

Cơ Sở Dữ Liệu

Chương 8. Ngôn ngữ SQL

Trần Hoài Thuận

Ngày 06 tháng 08 năm 2023



DEPARTMENT OF
INFORMATION TECHNOLOGY
GIA DINH UNIVERSITY



GIA DINH
UNIVERSITY



Nội Dung



Ngôn ngữ SQL

Giới thiệu

- SQL (Structured Query Language) là ngôn ngữ chuẩn cho các hệ quản trị cơ sở dữ liệu quan hệ thương mại ngày nay.



Giới thiệu

- SQL (**Structured Query Language**) là ngôn ngữ chuẩn cho các hệ quản trị cơ sở dữ liệu quan hệ thương mại ngày nay.
- Là ngôn ngữ giao tiếp dữ liệu cấp cao, người sử dụng chỉ cần đưa ra nội dung cần truy vấn.



Giới thiệu

- **SQL (Structured Query Language)** là ngôn ngữ chuẩn cho các hệ quản trị cơ sở dữ liệu quan hệ thương mại ngày nay.
- Là ngôn ngữ giao tiếp dữ liệu cấp cao, người sử dụng chỉ cần đưa ra nội dung cần truy vấn.
- SQL **sử dụng các khái niệm bảng, dòng, cột** để thay thế cho khái niệm tương ứng là quan hệ, bộ, thuộc tính trong mô hình dữ liệu quan hệ.





Các kiểu dữ liệu trong SQL

Các kiểu dữ liệu trong SQL

Các kiểu dữ liệu cơ bản cho một cột bao gồm:

- **Kiểu số** (numeric) bao gồm:

- + Các kiểu số nguyên với kích thước khác nhau (INTEGER hoặc INT, SMALLINT, TINYINT, BIT)
- + Các kiểu số thực với độ chính xác khác nhau (FLOAT, REAL)
- + Các kiểu dữ liệu dạng số thập phân (DECIMAL, NUMERIC),
được khai báo theo định dạng DECIMAL(i,j) hoặc DEC(i,j)
hoặc NUMERIC(i,j) với i là độ chính xác phần thập phân, j là
phần số lẻ.



Các kiểu dữ liệu trong SQL

Các kiểu dữ liệu cơ bản cho một cột bao gồm:

- **Kiểu số** (numeric) bao gồm:

- + Các kiểu số nguyên với kích thước khác nhau (INTEGER hoặc INT, SMALLINT, TINYINT, BIT)
- + Các kiểu số thực với độ chính xác khác nhau (FLOAT, REAL)
- + Các kiểu dữ liệu dạng số thập phân (DECIMAL, NUMERIC), được khai báo theo định dạng DECIMAL(i,j) hoặc DEC(i,j) hoặc NUMERIC(i,j) với i là độ chính xác phần thập phân, j là phần số lẻ.

- **Kiểu chuỗi** (character string) gồm:

- + Các kiểu chuỗi có chiều dài cố định (CHAR(n) hoặc CHARACTER(n)) với n là số ký tự của chuỗi.
- + Các kiểu chuỗi có chiều dài thay đổi (VARCHAR(n) hoặc CHAR VARYING(n) hoặc CHARACTER VARYING(n)) với n là số ký tự tối đa của chuỗi.



Các kiểu dữ liệu trong SQL

- Kiểu BOOLEAN có giá trị là TRUE hoặc FALSE.



Các kiểu dữ liệu trong SQL

- Kiểu BOOLEAN có giá trị là TRUE hoặc FALSE.
- Các kiểu dữ liệu liên quan đến ngày, giờ gồm:
 - + Kiểu DATE chứa ngày, tháng, năm
 - + Kiểu TIME chứa giờ, phút, giây
 - + Kiểu TIMESTAMP chứa ngày và giờ



