá bị trùng với một bản ghi có sẵn, lúc đó thuộc tính `other_field_1` sẽ được cập nhật giá trị mới, còn thuộc tính `other_field_2` vẫn mang giá trị cũ.

Thêm nhiều bản ghi

```
INSERT INTO `my_table` (`field_1`, `field_2`) VALUES
('data_1', 'data_2'),
('data_1', 'data_3'),
('data_4', 'data_5');
```

Bỏ qua lỗi khi thêm dữ liệu

- Các lỗi khi thêm dữ liệu:
  - Trùng nhau trên giá trị của khoá
  - Nhập sai kiểu dữ liệu trên các thuộc tính
- Sử dụng INSERT IGNORE
  - Nếu trùng nhau trên khoá chính, giá trị các thuộc tính khác sẽ đươc cập nhật
  - Nếu nhập sai kiểu dữ liệu, hệ thống sẽ tự động chuyển kiểu dữ liệu cho các giá trị mới về gần nhất với giá trị mong muốn.

INSERT với các biến tự cập nhật

```
CREATE TABLE t (
        id SMALLINT UNSIGNED AUTO_INCREMENT
        NOT NULL,
        this ...,
        that ...,
        PRIMARY KEY(id) );
INSERT INTO t (this, that) VALUES (..., ...);
```

Câu lệnh LAST_INSERT_ID(id) sẽ trích xuất giá trị id của bản ghi cuối cùng mới được thêm vào.

Ví dụ

```
CREATE TABLE iodku (
        id INT AUTO_INCREMENT NOT NULL,
        name VARCHAR(99) NOT NULL,
        misc INT NOT NULL,
        PRIMARY KEY(id),
        UNIQUE(name)
        );
INSERT INTO iodku (name, misc)
VALUES
('Leslie', 123),
('Sally', 456); -- thêm 2 bản ghi vào bảng
```

Ví dụ (tiếp)

INSERT INTO iodku (name, misc)

VALUES

('Sally', 3333) -- *bản ghi mới cần được thêm vào*

ON DUPLICATE KEY UPDATE -- *thuộc tính `name` bị trùng với giá trị trước đó*

id = LAST_INSERT_ID(id),

misc = **VALUES**(misc);

SELECT LAST_INSERT_ID(); -- *lấy ra giá trị id mới thêm vào, vẫn chỉ là giá trị số 2 vì không bản ghi nào được thêm*

INSERT đi kèm với SELECT

```
INSERT INTO `tableA` (`field_one`, `field_two`)
        SELECT `tableB`.`field_one`, `tableB`.`field_two`
        FROM `tableB`
        WHERE `tableB`.clmn <> 'someValue'
        ORDER BY `tableB`.`sorting_clmn`;
```

Xoá bản ghi khỏi bảng

DELETE FROM *tablename*

[WHERE *conditionlist*];

- Ví dụ

```
DELETE FROM PRODUCT
WHERE P_CODE = '23114-AA';
```

DELETE là một câu lệnh hướng tập hợp. Có nghĩa là điều kiện ở câu lệnh WHERE là có thể có hoặc không, nếu điều kiện đó không được chỉ rõ thì tất cả các hàng của bảng sẽ được xoá!

Một số từ khoá đi kèm DELETE

Các từ khoá	Ý nghĩa
LOW_PRIORITY	Với từ khoá này, lệnh delete sẽ được thực hiện sau cùng so với các câu lệnh khác được thực hiện đồng thời
IGNORE	Bỏ qua mọi lỗi xảy ra trong quá trình thực hiện câu lệnh
ORDER BY *expression*	Các bản ghi sẽ được xoá với thứ tự được nêu
LIMIT	Giới hạn số lượng tối đa các bản ghi sẽ được xoá

Xoá các bản ghi từ nhiều bảng

```
DELETE p1, p2
   FROM people p1 JOIN pets p2 ON p2.ownerId = p1.id
   WHERE p1.name = 'Paul';
```

Câu lệnh trên sẽ xoá các bản ghi ở cả 2 bảng people và pets thoả mãn điều kiện trong phần WHERE.

