



TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA HỆ THỐNG THÔNG TIN

MÔN CƠ SỞ DỮ LIỆU - IT004

CHƯƠNG 4: NGÔN NGỮ SQL (Structured Query Language)

ThS. TẠ VIỆT PHƯƠNG
phuongtv@uit.edu.vn

Nội dung

I. Giới thiệu

II. Ngôn ngữ định nghĩa dữ liệu

III. Ngôn ngữ thao tác dữ liệu

IV. Ngôn ngữ truy vấn dữ liệu



GIỚI THIỆU

Ngôn ngữ truy vấn có cấu trúc SQL

- **Structured Query Language**

- Là ngôn ngữ chuẩn để truy vấn và thao tác trên CSDL quan hệ
- Là ngôn ngữ phi thủ tục
- Khởi nguồn của SQL là SEQUEL - *Structured English Query Language*, năm 1974).

- **Các chuẩn SQL**

- SQL-86
- SQL-89
- SQL-92 (SQL2)
- SQL:1999 (SQL3); ...
- Hiện tại: SQL:2019, SQL:2023

Ngôn ngữ truy vấn có cấu trúc SQL

Bao gồm:

- **Ngôn ngữ định nghĩa dữ liệu** (*Data Definition Language - DDL*): cho phép khai báo, định nghĩa **cấu trúc bảng, các mối quan hệ và các ràng buộc**.
- **Ngôn ngữ thao tác dữ liệu** (*Data Manipulation Language - DML*): cho phép **thêm, xóa, sửa** dữ liệu.
- **Ngôn ngữ truy vấn dữ liệu** (*Data Query Language – DQL*): cho phép **truy vấn** dữ liệu.
- **Ngôn ngữ điều khiển dữ liệu** (*Data Control Language – DCL*): khai báo **bảo mật thông tin, cấp quyền và thu hồi quyền khai thác** trên cơ sở dữ liệu, không thay đổi cấu trúc hay nội dung dữ liệu.

Ngôn ngữ truy vấn có cấu trúc SQL

Lưu ý:

- Nhiều tài liệu và chuẩn SQL coi DQL là một phần của DML, bởi vì cả hai đều tác động trực tiếp lên dữ liệu.
- Tuy nhiên, DQL thường được tách riêng trong một số nguồn khác vì chỉ dùng để truy vấn dữ liệu, không sửa đổi nội dung cơ sở dữ liệu.
- DDL (Data Definition Language) được sử dụng để định nghĩa và quản lý cấu trúc của cơ sở dữ liệu (như bảng, chỉ mục, và ràng buộc), vì thế nó thay đổi cấu trúc cơ sở dữ liệu.
- DML (Data Manipulation Language) được sử dụng để thao tác và quản lý dữ liệu trong các bảng đã được định nghĩa. DML không thay đổi cấu trúc mà chỉ tác động đến nội dung dữ liệu

Ngôn ngữ truy vấn có cấu trúc SQL

- SQL sử dụng các thuật ngữ tương đương với thuật ngữ về mô hình dữ liệu quan hệ như sau:

| SQL | Mô hình dữ liệu quan hệ |
|--------------------|-------------------------|
| Bảng (Table) | Quan hệ |
| Cột (Column) | Thuộc tính |
| Dòng (Row, Record) | Bộ |

NGÔN NGỮ ĐỊNH NGHĨA DỮ LIỆU

2. Ngôn ngữ định nghĩa dữ liệu

Là ngôn ngữ mô tả

- Lược đồ cho mỗi quan hệ
- Miền giá trị tương ứng của từng thuộc tính
- Ràng buộc toàn vẹn
- Chỉ mục trên mỗi quan hệ

Bao gồm: Tạo, sửa và xóa bảng

2. Ngôn ngữ định nghĩa dữ liệu

1. Lệnh tạo bảng (CREATE)

Cú pháp

Một số kiểu dữ liệu

2. Lệnh sửa cấu trúc bảng (ALTER)

Thêm thuộc tính

Sửa kiểu dữ liệu của thuộc tính

Xóa thuộc tính

Thêm ràng buộc toàn vẹn

Xóa ràng buộc toàn vẹn

3. Lệnh xóa bảng (DROP)

2.1. Lệnh tạo bảng

Cú pháp

CREATE TABLE <tên_bảng>

(

<tên_cột1> <kiểu_dữ_liệu> [NOT NULL],

<tên_cột2> <kiểu_dữ_liệu> [NOT NULL],

...

<tên_cộtn> <kiểu_dữ_liệu> [NOT NULL],

khai báo khóa chính, khóa ngoại, ràng buộc

)

2.1. Lệnh tạo bảng

Một số kiểu dữ liệu cơ bản

| Kiểu dữ liệu | SQL Server |
|--------------|--|
| Chuỗi ký tự | varchar(n), char(n), nvarchar(n), nchar(n) varchar(max), nvarchar(max) |
| Số | tinyint, smallint, int, bigint, numeric(m,n), decimal(m,n) - m chữ số, n chữ số thập phân, float, real, smallmoney, money |
| Ngày tháng | smalldatetime, datetime |
| Luận lý | Bit (0-false, 1-true, NULL) |

2.1. Lệnh tạo bảng

Datetime, Smalldatetime, Date, Time, Datetime2

- Datetime: 1753-01-01 đến 9999-12-31 23:59:59.997
- Smalldatetime: 1900-01-01 đến 2079-06-06 23:59:59
- Date: 0001-01-01 đến 9999-12-31
- Time: 00:00:00.0000000 đến 23:59:59.9999999
- Datetime2: 0001-01-01 đến 9999-12-31 23:59:59.9999999
- Datetimeoffset: 0001-01-01 đến 9999-12-31, với múi giờ từ -14:00 đến +14:00

Trong SQL Server, lưu ngày tháng năm trước công nguyên (BC) như thế nào?

Tìm hiểu thêm.

2.1. Lệnh tạo bảng

Ví dụ:

Lược đồ CSDL quản lý bán hàng gồm có các quan hệ sau:

KHACHHANG (MAKH, HOTEN, DCHI, SODT, NGSINH,
DOANHISO, NGDK, CMND)

NHANVIEN (MANV,HOTEN, NGVL, SODT)

SANPHAM (MASP,TENSP, DVT, NUOCSX, GIA)

HOADON (SOHD, NGHD, MAKH, MANV, TRIGIA)

CTHD (SOHD,MASP,SL)

2.1. Lệnh tạo bảng

CREATE TABLE KHACHHANG

(

| | |
|-----------|----------------------|
| MAKH | char(4) primary key, |
| HOTEN | varchar(40), |
| DCHI | varchar(50), |
| SODT | varchar(20), |
| NGSINH | smalldatetime, |
| DOANHSONG | money, |
| NGDK | smalldatetime, |
| CMND | char(9) |

)

2.1. Lệnh tạo bảng

CREATE TABLE CTHD

(

SOHD int foreign key references HOADON(SOHD),

MASP char(4) foreign key references SANPHAM(MASP),

SL int, **constraint PK_CTHD primary key** (SOHD, MASP)

)

2.2. Lệnh sửa cấu trúc bảng

❖ Thêm thuộc tính

ALTER TABLE tênbảng ADD tên cột kiểu dữ liệu

— Ví dụ: thêm cột Ghi_chu vào bảng khách hàng

ALTER TABLE KHACHHANG ADD GHI_CHU varchar(20)'

❖ Sửa kiểu dữ liệu thuộc tính

ALTER TABLE tênbảng ALTER COLUMN tên cột kiểu dữ liệu mới

❖ Lưu ý:

Không phải sửa bất kỳ kiểu dữ liệu nào cũng được: độ dài dữ liệu, dữ liệu hiện có...

Lỗi: String or binary data would be truncated in table '.....', column '.....'. Truncated value: ''.

2.2. Lệnh sửa cấu trúc bảng

- **Ví dụ:** Sửa Cột Ghi_chu thành kiểu dữ liệu varchar(50)
ALTER TABLE KHACHHANG ALTER COLUMN GHI_CHU varchar(50)
- Nếu sửa kiểu dữ liệu của cột Ghi_chu thành varchar(5), mà trước đó đã nhập giá trị cho cột Ghi_chu có độ dài hơn 5 ký tự thì không được phép.
- Hoặc sửa từ kiểu chuỗi ký tự sang kiểu số, ...