Các kiểu dữ liệu trong SQL

Kiểu dữ liệu	Kích thước	Miền giá trị dữ liệu lưu trữ
Số nguyên		
Int	4 bytes	Từ -2,147,483,648 đến +2,147,483,648
Smallint	2 bytes	Từ -32,768 đến + 32,767
Tinyint	1 byte	Từ 0 đến 255
Bit	1 byte	0,1 hoặc Null
Các kiểu dữ liệu dạng số thập phân		
Decimal, Numeric	17 byte	Từ -10^38 đến +10^38
Các kiểu dữ liệu dạng số thực		
Float	8 bytes	Từ -1.79E + 308 đến +1.79E + 308
Real	4 bytes	Từ -1.79E + 308 đến +1.79E + 308



Các kiểu dữ liệu trong SQL

Kiểu dữ liệu	Kích thước	Miền giá trị dữ liệu lưu trữ
Các kiểu dữ liệu dạng chuỗi		
Char	N bytes	Từ 1 đến 8,000 ký tự, độ dài cố định
Varchar	N bytes	Từ 1 đến 8,000 ký tự, độ dài biến đổi
Text	N bytes	Từ 1 đến 2,147,483,647 ký tự
Nchar	2* n bytes	Unicode, từ 1 đến 4,000 ký tự, mỗi ký tự 2 bytes
Nvarchar	2* n bytes	Từ -10^38 đến +10^38
Ntext	2* n bytes	từ 1 đến 1,073,741,823 ký tự, mỗi ký tự 1 byte
Các kiểu dữ liệu dạng ngày giờ		
datetime	8 bytes	Từ 01/01/1753 đến 31/12/9999
smalldatetime	4 bytes	Từ 01/01/1900 đến 06/06/2079
Các kiểu dữ liệu dạng chuỗi nhị phân		
Image	N byte	Từ 1 đến 2,147,483,647 bytes
Binary	N byte	Từ 1 đến 8,000





Các lệnh SQL cơ bản

Các lệnh SQL cơ bản

Câu lệnh định nghĩa dữ liệu trong SQL là lệnh **CREATE**. Được sử dụng để tạo lược đồ (schema), tạo bảng (table), tạo khung nhìn (view) và ràng buộc (assertion, trigger).



Lệnh tạo bảng

Lệnh **CREATE TABLE** dùng để tạo một bảng bằng cách đưa ra tên bảng, danh sách các cột và các RBTV trên bảng đó.

Tạo bảng

```
CREATE TABLE <Tên_bảng> (
    <Tên_cột> <Kiểu_dữ_liệu> [<RBTV>],
    <Tên_cột> <Kiểu_dữ_liệu> [<RBTV>],
    ...
    [<RBTV>]
);
```



Ví dụ - Tạo bảng

```
CREATE TABLE NHANVIEN (
    MANV CHAR(9),
    HONV VARCHAR(10),
    TENLOT VARCHAR(20),
    TENNV VARCHAR(10),
    NGSINH DATETIME,
    DCHI VARCHAR(50),
    PHAI CHAR(3),
    LUONG INT,
    MA_NQL CHAR(9),
    PHG INT
);
```



Lệnh tạo bảng

<RBTV> gồm:

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

Đặt tên cho RBTV

CONSTRAINT <Ten_RBTV> <RBTV>



Ví dụ - RBTV

```
CREATE TABLE NHANVIEN (
    MANV CHAR(9) PRIMARY KEY,
    HONV VARCHAR(10) NOT NULL,
    TENLOT VARCHAR(20) NOT NULL,
    TENNV VARCHAR(10) NOT NULL,
    NGSINH DATETIME,
    DCHI VARCHAR(50),
    PHAI CHAR(3) CHECK (PHAI IN ('Nam', 'Nu')),
    LUONG INT DEFAULT (10000),
    MA_NQL CHAR(9),
    PHG INT
)
```



Ví dụ - RBTV

```
CREATE TABLE PHONGBAN (
    TENPB VARCHAR(20) UNIQUE,
    MAPHG INT NOT NULL,
    TRPHG CHAR(9),
    NG_NHANCHUC DATETIME DEFAULT (CURDATE())
)
```

```
CREATE TABLE PHANCONG (
    MA_NVIENTRANH CHAR(9) FOREIGN KEY (MA_NVIENTRANH)
    REFERENCES NHANVIEN(MANV),
    SODA INT REFERENCES DEAN(MADA),
    THOIGIAN DECIMAL(3,1)
)
```



