1. Các thành phần chính

Primary key

- Khóa chính là tập hợp một hoặc nhiều column giúp phân biệt các record trong một table
- Tạo primary key:

```
CREATE TABLE table_name (
   pk_column data_type PRIMARY KEY,
   ...
);
```

hoặc

```
ALTER TABLE table_name
ADD PRIMARY KEY(pk_column data_type);
```

Foreign Key

- Đây là mối liên kết giữa hai bảng với nhau tạo thành một lược đồ cơ sở dữ liệu quan hệ.
- Tạo Foreign key:

```
CONSTRAINT fk_constraint_name
FOREIGN KEY (column_1, column2,...)
REFERENCES parent_table_name(column1,column2,..)
```

hoặc

```
ALTER TABLE parent_table_name ADD FOREIGN KEY fk_constraint_name
FOREIGN KEY (column_1, column2,...) REFERENCES child_table_name(column_1,
column2,...)
```

2. Câu lệnh cơ bản - Điều kiện và các toán tử

Tạo bảng

```
CREATE TABLE <tên bảng> (

<tên thuộc tính> <kiểu dữ liệu> [<RBTV>],
<tên thuộc tính> <kiểu dữ liệu> [<RBTV>],
...
[<RBTV>]
)
```

RBTV

• NOT NULL
• NULL
• UNIQUE
• DEFAULT
• PRIMARY KEY
• FOREIGN KEY / REFERENCES
• CHECK

Đặt tên cho RBTV

• CONSTRAINT <tên RBTV> <RBTV>

Xóa bảng

```
DROP TABLE <tên bảng> [RESTRICT | CASCADE]

DROP TABLE <tên bảng>
```

Sửa bảng

```
ALTER TABLE <tên bảng> ADD COLUMN <tên thuộc tính> <kiểu dữ liệu>
[<RBTV>]

ALTER TABLE <tên bảng> ALTER COLUMN <tên thuộc tính> <kiểu dữ
liệu mới>

ALTER TABLE <tên bảng> DROP CONSTRAINT <tên RBTV>
```

Tạo và xóa miền giá trị

```
CREATE DOMAIN <tên kiểu dữ liệu mới> AS <kiểu dữ liệu cơ bản>

DROP DOMAIN <tên kiểu dữ liệu>
```

Cập nhật dữ liệu

Chèn dữ liệu

• Chèn từng dòng
- INSERT INTO <tên bảng> [(<danh sách thuộc tính>)] VALUES (<danh sách giá trị>)

• Chèn nhiều dòng
- INSERT INTO <tên bảng> [(<danh sách thuộc tính>)]<lệnh truy vấn>

Xóa dữ liệu

- `DELETE FROM <tên bảng> [WHERE <điều kiện>]`

Sửa dữ liệu

- `UPDATE <tên bảng>`

```
SET <tên thuộc tính> = <giá trị mới>,
<tên thuộc tính> = <giá trị mới>,
...
[WHERE <điều kiện>]
```

Truy vấn dữ liệu

Cú pháp

- `SELECT <dstt>`

```
FROM <dsb>
[WHERE <đk>]
[GROUP BY <dstt nhóm>]
[HAVING <đk nhóm>]
[ORDER BY <dstt sắp xếp>]
```

Kết quả của lệnh truy vấn là một bảng.

Bảng trong SQL có thể chứa các bộ trùng nhau.

Phép toán tập hợp

SQL hỗ trợ các phép toán

- UNION (Hội)
- EXCEPT (Hiệu).
- INTERSECT (Giao).
- Đặc điểm

- Các dòng giống nhau bị loại trong bảng kết quả.
- Các bảng tham gia phép toán phải có tính khả hợp

Giữ lại các dòng giống nhau

- UNION ALL
- EXCEPT ALL
- INTERSECT ALL

```
 (SELECT <dstt> FROM <dsb> WHERE <đk>)
UNION [ALL]
(SELECT <dstt> FROM <dsb> WHERE <đk>)
 (SELECT <dstt> FROM <dsb> WHERE <đk>)
EXCEPT [ALL]
(SELECT <dstt> FROM <dsb> WHERE <đk>)
 (SELECT <dstt> FROM <dsb> WHERE <đk>)
INTERSECT [ALL]
(SELECT <dstt> FROM <dsb> WHERE <đk>)
```