❖ **Xóa thuộc tính**

ALTER TABLE tên_bảng DROP COLUMN tên_cột

- Ví dụ: xóa cột Ghi_chu trong bảng KHACHHANG
ALTER TABLE KHACHHANG DROP COLUMN Ghi_chu

2.2. Lệnh sửa cấu trúc bảng

❖ Thêm ràng buộc toàn vẹn:

**ALTER TABLE <tên_bảng>
ADD CONSTRAINT
<tên_ràng_buộc>**

UNIQUE tên_cột

PRIMARY KEY (tên_cột)

FOREIGN KEY (tên_cột)

REFERENCES tên_bảng
(cột_là_khóa_chính)

CHECK (tên_cột điều_kiện)

2.2. Lệnh sửa cấu trúc bảng

- ❖ Ràng buộc toàn vẹn trong SQL được chia làm 2 loại chính:
 - Loại đơn giản: Sử dụng **Constraint** để mô tả
 - Loại phức tạp: Sử dụng **Trigger** để thực hiện
- ❖ Có thể khai báo RBTV ở mức cột hoặc mức bảng

2.2. Lệnh sửa cấu trúc bảng

- ❖ **Các loại RBTV đơn giản:**
 - Kiểm tra duy nhất: Primary Key, Unique
 - Kiểm tra dữ liệu được rỗng/khác rỗng: NULL/NOT NULL
 - Kiểm tra tồn tại (ràng buộc khóa ngoại): Foreign Key ... References...
 - Kiểm tra miền giá trị: Check, Default (Tạo giá trị mặc định)

2.2. Lệnh sửa cấu trúc bảng

• Ví dụ

- ALTER TABLE **NHANVIEN** ADD CONSTRAINT PK_NV
PRIMARY KEY (MANV)
- ALTER TABLE **CTHD** ADD CONSTRAINT FK_CT_SP
FOREIGN KEY (MASP) REFERENCES SANPHAM(MASP)
- ALTER TABLE SANPHAM ADD CONSTRAINT CK_GIA
CHECK (GIA >=500)
- ALTER TABLE KHACHHANG ADD CONSTRAINT UQ_KH
UNIQUE (CMND)

2.2. Lệnh sửa cấu trúc bảng

• Ví dụ

- ALTER TABLE CTHD ADD CONSTRAINT CK_SL CHECK (SL BETWEEN 1 AND 100);
- ALTER TABLE SANPHAM ADD CONSTRAINT UQ_TenSP UNIQUE (TenSP);
- ALTER TABLE KHACHHANG ADD DEFAULT 0 FOR DoanhSo;
- ALTER TABLE KHACHHANG ADD CONSTRAINT DF_DoanhSo DEFAULT 0 FOR DoanhSo;

2.2. Lệnh sửa cấu trúc bảng

Xóa ràng buộc toàn vẹn

***ALTER TABLE tên_bảng DROP CONSTRAINT
tên_ràng_buộc***

— Ví dụ:

- ***ALTER TABLE CTHD DROP CONSTRAINT
FK_CT_SP***
- ***ALTER TABLE SANPHAM DROP CONSTRAINT
CK_GIA***
- **Lưu ý:** đối với ràng buộc khóa chính, muốn xóa ràng buộc này phải xóa hết các ràng buộc khóa ngoại tham chiếu tới nó

2.2. Lệnh sửa cấu trúc bảng

Các nguyên tắc chung:

- Một constraint luôn gắn với một bảng.
- Nếu không đặt tên thì hệ thống sẽ tự động phát sinh tên cho constraint.
- Có thể tạo constraint:
 - Cùng với thời điểm tạo bảng.
 - Sau khi đã tạo bảng xong (dùng câu lệnh Alter).
- Có thể khai báo constraint ở mức cột hoặc mức bảng.
- Có thể xem các constraint hiện có trong database.

2.2. Lệnh sửa cấu trúc bảng

Cách viết Constraint: Định nghĩa ràng buộc ngay sau kiểu dữ liệu

```
CREATE TABLE GHISO  
(  
    MACN char(5) PRIMARY KEY,  
    HOTEN varchar(50) NOT NULL,  
    SDT varchar(20) UNIQUE,  
    SOTIEN money DEFAULT 0  
)
```

Lưu ý: Nên dùng với PRIMARY KEY, NOT NULL, UNIQUE, DEFAULT.

2.2. Lệnh sửa cấu trúc bảng

Cách viết Constraint: Định nghĩa ràng buộc ngay tại dòng cuối cùng của lệnh tạo bảng.

```
CREATE TABLE GHISO  
(  
    MACN char(5),  
    HOTEN varchar(50),  
    SDT varchar(20),  
    SOTIEN Money  
    CONSTRAINT PK_GS PRIMARY KEY(MACN)  
)
```

Lưu ý: Nên dùng với với Primary Key (Khóa chính có nhiều thuộc tính)

Có thể viết tắt: Primary key(MACN)

2.2. Lệnh sửa cấu trúc bảng

Cách viết Constraint: Thêm ràng buộc sau khi tạo bảng

- **ALTER TABLE NHANVIEN ADD CONSTRAINT PK_NV
PRIMARY KEY (MANV)**
- **ALTER TABLE CTHD ADD CONSTRAINT FK_CT_SP
FOREIGN KEY (MASP) REFERENCES SANPHAM(MASP)**

Lưu ý: Nên dùng với Foreign Key, Check

Khóa ngoại ở bảng nào thì sửa bảng đó

2.2. Lệnh sửa cấu trúc bảng

Ví dụ: Homework 2 – VIII – Quản lý cửa hàng hoa

1. Viết câu lệnh SQL: (1.5 điểm)

Đề 1:

- a. Tạo ràng buộc miền giá trị cho thuộc tính DAGIAO ghi nhận trạng thái đơn hàng. (0.75 điểm)
- b. Cập nhật tổng tiền giảm 2% nếu đơn hàng được bán từ ngày 1/11/2023 về sau. (0.75 điểm)

Đề 2:

- a. Tạo ràng buộc miền giá trị cho thuộc tính THELOAI của sản phẩm hoa. (0.75 điểm)
- b. Cập nhật phí ship tăng 5% cho ngày giao hàng là 20/11/2023. (0.75 điểm)

2.2. Lệnh sửa cấu trúc bảng

a. Tạo ràng buộc miền giá trị cho thuộc tính DAGIAO ghi nhận trạng thái đơn hàng. (0.75 điểm)

```
ALTER TABLE GIAOHANG ADD CONSTRAINT GH_DG_CHK  
CHECK (DAGIAO in (1,0))
```

```
ALTER TABLE GIAOHANG ADD CHECK (DAGIAO in (1,0))
```

a. Tạo ràng buộc miền giá trị cho thuộc tính THELOAI của sản phẩm hoa. (0.75 điểm)

```
ALTER TABLE SANPHAM ADD CONSTRAINT SP_TL_CHK CHECK  
(THELOAI in (N'chậu hoa cắm sẵn', N'hoa bó sẵn', N'hoa chậu', N'hoa cắt  
cành', N'hoa nhập khẩu', N'lá trang trí'))
```

```
ALTER TABLE SANPHAM ADD CHECK (THELOAI in (N'chậu hoa cắm  
sẵn', N'hoa bó sẵn', N'hoa chậu', N'hoa cắt cành', N'hoa nhập khẩu', N'lá  
trang trí'))
```

2.3. Lệnh xóa bảng (DROP)

- **Cú pháp**

DROP TABLE tên_bảng

- **Ví dụ:** xóa bảng KHACHHANG.

DROP TABLE KHACHHANG

- **Lưu ý:** khi muốn xóa một bảng phải xóa tất cả những khóa ngoại tham chiếu tới bảng đó trước.

2.4 Sửa tên bảng

Bằng cách nào?

2.4 Sửa tên bảng

- Sử dụng ALTER TABLE ... RENAME TO
 - Cú pháp: ALTER TABLE old_table_name RENAME TO new_table_name;
 - Ví dụ: ALTER TABLE Employees RENAME TO Staff;
 - **Lưu ý:** chỉ có **Oracle, PostgreSQL, SQLite** và **1 số DBMS khác** hỗ trợ
- Sử Dụng sp_rename (**SQL Server**)
 - Cú pháp: EXEC sp_rename 'old_table_name', 'new_table_name';
 - Ví dụ: EXEC sp_rename 'Employees', 'Staff';
 - sp_rename là một thủ tục hệ thống (stored procedure) trong SQL Server cho phép đổi tên các đối tượng trong cơ sở dữ liệu, bao gồm bảng, cột, chỉ mục và các đối tượng khác

NGÔN NGỮ THAO TÁC DỮ LIỆU

3. Ngôn ngữ thao tác dữ liệu

- Là ngôn ngữ cho phép thao tác trên dữ liệu của bảng
- **Gồm các lệnh:**
 1. Lệnh thêm dữ liệu (**INSERT**)
 2. Lệnh sửa dữ liệu (**UPDATE**)
 3. Lệnh xóa dữ liệu (**DELETE**)

3.1. Lệnh Insert

•Cú pháp

- Thêm 1 dòng dữ liệu vào bảng:

- INSERT INTO tên_bảng (cột1,...,cột n) VALUES (giá_trị_1,..., giá_trị_n)
- INSERT INTO tên_bảng VALUES (giá_trị_1, giá_trị_2,..., giá_trị_n)

- Thêm nhiều dòng dữ liệu vào bảng (thêm dữ liệu từ 1 bảng đã có)

- SELECT * INTO tên-bảng-mới FROM tên-bảng-có-sẵn
- INSERT INTO tên-bảng-tạo-trước SELECT * FROM tên-bảng-có-sẵn

3.1. Lệnh Insert

- **Ví dụ:**

- INSERT INTO SANPHAM VALUES('BC01','But chi', 'cay', 'Singapore', 3000)
- INSERT INTO SANPHAM (masp,tensp,dvt,nuocsx,gia) VALUES ('BC01','But chi','cay','Singapore',3000)

Chú ý: Chuỗi phải có dấu nháy đơn ‘ ’

3.1. Lệnh Insert

• Ví dụ của select...into

- `SELECT * INTO SANPHAM_NEW FROM SANPHAM`
- `SELECT * INTO SANPHAM_NEW FROM SANPHAM WHERE điều-kiện`
- `INSERT INTO SANPHAM_Sing SELECT * FROM SANPHAM WHERE NuocSX= 'Singapore'`
-- (Bảng SANPHAM_Sing phải được tạo trước)

• Ví dụ của insert into....select....