Câu lệnh trên cũng có thể viết như sau

```
DELETE
   FROM people p1 JOIN pets p2 ON p2.ownerId = p1.id
   WHERE p1.name = 'Paul';
```

Cập nhật dữ liệu cho bảng

- Cập nhật đơn giản

```
UPDATE tablename
   SET columnname = expression [, columnname = expression ]
   [ WHERE conditionlist ];
```

- Lưu ý rằng điều kiện WHERE là không bắt buộc trong câu lệnh UPDATE. Nếu không có điều kiện WHERE, thì câu lệnh UPDATE sẽ được thực hiện trên tất cả các bản ghi của bảng đó.

Cập nhật một lúc nhiều hàng

```
UPDATE people
        SET name =
                        (CASE id WHEN 1 THEN 'Karl'
                                 WHEN 2 THEN 'Tom'
                                 WHEN 3 THEN 'Mary'
                        END)
        WHERE id IN (1,2,3);
```

Câu lệnh cập nhật tổng quát

UPDATE [**LOW_PRIORITY**] [**IGNORE**] tableName
SET column1 = expression1,
column2 = expression2,
...
[**WHERE** conditions]
[**ORDER BY** expression [**ASC** | **DESC**]]
[**LIMIT** row_count];

Cập nhật trên nhiều bảng

UPDATE [**LOW_PRIORITY**] [**IGNORE**] table1, table2, ...
SET column1 = expression1,
column2 = expression2,
...
[**WHERE** conditions]

Truy vấn dữ liệu

- Câu lệnh chung

SELECT [ALL | DISTINCT] *columnlist*

FROM *tablelist*

[WHERE *condition*]

[GROUP BY *columnlist*]

[HAVING *condition*]

[ORDER BY *columnlist*];

Truy vấn dữ liệu

- **SELECT DISTINCT** `name`, `price` **FROM** CAR;

Câu truy vấn trên sẽ loại bỏ các bản ghi trùng nhau ở bảng kết quả đầu ra

- **SELECT** * **FROM** stack;

Câu truy vấn trên liệt kê tất cả các bản ghi có trong bảng stack

Các toán tử đặc biệt trong truy vấn

BETWEEN – được dùng để kiểm tra xem giá trị một thuộc tính có nằm trong một khoảng nào đó không.

IS NULL – được dùng để xác định liệu một thuộc tính có nhận giá trị NULL không.

LIKE – được dùng để gắn giá trị một thuộc tính với một kiểu chuỗi ký tự. Nhiều ký tự thay thế được sẵn có để sử dụng với toán tử này.

IN – được sử dụng để xác định liệu giá trị một thuộc tính có nằm trong một danh sách giá trị không.

EXISTS – được dùng để xác định liệu một truy vấn con có trả về một tập rỗng hay không.

Ví dụ về NULL

Cho lược đồ quan hệ:

Phim (tên, năm, độ_dài, loại, xưởng_phim, đạo_diễn)

trong đó "độ_dài" được tính theo phút. Hãy cho biết sự khác nhau giữa kết quả của hai câu truy vấn sau:

```
SELECT *
FROM Phim;

SELECT *
FROM Phim
WHERE độ_dài <= 120 OR độ_dài > 120;
```

Câu truy vấn thứ hai không liệt kê các bản ghi mà thuộc tính "độ_dài" có trạng thái NULL.

Toán tử LIKE

- Toán tử LIKE được sử dụng kết hợp với một hàm đặc biệt để tìm các khối ký tự (pattern) trong các thuộc tính có kiểu ký tự.
- SQL chuẩn cho phép sử dụng các ký tự đặc biệt "%" và "_" để tìm kiếm tương tự cho các xâu ký tự.

"%" được dùng đại diện cho một chuỗi ký tự.

'M%' có thể trả về: Mark, Marci, M-234x, v.v.

"_" được dùng đại diện cho một ký tự.