Ví dụ - Đặt tên cho RBTV

```
CREATE TABLE NHANVIEN (
    HONV VARCHAR(10) NOT NULL,
    TENLOT VARCHAR(20) NOT NULL,
    TENNV VARCHAR(10) NOT NULL,
    MANV CHAR(9),
    NGSINH DATETIME,
    DCHI VARCHAR(50),
    PHAI CHAR(3),
    LUONG INT DEFAULT (10000),
    MA_NQL CHAR(9),
    PHG INT,
    CONSTRAINT PK_NHANVIEN PRIMARY KEY(MANV),
    CONSTRAINT NV_PHAI_CHK CHECK (PHAI IN ('Nam', 'Nu'))
```



Ví dụ - Đặt tên cho RBTV

```
CREATE TABLE PHANCONG (
    MA_NVIENTHONG CHAR(9),
    SODA INT,
    THOIGIAN DECIMAL(3,1),
    CONSTRAINT PC_MANVIEN_SODA_PK PRIMARY KEY
    (MA_NVIENTHONG, SODA),
    CONSTRAINT PC_MANVIEN_FK FOREIGN KEY (MA_NVIENTHONG)
    REFERENCES NHANVIEN(MANV),
    CONSTRAINT PC_SODA_FK FOREIGN KEY (SODA)
    REFERENCES DEAN(MADA)
)
```



Lệnh sửa bảng

Lệnh **ALTER TABLE** dùng để thay đổi cấu trúc hoặc ràng buộc của một bảng. Các lệnh gồm có: thêm cột, xoá cột, mở rộng cột, thêm ràng buộc, xoá ràng buộc.

Thêm cột

```
ALTER TABLE <Tên_bảng> ADD <Tên_cột> <Kiểu_dữ_liệu> [<RBTV>]
```

Xoá cột

```
ALTER TABLE <Tên_bảng> DROP COLUMN <Tên_cột>
```

Mở rộng cột

```
ALTER TABLE <Tên_bảng> MODIFY <Tên_cột> <Kiểu_dữ_liệu_mới>
```



Lệnh sửa bảng

Lệnh **ALTER TABLE** dùng để thay đổi cấu trúc hoặc ràng buộc của một bảng. Các lệnh gồm có: thêm cột, xoá cột, mở rộng cột, thêm ràng buộc, xoá ràng buộc.

Thêm RBTV

```
ALTER TABLE <Tên_bảng> ADD  
    CONSTRAINT <Ten_RBTV> <RBTV>,  
    CONSTRAINT <Ten_RBTV> <RBTV>,  
    ...
```

Xoá RBTV

```
ALTER TABLE <Tên_bảng> DROP <Tên_RBTV>
```



Ví dụ - Thay đổi cấu trúc bảng

Ví dụ

Thêm cột NGHENGHIEP vào bảng NHANVIEN

```
ALTER TABLE NHANVIEN ADD NGHENGHIEP CHAR(20)
```

Xoá cột NGHENGHIEP ra khỏi bảng NHANVIEN

```
ALTER TABLE NHANVIEN DROP COLUMN NGHENGHIEP
```

Mở rộng chiều dài cột NGHENGHIEP thành 50

```
ALTER TABLE NHANVIEN MODIFY NGHENGHIEP CHAR(50)
```



Ví dụ - Thay đổi RBTV

```
CREATE TABLE PHONGBAN (
    TENPB VARCHAR(20),
    MAPHG INT NOT NULL,
    TRPHG CHAR(9),
    NG_NHANCHUC DATETIME
)
```

Thêm các ràng buộc vào bảng PHONGBAN

```
ALTER TABLE PHONGBAN ADD
CONSTRAINT PB_MAPHG_PK PRIMARY KEY (MAPHG),
CONSTRAINT PB_TRPHG FOREIGN KEY (TRPHG) REFERENCES NHANVIEN(MANV),
CONSTRAINT PB_NGNHANCHUC_DF DEFAULT (CURDATE()) FOR (NG_NHANCHUC),
CONSTRAINT PB_TENPB_UNI UNIQUE (TENPB)
```



Lệnh xoá bảng

Lệnh **DROP TABLE** dùng để xoá cấu trúc của một bảng. Tất cả dữ liệu của bảng cũng bị xoá.