- `INSERT INTO SANPHAM_COPY SELECT * FROM SANPHAM`
- `INSERT INTO SANPHAM_COPY SELECT * FROM SANPHAM WHERE điều-kiện`

3.1. Lệnh Insert

• Một số lưu ý:

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

3.2. Lệnh Update

- **Cú pháp**

UPDATE tên_bảng

SET cột_1 = giá_trị_1, cột_2 = giá_trị_2

[WHERE điều_kiện]

- **Ví dụ:** Tăng giá 10% đối với những sản phẩm do “VietNam” sản xuất

UPDATE SANPHAM

SET Gia = Gia*1.1

WHERE Nuocsx='VietNam'

3.2. Lệnh Update

❖ Lưu ý khi dùng câu lệnh Update:

- 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 giá trị của thuộc tính đó đều sẽ bị cập nhật
- Lệnh **UPDATE** có thể gây vi phạm RBTV khóa ngoại: không cho sửa

3.2. Lệnh Update

Ví dụ: Homework 2 – VIII – Quản lý cửa hàng hoa

1. Viết câu lệnh SQL: (1.5 điểm)

Đề 1:

- a. Tạo ràng buộc miền giá trị cho thuộc tính DAGIAO ghi nhận trạng thái đơn hàng. (0.75 điểm)
- b. Cập nhật tổng tiền giảm 2% nếu đơn hàng được bán từ ngày 1/11/2023 về sau. (0.75 điểm)

Đề 2:

- a. Tạo ràng buộc miền giá trị cho thuộc tính THELOAI của sản phẩm hoa. (0.75 điểm)
- b. Cập nhật phí ship tăng 5% cho ngày giao hàng là 20/11/2023. (0.75 điểm)

3.2. Lệnh Update

b. Cập nhật tổng tiền giảm 2% nếu đơn hàng được bán từ ngày 1/11/2023 về sau. (0.75 điểm)

```
UPDATE DONHANG SET TONGTIEN=TONGTIEN*0.98 WHERE  
NGAYBAN >= '1/11/2023'
```

b. Cập nhật phí ship tăng 5% cho ngày giao hàng là 20/11/2023. (0.75 điểm)

```
UPDATE GIAOHANG SET PHISHIP=PHISHIP*1.05 WHERE  
NGAYGIAO = '20/11/2023'
```

3.3. Lệnh Delete

- **Cú pháp**

DELETE FROM tên_bảng [WHERE điều_kiện]

- **Ví dụ:**

- **Xóa toàn bộ nhân viên**

DELETE FROM NHANVIEN

- **Xóa những sản phẩm do Trung Quốc sản xuất có giá thấp hơn 10000**

**DELETE FROM SANPHAM
WHERE (Gia <10000) AND (Nuocsx='Trung Quoc')**

3.3. Lệnh Delete

❖ Lưu ý khi sử dụng lệnh DELETE:

- Những dòng thỏa điều kiện tại mệnh đề WHERE sẽ bị xóa đi
- Nếu không chỉ định điều kiện ở mệnh đề WHERE, tất cả các dòng sẽ bị xóa
- Lệnh DELETE có thể gây vi phạm RBTV khóa ngoại: không cho xóa

3.4. Cascade

- CASCADE là một **tùy chọn** trong SQL được sử dụng trong các câu lệnh FOREIGN KEY hoặc ON DELETE và ON UPDATE để đảm bảo tính toàn vẹn tham chiếu (referential integrity) giữa các bảng có quan hệ với nhau khi thực hiện các thao tác xóa hoặc cập nhật dữ liệu. CASCADE cho phép tự động thực hiện các thay đổi trên các bảng liên quan khi dữ liệu ở bảng chính thay đổi.
- Tác dụng:
 - Duy trì tính toàn vẹn dữ liệu
 - Tự động hóa quản lý dữ liệu
 - Tránh "orphan records" (bản ghi mồ côi)
 - Đảm bảo ràng buộc tham chiếu

3.4. Cascade

ON DELETE CASCADE: Khi một dòng trong bảng chính bị xóa, tất cả các dòng trong bảng phụ liên quan đến nó cũng sẽ bị xóa tự động. Ví dụ: Nếu một phòng ban trong bảng Department bị xóa, tất cả nhân viên thuộc phòng ban đó trong bảng Employee cũng sẽ bị xóa.

```
CREATE TABLE Department (  
    DeptID INT PRIMARY KEY,  
    DeptName VARCHAR(100)  
);
```

```
CREATE TABLE Employee (  
    EmpID INT PRIMARY KEY,  
    EmpName VARCHAR(100),  
    DeptID INT,  
    FOREIGN KEY (DeptID) REFERENCES Department(DeptID) ON DELETE  
CASCADE  
);
```

3.4. Cascade

ON UPDATE CASCADE: Khi một giá trị trong bảng chính được cập nhật (chẳng hạn khóa chính), tất cả các giá trị tương ứng trong bảng phụ cũng được cập nhật theo.

```
CREATE TABLE Employee (  
    EmpID INT PRIMARY KEY,  
    EmpName VARCHAR(100),  
    DeptID INT,  
    FOREIGN KEY (DeptID) REFERENCES Department(DeptID) ON  
UPDATE CASCADE  
);
```

Khi DeptID trong bảng Department được cập nhật, các giá trị DeptID tương ứng trong bảng Employee cũng sẽ tự động được cập nhật theo.

3.4. Cascade

DROP TABLE ... CASCADE: Khi xóa một bảng với CASCADE, các bảng hoặc đối tượng liên quan đến bảng đó (ví dụ: bảng phụ thuộc, khóa ngoại, chỉ mục, khung nhìn) cũng sẽ được xóa theo, tránh lỗi xóa bị chặn do các ràng buộc.

Cú pháp: `DROP TABLE table_name CASCADE;`

Bài tập tại lớp

- ❖ Cho lược đồ CSDL sau:
- ❖ **SINHVIEN**(MaSV,Hoten,GT,Ngaysinh,CMND,MaKhoa)
- ❖ **KHOA** (MaKhoa,TenKhoa,SoluongGV,SoluongSV)
- ❖ **HOADONHOCPhi**(MAHDHP,MASV,SOTIEN)
- Yêu cầu:
 - 1. Sử dụng ngôn ngữ SQL để tạo bảng cho các quan hệ trên
 - 2. Tạo ràng buộc khóa chính, khóa ngoại cho các bảng trên
 - 3. Trong bảng **SINHVIEN** tạo ràng buộc kiểm tra CMND là duy nhất
 - 4. Giảm 10% học phí cho sinh viên do tình hình dịch COVID-19
 - 5. Sửa ngày sinh của SV có mã '2412001' sang 24/12/2003

NGÔN NGỮ TRUY VẤN DỮ LIỆU

4. Ngôn ngữ truy vấn dữ liệu

- Là ngôn ngữ chuẩn, có cấu trúc dùng để truy vấn và thao tác trên CSDL quan hệ.

- Câu truy vấn tổng quát:

SELECT [DISTINCT] danh_sách_cột | hàm

FROM danh sách các quan hệ (hay bảng, table)

[WHERE điều_kiện]

[GROUP BY danh_sách_cột_gom_nhóm]

[HAVING điều_kiện_trên_nhóm]

[ORDER BY cột1 ASC | DESC, cột2 ASC | DESC,...]

a) ~~where KH.TENKH, KH.TENKH, KH.HOSPHI~~
where KH.HOSPHI > 5,000,000 AND KH.THOTUONG < 3
? from KHOAHOC KH
' select KH.MARKH, KH.TENKH, KH.HOSPHI

4. Ngôn ngữ truy vấn dữ liệu

Thứ tự viết câu truy vấn là **cố định**, **KHÔNG** đổi được

1. **SELECT**: Chọn các cột hoặc biểu thức để hiển thị trong kết quả.
2. **FROM**: Xác định các bảng (quan hệ) được sử dụng trong truy vấn.
3. **WHERE**: Lọc các bản ghi (bộ) thỏa mãn điều kiện.
4. **GROUP BY**: Gom nhóm các bản ghi theo các cột được chỉ định.
5. **HAVING**: Lọc các nhóm thỏa mãn điều kiện.
6. **ORDER BY**: Sắp xếp kết quả theo các cột được chỉ định.
7. **LIMIT** hoặc **FETCH**: Giới hạn số lượng hàng được trả về.
(Chú ý: **LIMIT KHÔNG** có trong **SQL Server**)

4. Ngôn ngữ truy vấn dữ liệu

Thứ tự thực hiện câu truy vấn

1. FROM: Lấy dữ liệu từ bảng và xử lý JOIN, APPLY, UNION nếu có.
2. WHERE: Lọc các bản ghi (bộ) thỏa mãn điều kiện.
3. GROUP BY: Gom nhóm các bản ghi theo các cột được chỉ định.
4. HAVING: Lọc các nhóm thỏa mãn điều kiện sau khi GROUP BY
5. SELECT: Chọn các cột hoặc biểu thức để hiển thị trong kết quả.
6. ORDER BY: Sắp xếp kết quả theo các cột được chỉ định.
7. ~~LIMIT~~ hoặc FETCH: Giới hạn số lượng hàng được trả về.
(Chú ý: **LIMIT KHÔNG** có trong SQL Server, có ở MySQL)

4. Ngôn ngữ truy vấn dữ liệu

Liên hệ giữa đại số quan hệ (ĐSQH) và SQL:

Một số phép toán tương đương:

- Phép chọn σ tương đương với WHERE trong SQL (có 1 chút khác biệt)
- Phép chiếu π tương đương với việc chỉ định các cột trong mệnh đề SELECT (có 1 chút khác biệt)
- Phép hợp \cup tương đương với UNION trong SQL.
- Phép giao \cap tương đương với INTERSECT trong SQL.
- Phép hiệu – tương đương với EXCEPT trong SQL (hoặc MINUS trong Oracle).
- Phép kết (Join) trong đại số quan hệ có thể được biểu diễn bằng nhiều loại kết hợp trong SQL như INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL JOIN.