'_07-345-887_' trả về: 4**07-345-887**1, 0**07-345-887**5

Toán tử IN

- Các câu truy vấn có sử dụng toán tử logic OR có thể được thay thế bằng toán tử đặc biệt IN.
- Ví dụ, với câu truy vấn sau:

```
SELECT *
    FROM PRODUCT
    WHERE V_CODE = 21344 OR V_CODE = 24288;
```

Ta có thể thay thế bằng:

```
SELECT *
    FROM PRODUCT
    WHERE V_CODE IN (21344, 24288);
```

Ví dụ

Cho lược đồ quan hệ sau:

Quán (tên, bia, giá)

Hãy sử dụng toán tử IN để viết câu truy vấn liệt kê tên các loại bia được bán đồng thời ở hai quán Hải Xồm và Lan Chín.

```
SELECT bia
FROM Quán
WHERE tên = 'Hải Xồm'
AND bia IN (SELECT bia
            FROM Quán
            WHERE tên = 'Lan Chín');
```

Toán tử EXISTS

- Toán tử EXISTS được sử dụng khi có một câu lệnh được thực hiện dựa trên kết quả của một lệnh khác. Cụ thể, nếu một câu truy vấn phụ có trả về kết quả thì câu truy vấn chính mới chạy, không thì thôi.
- Cú pháp của toán tử EXISTS là

```
WHERE EXISTS  ( subquerry );
```

- Câu truy vấn phụ subquerry là một câu lệnh SELECT nào đó.
- Ta cũng có thể sử dụng toán tử NOT EXISTS với nghĩa ngược lại.
- Lưu ý: Cần hạn chế sử dụng câu lệnh SQL có sử dụng toán tử EXISTS vì câu truy vấn phụ luôn phải chạy lại sau mỗi dòng của câu truy vấn chính.

Ví dụ 1 về toán tử EXISTS

Cho các lược đồ sau

Sinh_vien (MaSV, ten, tuoi)
Mon_hoc (MaM, ten, so_tin_chi)
Hoc (MaSV, MaM, diem)

Viết câu truy vấn SQL để liệt kê các sinh viên đã học một môn nào đó

```
SELECT *
FROM Sinh_vien
WHERE EXISTS (SELECT *
                    FROM Hoc
                    WHERE Sinh_vien.MaSV =
Hoc.MaSV);
```

Ví dụ 2 sử dụng toán tử EXISTS

Cho các lược đồ sau

Sinh_vien (MaSV, ten, tuoi)
Mon_hoc (MaM, ten, so_tin_chi)
Hoc (MaSV, MaM, diem)

Viết câu truy vấn SQL để xóa các sinh viên không học một môn nào.

```
DELETE FROM Sinh_vien
WHERE NOT EXISTS (SELECT *
                  FROM Hoc
                  WHERE Sinh_vien.MaSV =
Hoc.MaSV);
```

Ví dụ 3

Cho lược đồ sau:

Beers (name, manf)

Hãy viết câu truy vấn SQL để liệt kê tên các bia duy nhất của các hãng sản xuất.

```
SELECT name
FROM Beers AS b1
WHERE NOT EXISTS  (SELECT *
                   FROM Beers AS b2
                   WHERE b2.manf = b1.manf AND
                         b2.name <> b1.name);
```

Truy vấn với CASE hoặc IF

```
SELECT st.name, st.percentage,
CASE WHEN st.percentage >= 35 THEN 'Pass' ELSE 'Fail'
END AS `Remark`
FROM student AS st ;
```

Tương đương với

```
SELECT st.name, st.percentage,
IF(st.percentage >= 35, 'Pass', 'Fail') AS `Remark`
FROM student AS st ;
```

Giới hạn số bản ghi truy vấn

```
SELECT *
FROM Customers
ORDER BY CustomerID
LIMIT 3;
```

Câu truy vấn trên chỉ trả về tối đa 3 bản ghi kết quả

```
SELECT *
FROM Customers
ORDER BY CustomerID
LIMIT 2,1;
```

Câu truy vấn trên sẽ bỏ 2 bản ghi đầu, chỉ hiện 1 bản ghi thứ 3

Các hàm trong SQL

function_name(*cột*, [tham số 1, tham số 2, …])

- Các hàm thao tác trên từng bản ghi:
  - LOWER(A) – chuyển đổi ký tự thành kiểu chữ thường
  - UPPER(a) – chuyển đổi ký tự thành kiểu chữ in hoa

    Ví dụ: **SELECT UPPER**(*LastName*), **LOWER**(*FirstName*) **FROM** *student* ;
  - ROUND(a) – làm tròn số
  - PI() – lấy giá trị pi
  - SQRT(a) – căn bậc hai
  - POWER(a,b) – hàm mũ