Xoá bảng

```
DROP TABLE <Tên_bảng>
```

Ví dụ

```
DROP TABLE NHANVIEN  
DROP TABLE PHONGBAN  
DROP TABLE PHANCONG
```



Lệnh INSERT

Lệnh **INSERT** dùng để thêm 1 hay nhiều dòng vào bảng.

Cú pháp (thêm 1 dòng)

```
INSERT INTO <tên bảng>(<ds các thuộc tính>) VALUES (<ds các giá trị>)
```

Ví dụ

```
INSERT INTO NHANVIEN(HONV, TENLOT, TENNV, MANV)
VALUES ('Le', 'Van', 'A', '052')
```

```
INSERT INTO NHANVIEN(HONV, TENLOT, TENNV, MANV, DCHI)
VALUES ('Le', 'Van', 'A', '052', NULL)
```

```
INSERT INTO NHANVIEN
VALUES ('Le', 'Van', 'A', '052', '1/3/1992', '98 HV', 'Nu', '300', 4)
```



Lệnh INSERT

Nhận xét:

- Thứ tự các giá trị phải trùng với thứ tự các cột
- Có thể thêm giá trị NULL ở những thuộc tính không là khóa chính và NOT NULL
- Câu lệnh INSERT sẽ gặp lỗi nếu vi phạm RBTV
 - Khóa chính
 - Tham chiếu
 - NOT NULL - các thuộc tính có ràng buộc NOT NULL bắt buộc phải có giá trị



Lệnh INSERT

Cú pháp (thêm nhiều dòng)

```
INSERT INTO <tên bảng>(<ds các thuộc tính>) <câu truy vấn con>
```

Ví dụ

```
CREATE TABLE THONGKE_PB (
    TENPHG VARCHAR(20),
    SL_NV INT,
    LUONG_TC INT
)
```

```
INSERT INTO THONGKE_PB(TENPHG, SL_NV, LUONG_TC)
SELECT TENPHG, COUNT(MANV), SUM(LUONG) FROM NHANVIEN, PHONGBAN
WHERE PHG=MAPHG GROUP BY TENPHG
```



Lệnh DELETE

Lệnh **DELETE** dùng để xoá các dòng trong bảng.

Cú pháp

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

Ví dụ

```
DELETE FROM NHANVIEN WHERE HONV='Tran'  
DELETE FROM NHANVIEN WHERE MANV='345345345'  
DELETE FROM NHANVIEN  
DELETE FROM NHANVIEN WHERE PHG IN (  
    SELECT MAPHG  
    FROM PHONGBAN  
    WHERE TENPHG='Nghien cuu')
```



Lệnh DELETE

Nhận xét:

- Số lượng số dòng bị xóa phụ thuộc vào điều kiện ở mệnh đề WHERE
- Nếu không chỉ định điều kiện ở mệnh đề WHERE, tất cả các dòng trong bảng sẽ bị xóa
- Lệnh DELETE có thể gây ra vi phạm RB tham chiếu
 - Không cho xóa
 - Xóa luôn những dòng có giá trị đang tham chiếu đến
 - Đặt NULL cho những giá trị tham chiếu



Lệnh DELETE

MANV	HONV	TENLOT	TENNVL	NGSINH	DCHI	PHAI	LUONG	MA_NQL	PHG
333445555	Nguyen	Thanh	Tung	12/08/1955	638 NVC Q5	Nam	40000	888665555	5
987987987	Nguyen	Manh	Hung	09/15/1962	Ba Ria VT	Nam	38000	333445555	5
453453453	Tran	Thanh	Tam	07/31/1972	543 MTL Q1	Nu	25000	333445555	5
999887777	Bui	Ngoc	Hang	07/19/1968	33 NTH Q1	Nu	38000	987654321	4
987654321	Le	Quynh	Nhu	07620/1951	219 TD Q3	Nu	43000	888665555	4
987987987	Tran	Hong	Quang	04/08/1969	980 LHP Q5	Nam	25000	987654321	4
888665555	Pham	Van	Vinh	11/10/1945	450 TV HN	Nam	55000	NULL	1