4. Ngôn ngữ truy vấn dữ liệu

Liên hệ giữa đại số quan hệ (ĐSQH) và SQL:

SQL sẽ có 1 số điểm khác với ĐSQH:

- **Phép chọn và WHERE:** SQL có thể sử dụng các toán tử phức tạp, phép tính toán, hàm và biểu thức hơn so với đại số quan hệ.
- **Phép chia:** SQL Không có phép chia trực tiếp. Phải sử dụng NOT EXISTS hoặc các truy vấn con để mô phỏng phép chia.
- **Từ khóa DISTINCT:** được sử dụng trong SQL để loại các dòng trùng lặp sau khi thực hiện phép chiếu (SELECT). Khác với ĐSQH phép chiếu tự động loại bỏ các bộ trùng lặp, trong SQL phải dùng từ khóa DISTINCT.
- ...

4. Ngôn ngữ truy vấn dữ liệu

Một số từ khóa đi ngay sau SELECT:

- **DISTINCT**: nhằm loại bỏ các hàng trùng lặp trong kết quả. Nếu một truy vấn không có từ khóa DISTINCT, SQL sẽ trả về tất cả các hàng, bao gồm cả các hàng trùng lặp. Khi thêm từ khóa DISTINCT, SQL sẽ loại bỏ các bản ghi trùng lặp và chỉ trả về các hàng duy nhất.
- **ALL**: nhằm trả về tất cả các hàng, bao gồm cả các hàng trùng lặp, là hành vi mặc định của SQL khi không có DISTINCT.
- **TOP**: Giới hạn số lượng hàng trả về (SQL Server)

Cú pháp: `SELECT TOP (n) [PERCENT] column1, column2
FROM table_name;`

Ví dụ: `SELECT TOP (5) * FROM Employees;`

4. Ngôn ngữ truy vấn dữ liệu

- **TOP WITH TIES**: để bao gồm cả các bản ghi có giá trị bằng với bản ghi cuối cùng trong tập hợp được chọn.

Ví dụ:

```
SELECT TOP 3 WITH TIES * FROM NhanVien  
ORDER BY Luong DESC;
```

(Lấy ra 3 nhân viên có lương cao nhất, cộng thêm các nhân viên có lương bằng với nhân viên thứ 3)

- **SELECT * FROM TenBang**: Truy xuất **tất cả các cột** của tất cả các dòng từ một bảng hoặc kết quả của một truy vấn.

Ví dụ: `SELECT * FROM NhanVien`

Lấy tất cả thông tin (thuộc tính) của các nhân viên

4.1. Truy vấn cơ bản

- **SELECT 1**: thường được sử dụng để kiểm tra điều kiện hoặc thực hiện các truy vấn lồng nhau. Trả về một hàng với giá trị 1. Hoặc dùng trong truy vấn con với EXISTS hoặc NOT EXISTS để **kiểm tra xem một điều kiện có tồn tại hay không** (sẽ đề cập ở phần bài học sau)

```
SELECT 1
FROM NHANVIEN
WHERE EXISTS (SELECT * FROM DUAN
              WHERE DUAN.MaNV = NHANVIEN.MaNV );
```

4.1. Truy vấn cơ bản

SELECT TOP và LIMIT đều có chức năng giới hạn số lượng bản ghi trả về, nhưng có cú pháp và DBMS hỗ trợ khác nhau.

SELECT TOP: Chủ yếu được hỗ trợ bởi Microsoft SQL Server và MS Access.

LIMIT: Chủ yếu được hỗ trợ bởi MySQL, PostgreSQL, và SQLite

4. Ngôn ngữ truy vấn dữ liệu

- ❖ Toán tử so sánh:
 - =, >, <, >=, <=, <>
 - BETWEEN
 - IS NULL, IS NOT NULL
 - LIKE (%,_): % đại diện nhiều ký tự, _ đại diện 1 ký tự
 - IN, NOT IN
 - EXISTS, NOT EXISTS
 - SOME, ALL, ANY
- ❖ Toán tử logic: AND, OR.
- ❖ Các phép toán: +, -, * (×), / (÷)
- ❖ Các hàm xử lý ngày (DAY()), tháng (MONTH()), năm (YEAR())

4. Ngôn ngữ truy vấn dữ liệu

Hàm hệ thống trong SQL là những hàm được tích hợp sẵn trong hệ quản trị cơ sở dữ liệu (DBMS) để thực hiện các tác vụ cụ thể, hỗ trợ việc quản lý và thao tác dữ liệu.

- Xử lý chuỗi: Thao tác trên chuỗi ký tự như cắt chuỗi, nối chuỗi, thay thế ký tự, chuyển đổi chữ hoa chữ thường,...
- Xử lý số: Thực hiện các phép toán số học, làm tròn số, tính giá trị tuyệt đối,...
- Xử lý ngày tháng: Thao tác trên dữ liệu ngày tháng như lấy ngày, tháng, năm, tính khoảng cách giữa hai ngày,...
- Chuyển đổi kiểu dữ liệu: Chuyển đổi dữ liệu từ kiểu này sang kiểu khác.
- Xử lý thông tin hệ thống: Lấy thông tin về người dùng, phiên làm việc, cơ sở dữ liệu,...
- Khác: Thực hiện các tác vụ như mã hóa dữ liệu, tạo số ngẫu nhiên,...

4. Ngôn ngữ truy vấn dữ liệu

Các hàm xử lý chuỗi:

SUBSTRING (): trích xuất một chuỗi con từ chuỗi ban đầu, bắt đầu từ vị trí được chỉ định và có độ dài bằng số ký tự được chỉ định

LEFT (): Trích xuất một chuỗi con từ bên trái của chuỗi ban đầu, với độ dài bằng số ký tự được chỉ định.

RIGHT (): Trích xuất một chuỗi con từ bên phải của chuỗi ban đầu, với độ dài bằng số ký tự được chỉ định

UPPER (): Chuyển đổi tất cả các ký tự trong chuỗi thành chữ hoa

LOWER (): Chuyển đổi tất cả các ký tự trong chuỗi thành chữ thường

REPLACE (): Thay thế tất cả các lần xuất hiện của chuỗi cũ trong chuỗi ban đầu bằng chuỗi mới.

CONCAT (): Nối các chuỗi lại với nhau.

...

4. Ngôn ngữ truy vấn dữ liệu

Các hàm xử lý chuỗi:

LENGTH(): Trả về độ dài của chuỗi (số lượng ký tự).

TRIM(): Loại bỏ khoảng trắng ở đầu và cuối chuỗi.

LTRIM(): Loại bỏ khoảng trắng ở đầu chuỗi.

RTRIM(): Loại bỏ khoảng trắng ở cuối chuỗi.

POSITION(): Tìm vị trí đầu tiên của chuỗi con trong chuỗi.

SUBSTR(): Tương tự như SUBSTRING, trích xuất chuỗi con từ chuỗi ban đầu.

LPAD(): Thêm ký tự đệm vào bên trái chuỗi cho đến khi đạt độ dài mong muốn.

RPAD(): Thêm ký tự đệm vào bên phải chuỗi cho đến khi đạt độ dài mong muốn.

4. Ngôn ngữ truy vấn dữ liệu

Các hàm xử lý số:

ABS(số): Trả về giá trị tuyệt đối của một số.

SQRT(số): Trả về căn bậc hai của một số.

ROUND(số, số_chữ_số_thập_phân): Làm tròn một số đến số chữ số thập phân được chỉ định.

SIGN(số): Trả về dấu của một số (-1 nếu âm, 0 nếu bằng 0, 1 nếu dương).

POWER(số, số_mũ): Trả về số lũy thừa của một số.

CEIL(số): Làm tròn một số lên đến số nguyên gần nhất.

FLOOR(số): Làm tròn một số xuống đến số nguyên gần nhất.

MOD(số1, số2): Trả về phần dư của phép chia số1 cho số2.

4. Ngôn ngữ truy vấn dữ liệu

Các hàm xử lý ngày:

DAY(ngày): Trả về ngày trong tháng của một ngày.

MONTH(ngày): Trả về tháng trong năm của một ngày.

YEAR(ngày): Trả về năm của một ngày.

DATE(biểu_thức): Trích xuất phần ngày từ một biểu thức ngày tháng.

CURRENT_DATE(): Trả về ngày hiện tại.

CURRENT_TIME(): Trả về thời gian hiện tại.

CURRENT_TIMESTAMP(): Trả về dấu thời gian hiện tại (bao gồm cả ngày và giờ).

DATE_ADD(ngày, INTERVAL giá_trị đơn_vị): Cộng thêm một khoảng thời gian vào ngày.

DATE_SUB(ngày, INTERVAL giá_trị đơn_vị): Trừ đi một khoảng thời gian từ ngày.

DATEDIFF(ngày1, ngày2): Tính số ngày chênh lệch giữa hai ngày.

4. Ngôn ngữ truy vấn dữ liệu

Các hàm tính toán trên nhóm:

MIN(cột): Trả về giá trị nhỏ nhất của cột được chỉ định trong nhóm dữ liệu.

MAX(cột): Trả về giá trị lớn nhất của cột được chỉ định trong nhóm dữ liệu.

AVG(cột): Trả về giá trị trung bình của cột được chỉ định trong nhóm dữ liệu.

SUM(cột): Trả về tổng giá trị của cột được chỉ định trong nhóm dữ liệu.

COUNT(cột): Trả về số lượng giá trị khác NULL của cột được chỉ định trong nhóm dữ liệu.