Phép toán số học

+, –, *, / có thể áp dụng cho các giá trị số trong mệnh đề SELECT
select 1.1 * Luong as ‘Luong moi’
from NHANVIEN
+, - có thể áp dụng cho các giá trị kiểu ngày giờ
select NgNhanChuc + 150 as ‘Cong Ngay’
from PHONGBAN

Phép toán so sánh và luận lý

Dùng trong mệnh đề WHERE hoặc HAVING để xây dựng các điều kiện chọn và điều kiện kết.
• =, <, ≤, >, ≥, <>
• BETWEEN <giá trị> AND <giá trị>
• AND, OR, NOT

Phép toán so sánh chuỗi

```
 LIKE <mẫu đối sánh>
[ESCAPE <kí tự thoát>]
```

Mẫu đối sánh

• Chuỗi ký tự để so sánh.
• % - thay cho một đoạn ký tự tùy ý.
• _ - thay cho một ký tự tùy ý.

Ký tự thoát

• Ký tự để loại bỏ chức năng đặc biệt của % và _.
• Có thể dùng ký tự bất kỳ không xuất hiện trong mẫu đối sánh.

Khử các dòng giống nhau

SQL không tự động loại các bộ trùng nhau
• Tốn thời gian so sánh và sắp xếp.
• Sử dụng cho các truy vấn thống kê

Các hàm tập hợp

SQL cung cấp 5 hàm tập hợp:
• `SUM(<tên thuộc tính>)` - tính tổng các giá trị của thuộc tính

• MAX(<tên thuộc tính>) - tìm giá trị lớn nhất của thuộc tính
• MIN(<tên thuộc tính>) - tìm giá trị nhỏ nhất của thuộc tính
• AVG(<tên thuộc tính>) - tính giá trị trung bình của thuộc tính
• COUNT(*) - đếm số dòng của bảng
• COUNT(<tên thuộc tính>) - đếm các giá trị khác null của thuộc tính

Gom nhóm các bộ

GROUP BY - HAVING

```
SELECT <dstt nhóm> [, <dshth>]
FROM <dsb>
[WHERE <đk>]
GROUP BY <dstt nhóm>
[HAVING <đk nhóm>]
```

Trong đó
• <dstt nhóm>: danh sách thuộc tính gom nhóm
• <dshth>: danh sách các hàm tập hợp.
• <đk>: điều kiện chọn hoặc điều kiện kết.
• <đk nhóm>: điều kiện lựa chọn các nhóm.
Chú ý
• WHERE được thực hiện trước GROUP BY.
• HAVING chỉ xuất hiện khi có GROUP BY

Sắp xếp kết quả

ORDER BY

```
• SELECT <dstt>
FROM <dsb>
[WHERE <đk>]
ORDER BY <dstt sắp xếp>
```

• <dstt sắp xếp>: danh sách các cặp (tên thuộc tính, thứ tự sắp xếp).
• Thứ tự:
- ASC - tăng dần.
- DESC - giảm dần.
- Mặc định là ASC.

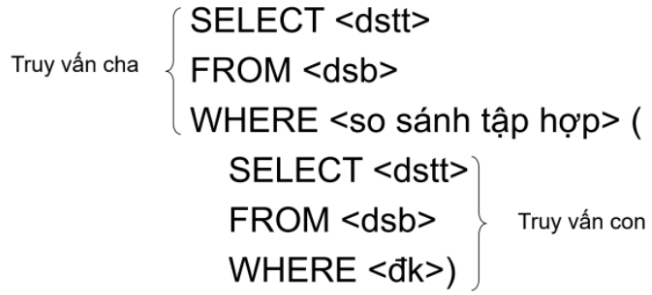
So sánh với NULL

NULL
• Không biết.
• Không sẵn sàng.
• Không thể áp dụng.
Tính toán và so sánh với NULL
• null + 3 => null.
• null > 3 => unknown.
SQL cung cấp 2 phép toán
• IS NULL.

• IS NOT NULL.