MA_NVIENTH	SODA	THOIGIAN
333445555	10	10.0
888665555	20	20.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
453453453	1	20.0



Lệnh DELETE

TENPHG	MAPHG	MA_NVIENTHUC	NG_NHANCHUC
Nghien cuu	5	333445555	05/22/1988
Dieu hanh	4	987987987	01/01/1995
Quan ly	1	888665555	06/19/1981

MANV	HONV	TENLOT	TENNVL	NGSINH	DCHI	PHAI	LUONG	MA_NQL	PHG
333445555	Nguyen	Thanh	Tung	12/08/1955	638 NVC Q5	Nam	40000	888665555	NULL
987987987	Nguyen	Manh	Hung	09/15/1962	Ba Ria VT	Nam	38000	333445555	NULL
453453453	Tran	Thanh	Tam	07/31/1972	543 MTL Q1	Nu	25000	333445555	NULL
999887777	Bui	Ngoc	Hang	07/19/1968	33 NTH Q1	Nu	38000	987654321	4
987654321	Le	Quynh	Nhu	07620/1951	219 TD Q3	Nu	43000	888665555	4
987987987	Tran	Hong	Quang	04/08/1969	980 LHP Q5	Nam	25000	987654321	4
888665555	Pham	Van	Vinh	11/10/1945	450 TV HN	Nam	55000	NULL	1



Lệnh UPDATE

Lệnh **UPDATE** dùng để thay đổi giá trị của thuộc tính cho các dòng của bảng.

Cú pháp

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



Lệnh UPDATE

Ví dụ

Cập nhật ngày sinh của nhân viên có mã nhân viên ‘333445555’ là ’08/12/1992’

```
UPDATE NHANVIEN SET NGSINH='08/12/1992' WHERE MANV='333445555'
```

Tăng toàn bộ lương của nhân viên lên 10

```
UPDATE NHANVIEN SET LUONG=LUONG*1.1
```

Với đề án có mã số 10, hãy thay đổi nơi thực hiện đề án thành ‘Vung Tau’ và phòng ban phụ trách là phòng 5

```
UPDATE DEAN
```

```
SET DIADIEM_DA='Vung Tau', PHONG=5 WHERE MADA=10
```



Lệnh UPDATE

Nhận xét:

- Những dòng thỏa điều kiện tại mệnh đề WHERE sẽ được cập nhật giá trị mới
- Nếu không chỉ định điều kiện ở mệnh đề WHERE, tất cả các dòng trong bảng sẽ bị cập nhật
- Lệnh UPDATE có thể gây ra vi phạm RB tham chiếu
 - Không cho xóa
 - Xóa luôn những dòng có giá trị đang tham chiếu đến





Truy vấn dữ liệu

Truy vấn dữ liệu - Mệnh đề SELECT

- Là hành động **rút trích dữ liệu** thỏa một số điều kiện nào đó, được dựa trên các **phép toán đại số quan hệ** + một số bổ sung.
- Câu lệnh truy vấn thường gồm có 3 mệnh đề **SELECT, FROM** và **WHERE** được thể hiện như sau:

Cú pháp

```
SELECT <danh sách các cột>
FROM <danh sách các bảng>
WHERE <điều kiện>
```

<danh sách các cột> : Tên các cột cần được hiển thị ở bảng kết quả

<danh sách các bảng> : Tên các bảng liên quan đến câu truy vấn

<điều kiện> : Biểu thức boolean xác định dòng nào sẽ được rút trích



Mệnh đề SELECT

Ví dụ 1

```
# Lấy mã sinh viên + tên sinh viên trong bảng sinh viên  
SELECT MASV, TENSV  
FROM SINHVIEN;
```