COUNT(*): Trả về số lượng dòng trong nhóm dữ liệu, kể cả dòng có giá trị NULL. **COUNT(1):** tương tự COUNT(*)

4. Ngôn ngữ truy vấn dữ liệu

Chuyển đổi kiểu dữ liệu: Dùng CAST

SELECT CAST('123' AS INT); -- Chuyển chuỗi '123' thành số nguyên

SELECT CAST('123.45' AS FLOAT); -- Chuyển chuỗi thành số dấu chấm động

SELECT CAST(100 AS VARCHAR(10)); -- Chuyển số nguyên 100 thành chuỗi

Chuyển đổi kiểu dữ liệu: Dùng CONVERT

SELECT CONVERT(VARCHAR(10), 12345); -- Chuyển số nguyên 12345 thành chuỗi

SELECT CONVERT(VARCHAR(10), GETDATE(), 103); -- Chuyển ngày hiện tại thành chuỗi với định dạng 'DD/MM/YYYY'

4. Ngôn ngữ truy vấn dữ liệu

- **ALL:** trả về giá trị là TRUE nếu **tất cả** các giá trị trong câu truy vấn con đáp ứng điều kiện, được sử dụng với SELECT, WHERE, HAVING.
- **ANY:** trả về TRUE nếu **BẤT KỲ** (ít nhất một) giá trị nào trong số các giá trị truy vấn con đáp ứng điều kiện
ANY có nghĩa là điều kiện sẽ đúng nếu hoạt động đúng với bất kỳ giá trị nào trong phạm vi.

Ví dụ: SELECT * FROM SanPham WHERE Gia > ANY
(SELECT Gia FROM SanPham WHERE NuocSX = 'Trung Quốc')

Truy vấn này sẽ chọn ra tất cả sản phẩm có giá lớn hơn bất kỳ sản phẩm nào do Trung Quốc sản xuất.

- **SOME:** tương đương **ANY**

4. Ngôn ngữ truy vấn dữ liệu

- Phân loại câu SELECT:
 - SELECT đơn giản.
 - SELECT có mệnh đề ORDER BY.
 - SELECT lồng (câu SELECT lồng câu SELECT khác).
 - SELECT gom nhóm (GROUP BY).
 - SELECT gom nhóm (GROUP BY) có điều kiện HAVING.

4. Ngôn ngữ truy vấn dữ liệu

Bài tập: Cho lược đồ CSDL “quản lý đề án công ty” như sau

NHANVIEN (MaNV, HoTen, Phai, Luong, NTNS, Ma_NQL, MaPH)

PHONGBAN (MaPH, TenPH, TRPH)

DEAN (MaDA, TenDA, Phong, NamThucHien)

PHANCONG (MaNV, MaDA, ThoiGian)

| MANV | HOTEN | NTNS | PHAI | MA_NQL | MaPH | LUONG |
|------|-------------------|------------|------|--------|------|-----------|
| 001 | Vu Le Quyen | 02/04/1981 | Nu | | QL | 3.000.000 |
| 002 | Nguyen Thanh Tung | 05/07/1994 | Nam | 001 | NC | 2.500.000 |
| 003 | Van Mai Huong | 27/09/1994 | Nu | 001 | DH | 2.500.000 |
| 004 | Dang Thanh An | 09/01/2001 | Nam | 002 | NC | 2.200.000 |
| 005 | Tran Minh Hieu | 28/09/1999 | Nam | 003 | DH | 2.200.000 |
| 006 | Pham Bao Khang | 05/04/1999 | Nam | 002 | NC | 2.000.000 |
| 007 | Nguyen Dieu Huyen | 28/03/2003 | Nu | 002 | NC | 2.200.000 |
| 008 | Nguyen Thao Linh | 07/10/2000 | Nu | 004 | NC | 1.800.000 |

NHANVIEN

DEAN

| MADA | TENDA | PHONG | NamThucHien |
|-------|---------------|-------|-------------|
| TH001 | Tin hoc hoa 1 | NC | 2022 |
| TH002 | Tin hoc hoa 2 | NC | 2023 |
| DT001 | Dao tao 1 | DH | 2024 |
| DT002 | Dao tao 2 | DH | 2024 |

PHONGBAN

| MAPH | TENPH | TRPH |
|------|------------|------|
| QL | Quan Ly | 001 |
| DH | Dieu Hanh | 003 |
| NC | Nghien Cuu | 002 |

PHANCONG

| MANV | MADA | THOIGIAN |
|------|-------|----------|
| 001 | TH001 | 30,0 |
| 001 | TH002 | 12,5 |
| 002 | TH001 | 10,0 |
| 002 | TH002 | 10,0 |
| 002 | DT001 | 10,0 |
| 002 | DT002 | 10,0 |
| 003 | TH001 | 37,5 |
| 004 | DT001 | 22,5 |
| 004 | DT002 | 10,0 |
| 006 | DT001 | 30,5 |
| 007 | TH001 | 20,0 |
| 007 | TH002 | 10,0 |
| 008 | DT002 | 12,5 |

4.1. Truy vấn cơ bản

SELECT [DISTINCT] danh_sách_cột | hàm
FROM danh sách các quan hệ (hay bảng, table)
[WHERE điều_kiện]
[ORDER BY cột1 ASC | DESC, cột2 ASC | DESC,...]

Câu hỏi : Liệt kê mã, họ tên của tất cả các nhân viên

➡ SELECT MaNV, HoTen FROM NhanVien

| MaNV | HoTen |
|------|-------------------|
| 001 | Vu Le Quyen |
| 002 | Nguyen Thanh Tung |
| 003 | Van Mai Huong |
| 004 | Dang Thanh An |
| 005 | Tran Minh Hieu |
| 006 | Pham Bao Khang |
| 007 | Nguyen Dieu Huyen |
| 008 | Nguyen Thao Linh |

4.1. Truy vấn cơ bản

Câu hỏi : Liệt kê mã, họ tên các nhân viên thuộc phòng có mã 'NC'

➡ `SELECT MaNV, HoTen FROM NhanVien
WHERE MaPH = 'NC'`

| MaNV | HoTen |
|------|-------------------|
| 002 | Nguyen Thanh Tung |
| 004 | Dang Thanh An |
| 006 | Pham Bao Khang |
| 007 | Nguyen Dieu Huyen |
| 008 | Nguyen Thao Linh |

Câu hỏi : Liệt kê thông tin của tất cả các nhân viên

➡ `SELECT * FROM NhanVien`

Câu hỏi : Liệt kê mã, họ tên các nhân viên thuộc phòng có mã 'DH' và lương tạm tính sau khi tăng 20%

➡ `SELECT MaNV, HoTen, Luong*1.2 'Luong tam
tinh'
FROM NhanVien
WHERE MaPH = 'DH'`

| MaNV | HoTen | Luong tam tinh |
|------|----------------|----------------|
| 003 | Van Mai Huong | 3000000.000 |
| 005 | Tran Minh Hieu | 2640000.000 |

4.1. Truy vấn cơ bản – bổ sung

```
SELECT MaNV, HoTen, Luong*1.2 AS 'Luong tam tinh'  
FROM NhanVien  
WHERE MaPH = 'DH'
```

- **Alias (bí danh)** trong SQL là một **tên tạm thời** được gán cho một bảng hoặc cột trong truy vấn SQL để làm cho câu lệnh dễ đọc, dễ hiểu hơn hoặc để rút gọn các tên dài. Alias không làm thay đổi tên thực tế của bảng hoặc cột trong cơ sở dữ liệu, nó chỉ có hiệu lực trong kết quả của câu truy vấn hoặc trong phạm vi truy vấn SQL đang thực thi.
- Alias giúp truy vấn trở nên ngắn gọn và dễ bảo trì, đặc biệt khi làm việc với nhiều bảng, nhiều cột, hoặc khi cần tạo ra tên rõ ràng hơn cho các trường tính toán.
- Có thể sử dụng từ khóa AS hoặc không, nhưng cách viết có AS được xem là chuẩn SQL và dễ hiểu hơn, giúp mã nguồn rõ ràng và dễ bảo trì, tạo ra sự tường minh, đặc biệt trong các truy vấn lớn hoặc trong các môi trường phát triển phần mềm.

4.1. Truy vấn cơ bản – bổ sung

Alias cho Bảng (Table Alias): Dùng để gán tên ngắn gọn cho một bảng, giúp rút ngắn cú pháp khi truy vấn và khi làm việc với nhiều bảng

```
SELECT column_name FROM table_name AS alias_name;
```

Ví dụ:

```
SELECT NV.MaNV, NV.HoTen, PB.TenPH  
FROM NhanVien AS NV JOIN PhongBan AS PB  
ON NV.MaPH = PB.MaPH;
```

Alias cho Cột (Column Alias): Dùng để thay đổi tên hiển thị của cột trong kết quả của truy vấn

```
SELECT column_name AS alias_name  
FROM table_name;
```

4.1. Truy vấn cơ bản – bổ sung

Phép gán vào các bảng tạm, lưu trữ kết quả trung gian của một truy vấn.

- **WITH ... AS** (hay còn gọi là Common Table Expressions - CTEs) là một phương pháp để định nghĩa một bảng tạm thời trong một truy vấn SQL.

Ví dụ: WITH EmployeeCTE AS (

SELECT EmployeeID, DepartmentID, Salary

FROM Employees

WHERE Salary > 50000

)

SELECT DepartmentID, COUNT(EmployeeID) AS NumHighSalaryEmployees

FROM EmployeeCTE

GROUP BY DepartmentID;

4.1. Truy vấn cơ bản – bổ sung

Phép gán vào các bảng tạm, lưu trữ kết quả trung gian của một truy vấn.

- **SELECT ... AS** cũng tương tự câu trên. Sử dụng cấu trúc này khi chỉ cần dùng một bảng tạm trong một phần của truy vấn chính.