Các hàm trong SQL

- Các hàm thao tác trên từng bản ghi:
  - Hàm chuyển đổi kiểu dữ liệu CONVERT

    Ví dụ1: chuyển đổi số 2011 thành xâu ký tự “2011” như sau

    SELECT CONVERT(2011, **CHAR**(5));

    Ví dụ 2: Lấy tất cả các thông tin về khóa học bắt đầu vào ngày 25 một tháng bất kỳ. (Ngày bắt đầu của khóa học lưu trong cột *StartDate* của bảng **course**)

    SELECT *

    FROM *course*

    WHERE CONVERT(*StartDate***, CHAR(**15**)) LIKE** "25%";

Các hàm trong SQL

- Các hàm thao tác trên nhiều bản ghi:
  - MAX() – tìm giá trị lớn nhất cho các thuộc tính kiểu số
  - MIN() – tìm giá trị nhỏ nhất cho các thuộc tính kiểu số
  - AVG() – tìm giá trị trung bình cho các thuộc tính kiểu số
  - COUNT() – đếm số bản ghi
  - SUM() – tìm tổng các giá trị cho các thuộc tính kiểu số

  Ví dụ:

  SELECT AVG(*Age*) **, SUM**(*Salary*)

  FROM *nhanvien;*

Gom nhóm các bản ghi

SELECT *columnlist*

FROM *tablelist*

[WHERE *conditionlist*]

[GROUP BY *columnlist*]

[HAVING *condtionlist*];

Gom nhóm các bản ghi

- Một số quy tắc cần nhớ khi sử dụng vế câu GROUP BY trong câu lệnh SELECT:
  - Trong danh mục *columnlist* của SELECT phải có sự kết hợp giữa tên cột và các hàm thống kê.
  - Trong danh mục *columnlist của vế GROUP BY* bao gồm các cột trong *columnlist* của phần SELECT mà không chứa hàm thống kế. Tùy từng trường hợp, ta có thể gom nhóm theo bất kỳ cột chứa hàm thống kê nào có trong *columnlist* của phần SELECT.
  - Danh mục *columnlist* của vế GROUP BY có thể bao gồm bất kỳ cột nào trong bảng xác định bởi về FROM của câu lệnh SELECT, kể cả khi chúng không xuất hiện trong danh mục *columnlist* của SELECT.

Ví dụ 1

- Cho các lược đồ quan hệ sau:

  SinhVien (MaSV, TEN, DiaChi, NgaySinh)

  Hoc (MaSV, MaMH, Diem)

Hãy viết câu lệnh SQL để:

- Liệt kê mã số của các sinh viên cùng với tổng số môn mà các sinh viên đó đã học.
- Liệt kê tên của từng sinh viên cùng điểm cao nhất mà họ đã đạt được trong các môn học.

Ví dụ 2

- Cho các lược đồ quan hệ sau:

  SinhVien (MaSV, TEN, DiaChi, NgaySinh)

  Hoc (MaSV, MaMH, Diem)

Hãy viết câu lệnh SQL để:

- Liệt kê tên của các sinh viên có điểm cao nhất trong từng môn học.
- Liệt kê tên của các sinh viên đã học tất cả các môn học.
- Liệt kê tên của các sinh viên học nhiều môn nhất.

Ví dụ 3

Cho các lược đồ quan hệ sau:

Sinh_vien (MaSV, ten, tuoi)
Mon_hoc (MaM, ten)
Hoc (MaSV, MaM, diem)

Hãy viết các câu lệnh SQL để:

- Liệt kê tên sinh viên cùng điểm trung bình của các sinh viên xếp loại khá trở lên (điểm trung bình >=2.5)
- Liệt kê mã sinh viên có điểm số lớn hơn điểm trung bình của từng môn học

Ví dụ 4

- Cho các lược đồ quan hệ sau:

  SinhVien (MaSV, TEN, DiaChi, NgaySinh)

  Hoc (MaSV, MaMH, Diem)

Hãy viết câu lệnh SQL để:

- Liệt kê tên của các sinh viên đã từng học ít nhất 3 môn học.