Ví dụ 2

```
# Lấy mã giáo viên, họ tên giáo viên có mức lương > 2000  
SELECT MAGV, HOTEN  
FROM GIAOVIEN  
WHERE LUONG > 2000;
```

```
# Lấy toàn bộ thông tin của giáo viên có mức lương > 2000  
SELECT *  
FROM GIAOVIEN  
WHERE LUONG > 2000;
```



Mệnh đề SELECT

SQL và DSQH:



SELECT L
FROM R —————— $\pi_L(\sigma_C(R))$
WHERE C



Mệnh đề SELECT

Ví dụ:

Lấy tất cả các cột của
quan hệ kết quả

```
SELECT *
  FROM NHANVIEN
 WHERE PHG=5
```

MANV	HONV	TENLOT	TENNVL	NGSINH	DCHI	PHAI	LUONG	MA_NQL	PHG
333445555	Nguyen	Thanh	Tung	12/08/1955	638 NVC Q5	Nam	40000	888665555	5
987987987	Nguyen	Manh	Hung	09/15/1962	Ba Ria VT	Nam	38000	333445555	5

$\sigma_{\text{PHG}=5}(\text{NHANVIEN})$



Mệnh đề SELECT

Ví dụ:

```
SELECT MANV, HONV, TENLOT, TENNV  
FROM NHANVIEN  
WHERE PHG=5 AND PHAI='Nam'
```

MANV	HONV	TENLOT	TENNV
333445555	Nguyen	Thanh	Tung
987987987	Nguyen	Manh	Hung

$$\pi_{\text{MANV}, \text{HONV}, \text{TENLOT}, \text{TENNV}}(\sigma_{\text{PHG}=5 \wedge \text{PHAI}=\text{'Nam'}}(\text{NHANVIEN}))$$


Mệnh đề SELECT

Ví dụ:

```
SELECT MANV, HONV AS HO, TENLOT AS 'TEN LOT', TENNV AS TEN  
FROM NHANVIEN  
WHERE PHG=5 AND PHAI='Nam'
```

MANV	HO	TEN LOT	TEN
333445555	Nguyen	Thanh	Tung
987987987	Nguyen	Manh	Hung

$$\rho_{MANV, HO, TEN\ LOT, TEN}(\pi_{MANV, HONV, TENLOT, TENNV}(\sigma_{PHG=5 \wedge PHAI='Nam'}(NHANVIEN)))$$


Mệnh đề SELECT

Ví dụ:

```
SELECT MANV, HONV + ' ' + TENLOT + ' ' + TENNV AS 'HO TEN'  
FROM NHANVIEN  
WHERE PHG=5 AND PHAI='Nam'
```

MANV	HO TEN
333445555	Nguyen Thanh Tung
987987987	Nguyen Manh Hung

$$\rho_{MANV, HO\ TEN}(\pi_{MANV, HONV+TENLOT+TENVN}(\sigma_{PHG=5 \wedge PHAI='Nam'}(NHANVIEN)))$$


Mệnh đề SELECT

Ví dụ:

```
SELECT MANV, LUONG*1.1 AS 'LUONG10%'  
FROM NHANVIEN  
WHERE PHG=5 AND PHAI='Nam'
```

MANV	LUONG10%
333445555	33000
987987987	27500

$$\rho_{MANV,LUONG10\%}(\pi_{MANV,LUONG*1.1}(\sigma_{PHG=5 \wedge PHAI='Nam'}(NHANVIEN)))$$


Mệnh đề SELECT

Ví dụ: Dùng DISTINCT để loại bỏ các dòng trùng nhau

```
SELECT LUONG  
FROM NHANVIEN  
WHERE PHG=5 AND PHAI='Nam'
```

```
SELECT DISTINCT LUONG  
FROM NHANVIEN  
WHERE PHG=5 AND PHAI='Nam'
```

LUONG	
30000	- Tốn chi phí
25000	- Người dùng muốn thấy
38000	

LUONG
30000
25000
38000



Mệnh đề SELECT

Ví dụ: Cho biết MANV và TENNV làm việc ở phòng ‘Nghien cuu’