Ví dụ:

```
SELECT DepartmentID, COUNT(EmployeeID) AS  
NumHighSalaryEmployees  
FROM
```

```
    (SELECT EmployeeID, DepartmentID, Salary  
     FROM Employees  
     WHERE Salary > 50000  
     ) AS EmployeeTemp
```

```
GROUP BY DepartmentID;
```

4.1. Truy vấn cơ bản – bổ sung

Phép gán vào các bảng tạm, lưu trữ kết quả trung gian của một truy vấn.

- **TEMPORARY TABLE** là một bảng tạm thời được tạo ra để lưu trữ dữ liệu và có thể được sử dụng trong suốt phiên làm việc hiện tại.

Ví dụ: CREATE TEMPORARY TABLE tempEmployee AS
SELECT EmployeeID, Salary
FROM Employees
WHERE Salary > 50000;

SELECT *
FROM tempEmployee;

4.1. Truy vấn cơ bản – bổ sung

Phép gán vào các bảng tạm, lưu trữ kết quả trung gian của một truy vấn.

TABLE VARIABLE là một loại biến tạm thời được lưu trữ trong bộ nhớ và có phạm vi tồn tại trong một khối mã hoặc một hàm cụ thể.

Ví dụ: DECLARE @HighSalaryEmployees TABLE (
EmployeeID INT,
Salary MONEY

);
INSERT INTO @HighSalaryEmployees
SELECT EmployeeID, Salary
FROM Employees
WHERE Salary > 50000;

SELECT *
FROM @HighSalaryEmployees;

4.1. Truy vấn cơ bản

Câu hỏi: Liệt kê mã, họ tên và năm sinh của các nhân viên nam

➔
SELECT MaNV, HoTen, YEAR(NTNS) 'Nam Sinh'
FROM NhanVien
WHERE Phai = 'Nam'

| MaNV | HoTen | Nam Sinh |
|------|-------------------|----------|
| 002 | Nguyen Thanh Tung | 1994 |
| 004 | Dang Thanh An | 2001 |
| 005 | Tran Minh Hieu | 1999 |
| 006 | Pham Bao Khang | 1999 |

Sử dụng **DISTINCT**: loại bỏ các dòng trùng nhau trong kết quả

Câu hỏi : Liệt kê các mức lương của nhân viên. Sắp xếp theo thứ tự giảm dần

➔
SELECT DISTINCT Luong 'Muc Luong'
FROM NhanVien
ORDER BY Luong DESC

| Muc Luong |
|------------|
| 3000000.00 |
| 2500000.00 |
| 2200000.00 |
| 2000000.00 |
| 1800000.00 |

Khi SELECT DISTINCT, các cột trong ORDER BY phải nằm trong danh sách SELECT

4.2. BETWEEN, ORDER BY, IS NULL

Câu hỏi 13: Sử dụng =, >, >=, ... Danh sách các nhân viên sinh trong khoảng từ năm 1990 đến 2000?

➔ SELECT MaNV, HoTen FROM NhanVien
WHERE Year(NTNS) >= 1990 AND Year(NTNS) <= 2000

Câu hỏi 14: Sử dụng BETWEEN, ORDER BY. Danh sách các nhân viên sinh trong khoảng từ năm 1990 đến 2000? Sắp xếp theo mức lương giảm dần.

➔ SELECT * FROM NhanVien WHERE Year(NTNS) BETWEEN 1990
AND 2000 ORDER BY Luong DESC

Câu hỏi 15: Sử dụng IS NULL. Cho biết những nhân viên không có người quản lý trực tiếp? (không chịu sự quản lý trực tiếp của người nào)

➔ SELECT MaNV, HoTen, NTNS, Ma_NQL FROM NhanVien WHERE
Ma_NQL IS NULL

| MaNV | HoTen | NTNS | Ma_NQL |
|------|-------------|------------|--------|
| 001 | Vu Le Quyên | 1981-04-02 | NULL |

SELECT MaNV, HoTen, FORMAT(NTNS, 'dd/MM/yyyy'), Ma_NQL
FROM NhanVien WHERE Ma_NQL IS NULL

| MaNV | HoTen | NTNS | Ma_NQL |
|------|-------------|------------|--------|
| 001 | Vu Le Quyên | 02/04/1981 | NULL |

Bổ sung

Định dạng số và ngày tháng:

1. Định dạng ngày tháng (Date):

- Sử dụng hàm FORMAT:

```
SELECT FORMAT(NTNS,'dd/MM/yyyy') AS 'NgàySinh'  
FROM NhanVien;
```

- 'dd/MM/yyyy': Ngày/Tháng/Năm (Ví dụ: 31/12/2024)
- 'MM-dd-yyyy': Tháng-Ngày-Năm (Ví dụ: 12-31-2024)
- 'yyyy-MM-dd': Năm-Tháng-Ngày (Ví dụ: 2024-12-31)

- Sử dụng hàm CONVERT

```
SELECT CONVERT(VARCHAR,NTNS,103) AS 'NgàySinh'  
FROM NhanVien;
```

- 101: MM/dd/yyyy
- 102: yyyy.MM.dd
- 103: dd/MM/yyyy

Bổ sung

Định dạng số và ngày tháng:

2. Định dạng số (Number)

- Sử dụng hàm FORMAT để định dạng số

```
SELECT FORMAT(Luong, '#,###0.00') AS 'Luong Dinh Dang'  
FROM NhanVien;
```

- #,###.00: Số có dấu phân cách hàng nghìn và 2 chữ số thập phân.
- C: Định dạng đơn vị tiền tệ theo cài đặt hệ thống (Ví dụ: \$1,234.56).
- N2: Định dạng số với dấu phân cách hàng nghìn và 2 chữ số thập phân (Ví dụ: 1,234.56).

- Sử dụng hàm ROUND để làm tròn số

```
SELECT ROUND(Luong, 2) AS 'Luong Lam Tron'  
FROM NhanVien;
```

Bổ sung

```
SELECT MaNV, HoTen, Luong*1.2 'Luong tam tinh'
FROM NhanVien
WHERE MaPH = 'DH'
```

| MaNV | HoTen | Luong tam tinh |
|------|----------------|----------------|
| 003 | Van Mai Huong | 3000000.000 |
| 005 | Tran Minh Hieu | 2640000.000 |

```
SELECT MaNV, HoTen, FORMAT(Luong*1.2,'#',##0.0') AS 'Luong tam tinh'
FROM NhanVien
WHERE MaPH = 'DH'
```

| MaNV | HoTen | Luong tam tinh |
|------|----------------|----------------|
| 003 | Van Mai Huong | 3,000,000.0 |
| 005 | Tran Minh Hieu | 2,640,000.0 |

4.3. SQL - SO SÁNH IN & NOT IN

Câu hỏi 16: **Sử dụng Is Not Null.** Cho biết những nhân viên có người quản lý trực tiếp? Thông tin hiển thị gồm: mã nhân viên, họ tên, mã người quản lý.

➡ `SELECT MaNV, HoTen, Ma_NQL FROM NhanVien
WHERE Ma_NQL IS NOT NULL`

Câu hỏi 17: **Sử dụng IN (so sánh với một tập hợp giá trị cụ thể).** Cho biết họ tên nhân viên thuộc phòng 'NC' hoặc phòng 'DH'?

➡ `SELECT DISTINCT Hoten FROM NhanVien WHERE MaPH IN
('NC', 'DH')`

Câu hỏi 18: **Sử dụng IN (so sánh với một tập hợp giá trị chọn từ câu SELECT khác).** Cho biết họ tên nhân viên thuộc phòng 'NC' hoặc phòng 'DH'?

➡ `SELECT Hoten FROM NhanVien WHERE MaPH IN (SELECT MaPH
FROM PHONGBAN WHERE TenPH = 'Nghien Cuu' OR TenPH =
'Dieu Hanh')`

4.3. SQL - SO SÁNH IN & NOT IN

Câu hỏi 19 (tt): Cho biết mã số, họ tên, ngày tháng năm sinh của những nhân viên đã tham gia đề án?

➡ `SELECT MaNV, HoTen, NTNS FROM NhanVien
WHERE MaNV in (SELECT MaNv FROM PhanCong)`

Câu hỏi 20: **Sử dụng NOT IN**. Cho biết mã số, họ tên, ngày tháng năm sinh của những nhân viên không tham gia đề án nào?

Gợi ý cho mệnh đề NOT IN: thực hiện câu truy vấn “tìm nhân viên có tham gia đề án (dựa vào bảng PhanCong)”, sau đó lấy phần bù.

➡ `SELECT MaNV, HoTen, NTNS FROM NhanVien
WHERE MaNV not in (SELECT MaNv FROM PhanCong)`

Câu hỏi 21 (tt): Cho biết tên phòng ban không chủ trì các đề án triển khai năm 2023? Gợi ý: thực hiện câu truy vấn “tìm phòng ban chủ trì các đề án triển khai năm 2023”, sau đó lấy phần bù.