Truy vấn lồng

Truy vấn sử dụng các giá trị của truy vấn khác trong điều kiện so sánh.



Chỉ xuất hiện trong mệnh đề WHERE.

Phép toán

• IN - kiểm tra sự tồn tại của một giá trị trong một tập hợp.
• ALL - so sánh một giá trị với tất cả các giá trị của tập hợp.
• ANY - so sánh một giá trị với một giá trị nào đó của tập hợp.
• ALL, ANY được kết hợp với các phép toán so sánh {=, <, <=, >, >=,<>}.
• EXISTS - kiểm tra sự tồn tại của kết quả của một câu truy vấn.

Cú pháp

- `<tên thuộc tính> IN <truy vấn con>`
- `<tên thuộc tính> <phép toán so sánh> ALL <truy vấn con>`
- `<tên thuộc tính> <phép toán so sánh> ANY <truy vấn con>`
- `EXISTS <truy vấn con>`

Truy vấn lồng phân cấp

Mệnh đề WHERE của truy vấn con không tham chiếu đến thuộc tính của các bảng trong mệnh đề FROM của truy vấn cha.

Truy vấn con được thực hiện trước truy vấn cha.

Truy vấn lồng tương quan

Mệnh đề WHERE của truy vấn con tham chiếu đến thuộc tính của các bảng trong mệnh đề FROM của truy vấn cha.

Truy vấn con được thực hiện nhiều lần, mỗi lần ứng với một bộ của truy vấn cha.

Phép kết trong SQL

JOIN, INNER JOIN

• Dùng kết nối hai bảng trong mệnh đề FROM.
• SELECT <dstt>
FROM (<bảng 1> JOIN <bảng 2> ON <đkk>)

Các phép kết mở rộng:

• LEFT OUTER JOIN, LEFT JOIN.
• RIGHT OUTER JOIN, RIGHT JOIN.
• FULL OUTER JOIN, FULL JOIN.

3. Nâng cao

View

- View cũng là một table nhưng chức năng của nó là chỉ để đọc
- Tạo view:

```
CREATE [OR REPLACE] VIEW [db_name.]view_name [(column_list)]
AS
  select-statement;
```

Trong đó :

- OR REPLACE - thêm vào để ghi đè lên view cũ trùng tên nếu có.
- db_name. - tên cơ sở dữ liệu
- (column_list) - mặc định các column của view sẽ được lấy luôn bằng result set của select-statement, dùng khi muốn rename chúng.

- Thay đổi view :

```
ALTER VIEW view_name [(column_list)]
    AS select_statement;
```

- Đổi tên view :

```
RENAME TABLE view_name
TO new_view_name;
```

- Xóa view :

```
DROP VIEW [IF EXISTS] view_name;
```

Stored Procedure

- Stored Procedure được tạo ra nhằm thực hiện các lệnh của mysql theo một nhóm việc cụ thể thay vì thực hiện từng thao tác (insert,update,delete).
- Khởi tạo:

```
CREATE PROCEDURE <owner>.<procedure name>
     <Param> <datatype>
AS
     <Body>
Ví dụ:
CREATE PROCEDURE Users_GetUserInfo
    @login nvarchar(30)=null
```

```
AS
    SELECT * from users
    WHERE ISNULL(@login,login)=login
```

Prepared Statement

Prepared Statement (còn được gọi là câu lệnh được tham số hóa) chỉ đơn giản là một mẫu truy vấn SQL có chứa trình giữ chỗ thay vì các giá trị tham số thực tế.

Function

- Là đoạn chương trình kịch bản (programming scripts) với các câu lệnh SQL nhúng (embedded SQL) được lưu dưới dạng đã được biên dịch và thi hành thực tiếp bởi MySQL server.
- Khởi tạo:

```
CREATE FUNCTION name ([parameterlist]) RETURNS datatype [options]
sqlcode
```

Trigger

- Triggers là quá trình tự động thi hành các lệnhSQL hoặc hàm/thủ tục sau hoặc trước các lệnh INSERT, UPDATE, hoặc DELETE.
- Khởi tạo:

```
CREATE TRIGGER name BEFORE | AFTER INSERT | UPDATE | DELETE ON
tablename
FOR EACH ROW sql-code
```