$R1 \leftarrow NHANVIEN \bowtie_{PHG=MAPHG} PHONGBAN$

$KQ \leftarrow \pi_{MANV, TENNV} (\sigma_{TENPHG='Nghien cuu'}(R1))$

```
SELECT MANV, TENNV  
FROM NHANVIEN, PHONGBAN  
WHERE PHG=MAPHG      AND      TENPHG='Nghien cuu'
```



Mệnh đề WHERE

SELECT <danh sách các cột>

FROM <danh sách các bảng>

WHERE <điều kiện> có giá trị là hoặc đúng (true) hoặc sai (false)

- Nếu điều kiện này chỉ liên quan đến một quan hệ thì gọi là điều kiện chọn, nếu điều kiện liên quan đến từ hai quan hệ trở lên thì gọi là điều kiện kết.
- Các điều kiện chọn và điều kiện kết có thể phối hợp với nhau bởi các toán tử logic (AND, OR, NOT) để tạo nên những biểu thức logic phức tạp hơn.



Truy vấn dữ liệu - Mệnh đề WHERE

Ví dụ:

```
SELECT MANV, TENNV  
FROM NHANVIEN, PHONGBAN  
WHERE TENPHG='Nghien cuu' AND PHG=MAPHG
```

Biểu thức luận lý

The diagram illustrates the logical expression 'TENPHG='Nghien cuu' AND PHG=MAPHG'. Two dashed boxes enclose the conditions 'TENPHG='Nghien cuu'' and 'PHG=MAPHG'. Dotted arrows point from these boxes to the word 'TRUE' below them, indicating that both conditions must be true for the entire WHERE clause to evaluate to TRUE.

Dộ ưu tiên

```
SELECT MANV, TENNV  
FROM NHANVIEN, PHONGBAN  
WHERE (TENPHG='Nghien cuu' OR TENPHG='Quan ly') AND PHG=MAPHG
```



Truy vấn dữ liệu - Mệnh đề WHERE

Ví dụ: LIKE

```
SELECT MANV, TENNV  
FROM NHANVIEN  
WHERE DCHI LIKE 'Nguyen _ _ _'
```

Ký tự bất kỳ

Chuỗi bất kỳ



Truy vấn dữ liệu - Mệnh đề WHERE

Ví dụ: BETWEEN

```
SELECT MANV, TENNV  
FROM NHANVIEN  
WHERE LUONG>=20000 AND LUONG<=30000
```

```
SELECT MANV, TENNV  
FROM NHANVIEN  
WHERE LUONG BETWEEN 20000 AND 30000
```



Truy vấn dữ liệu - Mệnh đề WHERE

Ví dụ: NOT BETWEEN

```
SELECT MANV, TENNV  
FROM NHANVIEN  
WHERE LUONG NOT BETWEEN 20000 AND 30000
```



Truy vấn dữ liệu - Mệnh đề WHERE

Ví dụ: IN, OR

```
SELECT MANV, TENNV  
FROM NHANVIEN  
WHERE PHG IN (4,5)
```

```
SELECT MANV, TENNV  
FROM NHANVIEN  
WHERE PHG = 4 OR PHG=5
```



Truy vấn dữ liệu - Mệnh đề WHERE

Ví dụ: NOT IN

```
SELECT MANV, TENNV  
FROM NHANVIEN  
WHERE PHG NOT IN (4,5)
```



Truy vấn dữ liệu - Mệnh đề WHERE

Ví dụ: NULL sử dụng trong trường hợp

- + Không biết (value unknown)
- + Không thể áp dụng (value inapplicable)
- + Không tồn tại (value with held)



Truy vấn dữ liệu - Mệnh đề WHERE

Ví dụ: NULL, NOT NULL

```
SELECT MANV, TENNV  
FROM NHANVIEN  
WHERE MA_NQL IS NULL
```

```
SELECT MANV, TENNV  
FROM NHANVIEN  
WHERE MA_NQL IS NOT NULL
```



STORE PROCEDURE

- Tương tự như khái niệm function trong các ngôn ngữ lập trình.
Store procedure là một **thủ tục** (**phương thức**), hay là một tập hợp các câu lệnh SQL thực hiện công việc nào đó.
- Bao gồm 3 loại tham số: **IN**, **OUT**, **INOUT**.