➡ `SELECT TenPH FROM PhongBan WHERE MaPH not in (SELECT
DISTINCT Phong FROM DEAN WHERE NamThucHien=2023)`

4.4. SQL – SO SÁNH LIKE

Câu hỏi 22: so sánh chuỗi = chuỗi. Liệt kê mã nhân viên, ngày tháng năm sinh, mức lương của nhân viên có tên “Văn Mai Hương”?

```
SELECT MaNV, NTNS, Luong FROM NhanVien  
WHERE HoTen = 'Van Mai Huong'
```

Câu hỏi 23: Sử dụng LIKE (%: thay thế 1 chuỗi ký tự). Tìm những nhân viên có họ Nguyễn.

```
SELECT MaNV, HoTen FROM NhanVien WHERE HoTen like 'Nguyen %'
```

Câu hỏi 24 (tt): Tìm những nhân viên có tên Lan.

```
SELECT MaNV, HoTen FROM NhanVien WHERE HoTen like '% Lan'
```

Câu hỏi 25 (tt): Tìm những nhân viên có tên lót là “Thanh”.

```
SELECT MaNV, HoTen FROM NhanVien WHERE HoTen like '% Thanh %'
```

Câu hỏi 26: Sử dụng LIKE (_: thay thế 1 ký tự bất kỳ). Tìm những nhân viên tên có tên ‘Nguyễn La_’ (ví dụ Lam, Lan)

```
SELECT MaNV, HoTen FROM NhanVien WHERE HoTen like 'Nguyen La_'
```

4.5. Phép kết

Sử dụng **Phép kết bằng**: Để kết 02 hay nhiều bảng

- **Cách 1**: Sử dụng **inner join on** <điều kiện kết> ở **FROM**
- **Cách 2**: Đưa <điều kiện kết> xuống **WHERE**

Ví dụ: Liệt kê các nhân viên đang làm việc tại Phòng Nghiên cứu

C1: SELECT MaNV, HoTen, MaPH, TenPH
 FROM NhanVien inner join PhongBan
 ON Nhanvien.MaPH=PhongBan.MaPH
 AND PhongBan.TenPH = 'Nghien Cuu'

C2: SELECT MaNV, HoTen, MaPH, TenPH
 FROM NHANVIEN, PHONGBAN
 WHERE Nhanvien.MaPH=PhongBan.MaPH
 AND PhongBan.TenPH = 'Nghien Cuu'

4.5. SQL – PHÉP KẾT (Cách 1)

Cú pháp:

SELECT column_name

FROM table1 **INNER JOIN** table2

ON table1.column_name = table2.column_name

Cho biết mã số, họ tên, ngày tháng năm sinh của những nhân viên đã tham gia đề án

```
SELECT MaNV, HoTen, NTNS FROM NhanVien inner join
PhanCong
on Nhanvien.MaNV = PhanCong.MaNV
```

Cho biết họ tên nhân viên tham gia đề án 'TH01'

```
SELECT HoTen FROM NhanVien inner join PhanCong
ON NhanVien.MaNV = PhanCong.MaNV
WHERE MaDA = 'TH01'
```

4.5. SQL – PHÉP KẾT (Cách 2)

Câu 27. Phép kết: Cho biết mã số, họ tên, ngày tháng năm sinh của những nhân viên đã tham gia đề án?

```
SELECT MaNV, NTNS, Luong FROM NhanVien, PhanCong  
WHERE Nhanvien.MaNV=PhanCong.MaNV
```

Câu 28. Phép kết: Cho biết họ tên nhân viên tham gia đề án 'TH01'

```
SELECT HoTen FROM NhanVien, PhanCong  
WHERE MaDA='TH01' AND Nhanvien.MaNV=PhanCong.MaNV
```

Câu 29. Phép kết chính nó (Self join): Cho biết họ tên nhân viên và họ tên người quản lý của nhân viên đó

```
SELECT A.HoTen, B.HoTen FROM NhanVien as A, NhanVien as B  
WHERE A.MaNV=B.Ma_NQL
```

4.5. Phép kết

Giữa 2 cách kết:

- INNER JOIN thường rõ ràng hơn trong việc biểu diễn mối quan hệ giữa các bảng. Điều kiện kết nối được tách biệt rõ ràng trong mệnh đề ON, giúp cho việc đọc và hiểu dễ dàng hơn, đặc biệt là khi làm việc với nhiều bảng.
- Cách INNER JOIN giúp cơ sở dữ liệu tối ưu hóa truy vấn tốt hơn, đặc biệt là khi có nhiều bảng hoặc dữ liệu lớn. Trình tối ưu hóa truy vấn của SQL Server và các hệ quản trị cơ sở dữ liệu khác có thể xử lý các phép nối rõ ràng hiệu quả hơn.
- Cách WHERE: Sử dụng khi chỉ có một số điều kiện kết nối đơn giản hoặc đang làm việc với các truy vấn đơn giản, ít bảng hoặc làm việc với các truy vấn cũ, có thể từ hệ thống không hỗ trợ từ khóa INNER JOIN

4.5. Phép kết

JOIN và INNER JOIN:

- JOIN mặc định trong SQL là INNER JOIN, nên nếu chỉ dùng từ khóa JOIN mà không chỉ định, SQL sẽ tự hiểu đó là phép kết trong.
- Từ khóa INNER không bắt buộc. Tuy nhiên, INNER JOIN thường được sử dụng để đảm bảo tính rõ ràng trong câu lệnh SQL, đặc biệt trong các hệ thống lớn hoặc khi làm việc với nhiều loại phép kết khác nhau.

4.5. Phép kết

- **CROSS JOIN:** CROSS JOIN trong SQL là một phép kết hợp (join) đặc biệt giữa hai bảng, tạo ra tất cả các kết hợp có thể giữa các hàng của hai bảng đó. Kết quả của CROSS JOIN là một Cartesian product, nghĩa là nếu bảng A có m hàng và bảng B có n hàng thì kết quả của CROSS JOIN sẽ có $m \times n$ hàng.
- Cú pháp: `SELECT * FROM TableA CROSS JOIN TableB;`
- So sánh với phép kết không điều kiện: tương tự nhau.
- Điểm khác biệt:
 - Sử dụng CROSS JOIN: Khi user muốn rõ ràng rằng mục tiêu là tạo ra tất cả các kết hợp có thể giữa hai bảng (tích Cartesian).
 - Sử dụng JOIN không điều kiện kết: có thể thêm các điều kiện lọc bổ sung sau đó trong WHERE

4.5. Phép kết

Liệt kê tất cả các khách hàng và thông tin hóa đơn của họ. Kết quả cần bao gồm cả những khách hàng chưa có hóa đơn.

```
SELECT KH.MAKH, KH.HOTEN, HD.SOHD, HD.TRIGIA  
FROM KHACHHANG KH LEFT JOIN HOADON HD  
ON KH.MAKH = HD.MAKH;
```

Hiển thị thông tin tất cả các hóa đơn, bao gồm cả những hóa đơn mà nhân viên chưa có thông tin đầy đủ

```
SELECT HD.SOHD, HD.MANV, NV.HOTEN, NV.NGVL  
FROM HOADON HD RIGHT JOIN NHANVIEN NV  
ON HD.MANV = NV.MANV;
```


4.6. SQL – PHÉP HỢP, PHÉP GIAO

Truy vấn sử dụng phép toán tập hợp

Câu 30. Phép hợp: Sử dụng mệnh đề UNION hợp hai tập hợp. Cho biết họ tên nhân viên thuộc phòng 'NC' hoặc phòng 'DH'

```
(SELECT HoTen FROM NhanVien WHERE MaPH='NC')  
UNION  
(SELECT HoTen FROM NhanVien WHERE MaPH='DH')
```

Câu 31. Phép giao: Sử dụng mệnh đề INTERSECT giao hai tập hợp. Cho biết họ tên nhân viên thuộc cả hai phòng 'NC' và 'DH'

```
(SELECT HoTen FROM NhanVien WHERE MaPH='NC')  
INTERSECT  
(SELECT HoTen FROM NhanVien WHERE MaPH='DH')
```

4.6. SQL – PHÉP HỢP, PHÉP GIAO

Câu 32. Phép giao: Cho biết mã nhân viên tham gia cả hai đề án 'TH01' và 'TH02'

```
(SELECT MaNV FROM PhanCong WHERE MaDA='TH01')  
INTERSECT  
(SELECT MaNV FROM PhanCong WHERE MaDA='TH02')
```

Câu 33. Phép giao: Sử dụng từ khóa IN Cho biết họ tên nhân viên tham gia cả hai đề án 'TH01' và 'TH02'

```
(SELECT Hoten FROM NhanVien  
WHERE MaNV IN ((SELECT MaNV FROM PhanCong WHERE  
MaDA='TH01')  
INTERSECT  
(SELECT MaNV FROM PhanCong WHERE MaDA='TH02'))
```

4.7. SQL – PHÉP TRỪ

Câu 34. Phép trừ: Sử dụng mệnh đề **EXCEPT**.

Cho biết mã số của những nhân viên không tham gia đề án nào?

Gợi ý: Tìm tập hợp B “Nhân viên có tham gia đề án (bảng PhanCong), sau đó lấy phần bù bằng cách: Tất cả nhân viên tập hợp A trừ B.

```
(SELECT MaNV FROM NhanVien)
```

```
EXCEPT
```

```
(SELECT MaNV FROM PhanCong)
```

Câu 35. Phép trừ: Cho biết phòng ban không chủ trì các đề án triển khai năm 2023?

Gợi ý: thực hiện tìm tập hợp B “phòng ban chủ trì các đề án triển khai năm 2023”, sau đó lấy phần bù bằng cách: tất cả phòng ban (tập hợp A) trừ B

```
(SELECT MaPH FROM PhongBan)
```

```
EXCEPT
```

```
(SELECT Phong as MaPH FROM DEAN WHERE NamThucHien=2023)
```

4.7. SQL – PHÉP TRỪ

Câu 36. Phép trừ: Sử dụng NOT IN. Cho biết mã số, họ tên, ngày tháng năm sinh của những nhân viên không tham gia đề án nào?

```
SELECT MaNV, HoTen, NTNS FROM NhanVien  
WHERE MaNV NOT IN (select MaNV FROM PhanCong)
```

Câu 37. Phép trừ: Cho biết tên phòng ban không chủ trì các đề án triển khai năm 2023?

```
SELECT TenPH FROM PhongBan WHERE MaPH not in (select  
DISTINCT Phong FROM DeAn WHERE NamThucHien=2023)
```

4.8. SQL – TRUY VẤN LỒNG

Truy vấn lồng (Nested Queries) là một câu truy vấn được định nghĩa bên trong một câu truy vấn khác. Câu truy vấn được đặt bên trong một câu truy vấn khác được gọi là câu truy vấn con.