STORE PROCEDURE

Cú pháp

```
CREATE PROCEDURE sp_name ([ IN | OUT | INOUT ] param_name type)
BEGIN
    [sql_statements]
END;
```

Thực thi (gọi store procedure)

```
CALL sp_name;
```



STORE PROCEDURE

Ví dụ 1

```
DELIMITER //
CREATE PROCEDURE in_ds_nhanvien()
BEGIN
    SELECT * FROM NHANVIEN;
END
//
#CALL in_ds_nhanvien();
```



STORE PROCEDURE

Ví dụ 2

```
DELIMITER //
CREATE PROCEDURE tim_ds_nhanvien(str VARCHAR(10))
BEGIN
    SELECT * FROM NHANVIEN
    WHERE TENNV LIKE CONCAT('%',str,'%');
END
//
CALL tim_ds_nhanvien('A');
```



STORE PROCEDURE

Cú pháp xoá store procedure

```
DROP PROCEDURE [IF EXISTS] [schema_name.]sp_name;
```

- Trong đó sp_name chính là tên store procedure mà bạn đã tạo.
schema_name chính là schema mà trigger này thuộc về, mặc định nó là schema hiện tại mà bạn đang kết nối.
- Lệnh IF EXISTS sẽ giúp việc xóa store procedure được an toàn hơn, nghĩa là nó chỉ xóa chỉ khi nào procedure đó tồn tại.

Ví dụ

```
DROP PROCEDURE IF EXISTS 'DB'.'in_ds_giaovien';;
```



TRIGGER

- Trigger là một **store procedure** không có tham số. Trigger thực thi một cách tự động.
- Gắn với sự kiện xảy ra trên một table nào đó. Ví dụ như: **INSERT, UPDATE, DELETE**.
- Sử dụng các từ khoá **OLD, NEW** để lấy các record tương ứng.
- Trong trigger **INSERT** chỉ **NEW.col_name** là có thể được sử dụng.
- Trong trigger **DELETE** chỉ **OLD.col_name** là có thể được sử dụng.
- Trong trigger **UPDATE** cả **OLD.col_name** và **NEW.col_name** đều có thể được dùng.



TRIGGER

Cú pháp tạo trigger

```
CREATE TRIGGER trigger_name  
trigger_time trigger_event  
ON table_name FOR EACH ROW  
trigger_body
```

Trong đó

trigger_time: { BEFORE | AFTER }

trigger_event: { INSERT | UPDATE | DELETE }



TRIGGER

Ví dụ

```
DELIMITER //
CREATE TRIGGER them_nhanvien
AFTER INSERT
ON NHANVIEN FOR EACH ROW
BEGIN
    INSERT INTO PHANCONG(MA_NVIEN, SODA, THOIGIAN)
    VALUES (NEW.MANV, 1, 10.0);
END
//
```



TRIGGER

Cú pháp xoá trigger

```
DROP TRIGGER [IF EXISTS] [schema_name.]trigger_name;
```

- Trong đó trigger_name chính là tên trigger mà bạn đã tạo.
schema_name chính là schema mà trigger này thuộc về, mặc định nó là schema hiện tại mà bạn đang kết nối.
- Lệnh IF EXISTS sẽ giúp việc xóa trigger được an toàn hơn, nghĩa là nó chỉ xóa chỉ khi nào trigger đó tồn tại.

Ví dụ

```
DROP TRIGGER them_nhanvien;
```



Cơ Sở Dữ Liệu

Chương 8. Ngôn ngữ SQL

Trần Hoài Thuận

Ngày 06 tháng 08 năm 2023



DEPARTMENT OF
INFORMATION TECHNOLOGY
GIA DINH UNIVERSITY