Cú pháp:

```
SELECT <danh sách các cột>
FROM <danh sách các bảng>
WHERE <so sánh tập hợp> / <kiểm tra sự tồn tại>
    (
        SELECT <danh sách các cột>
        FROM <danh sách các bảng>
        WHERE <điều kiện>
    )
```

4.8. SQL – TRUY VẤN LÒNG

- Các câu truy vấn có thể lồng nhau ở nhiều mức
- Các câu truy vấn con trong cùng một mệnh đề WHERE được kết hợp bằng phép nối logic
- Câu truy vấn con thường trả về một tập các giá trị
- So sánh tập hợp thường đi cùng một số toán tử như: IN, NOT IN, ALL, ANY hoặc SOME.
- Kiểm tra sự tồn tại: EXISTS, NOT EXISTS: Kiểm tra sự tồn tại trong kết quả của truy vấn con.

4.8. SQL – TRUY VẤN LÒNG

<So sánh tập hợp>: Truy vấn con trả về giá trị tập hợp

- <biểu thức> [NOT] IN (<truy vấn con>)
- <biểu thức> <phép toán so sánh> ANY (<truy vấn con>)
- <biểu thức> <phép toán so sánh> ALL (<truy vấn con>)

4.8. SQL – TRUY VẤN LÒNG

IN và EXISTS

- Cả IN và EXISTS đều được sử dụng trong SQL để kiểm tra sự tồn tại của một giá trị hoặc dòng trong một tập hợp hoặc kết quả của truy vấn con.

- **IN**: Kiểm tra xem **một giá trị** có **nằm trong** một tập hợp giá trị cụ thể hoặc kết quả trả về từ một truy vấn con hay không.

```
SELECT * FROM NhanVien WHERE MaNV IN (SELECT MaNV FROM PhanCong WHERE MaDA = 'DA01')
```

- **EXISTS**: Kiểm tra xem một truy vấn con có **trả về bất kỳ bản ghi/dòng** nào hay không.

```
SELECT * FROM NhanVien WHERE EXISTS (SELECT * FROM PhanCong WHERE PhanCong.MaNV = NhanVien.MaNV)
```


4.8. SQL – TRUY VẤN LÒNG

IN và EXISTS

Không phải lúc nào cũng có thể chuyển đổi trực tiếp giữa IN và EXISTS.

IN -> EXISTS: Có thể chuyển đổi hầu hết các trường hợp IN sang EXISTS, bằng cách sử dụng truy vấn con với điều kiện liên kết.

EXISTS -> IN: Khó hoặc không thể chuyển đổi trực tiếp trong một số trường hợp, đặc biệt khi truy vấn con trong EXISTS phức tạp hoặc không liên quan trực tiếp đến bảng chính.

Nên ưu tiên sử dụng EXISTS khi truy vấn con phức tạp hoặc trả về kết quả lớn.

4.8. SQL – TRUY VẤN LÒNG

IN và EXISTS

Ví dụ:

-- IN

```
SELECT * FROM NhanVien WHERE MaNV IN (SELECT MaNV  
FROM PhanCong WHERE MaDA = 'DA01');
```

-- EXISTS

```
SELECT * FROM NhanVien WHERE EXISTS (SELECT *  
FROM PhanCong WHERE PhanCong.MaNV =  
NhanVien.MaNV AND MaDA = 'DA01');
```

4.8. SQL – TRUY VẤN LÒNG

EXISTS được dùng để kết hợp với truy vấn con (subquery).
Điều kiện được đáp ứng nếu truy vấn con trả về ít nhất 1 hàng
(có thể nhiều)

Cú pháp:

SELECT *

FROM tên bảng

WHERE EXISTS

(SELECT tên cột FROM tên bảng WHERE điều kiện);

4.8. SQL – TRUY VẤN LÒNG

Điều kiện EXISTS có thể kết hợp với toán tử NOT.

Ví dụ:

```
SELECT * FROM NhanVien
```

```
WHERE NOT EXISTS
```

```
    (SELECT * FROM PhanCong
```

```
        WHERE NhanVien.MANV = PhanCong. MANV );
```

-- Kết quả trả về là tất cả các bộ/dòng trong bảng NhanVien nếu không có bộ/dòng nào trong bảng PhanCong khớp với thuộc tính MANV trong bảng NhanVien

4.8. SQL – TRUY VẤN LỒNG

❖ Có 2 loại truy vấn lồng (nested subqueries):

▪ Lồng phân cấp (Non-correlated subqueries)

- ✓ Đây là loại mà truy vấn con không phụ thuộc vào câu truy vấn bên ngoài.
- ✓ Mệnh đề WHERE trong câu truy vấn con không tham chiếu đến thuộc tính của các quan hệ trong mệnh đề FROM trong câu truy vấn chính (hay còn gọi là truy vấn cha)
- ✓ Khi thực hiện, câu truy vấn con sẽ thực hiện trước, và chỉ thực hiện 1 lần. Sau đó kết quả trả về sẽ được sử dụng để so sánh hoặc lấy dữ liệu từ truy vấn chính.
- ✓ Ví dụ:

SELECT TENSPP FROM SANPHAM WHERE GIA > (SELECT AVG(GIA) FROM SANPHAM);

- ✓ Truy vấn con (SELECT AVG(GIA) FROM SANPHAM) tính trung bình GIA của tất cả sản phẩm. Truy vấn chính sẽ lấy ra các TENSPP có GIA lớn hơn giá trị trung bình này. Truy vấn con này chỉ thực hiện một lần.

4.8. SQL – TRUY VẤN LÒNG

- **Lồng phân cấp (Non-correlated subqueries)**

Sơ đồ: Main Query (Outer Query) --> Nhận kết quả từ Subquery (Truy vấn con)

Ví dụ:

```
SELECT customer_name
FROM Customers
WHERE customer_id IN (
    SELECT customer_id
    FROM Orders
    WHERE order_date > '2024-01-01'
);
```

Truy vấn con lấy ra customer_id từ bảng Orders với các đơn hàng sau ngày 01/01/2024. Truy vấn chính sau đó sử dụng kết quả của truy vấn con để tìm tên khách hàng trong bảng Customers.

4.8. SQL – TRUY VẤN LÒNG

❖ Có 2 loại truy vấn lồng:

▪ Lồng tương quan (correlated subqueries)

- ✓ Là loại mà truy vấn con có phụ thuộc vào từng hàng của truy vấn ngoài. Phụ thuộc vào dữ liệu từ truy vấn bên ngoài, và truy vấn con sẽ thực thi cho mỗi hàng của truy vấn cha, vì nó sử dụng dữ liệu từ từng hàng đó để lọc kết quả
- ✓ Mệnh đề WHERE trong câu truy vấn con tham chiếu ít nhất một thuộc tính của các quan hệ trong mệnh đề FROM trong câu truy vấn chính.
- ✓ Khi thực hiện, câu truy vấn con sẽ được **thực hiện lặp lại nhiều lần**, mỗi lần tương ứng với một bộ của truy vấn chính.
- ✓ Dùng để so sánh từng hàng với một nhóm con của dữ liệu
- ✓ Sơ đồ:

Outer Query (Truy vấn ngoài) --> Từng hàng --> Subquery (Truy vấn con được thực thi)

4.8. SQL – TRUY VẤN LÒNG

Ví dụ:

```
SELECT SP1.TENSP  
FROM SANPHAM SP1  
WHERE GIA > (SELECT AVG(GIA)  
              FROM SANPHAM SP2  
              WHERE SP1.THELOAI = SP2.THELOAI);
```

- Truy vấn ngoài duyệt qua từng sản phẩm trong bảng SANPHAM.
- Đối với từng sản phẩm, truy vấn con (SELECT AVG(GIA) FROM SANPHAM SP2 WHERE SP1.THELOAI = SP2.THELOAI) tính trung bình GIA của các sản phẩm cùng THELOAI với sản phẩm trong SP1.
- Mỗi lần chạy truy vấn chính, truy vấn con sẽ được thực thi cho từng hàng của SP1.

4.8. SQL – TRUY VẤN LÒNG

- ❖ Cho lược đồ cơ sở dữ liệu quản lý hàng hóa như sau:
- ❖ KHACHHANG (MaKH, HoTen, DiaChi, DT, NGSinh, NgayDK, DoanhSo)
- ❖ NHANVIEN (MaNV, Hoten, SoDT, NgayLamviec)
- ❖ SANPHAM (MaSP, TenSP, DVT, NuocSX, Gia)
- ❖ HoaDon (SoHD, NGHD, MaKH, MaNV, TriGia)
- ❖ CTHD (SoHD, MaSP, SL)

4.8. SQL – TRUY VẤN LỒNG

❖ Ví dụ lồng phân cấp:

Câu 1: Tìm những nhân viên không lập hóa đơn nào

```
SELECT MaNV, HoTen, NTNS FROM NhanVien  
WHERE MaNV NOT IN (select MaNV FROM HOADON)
```

Câu 2: Tìm những nhân viên lập hóa đơn cho khách hàng có mã số là 'KH01'

```
SELECT MaNV, HoTen, NTNS FROM NhanVien  
WHERE MaNV IN (select MaNV FROM HOADON  
WHERE MaKH='KH01')
```

4.8. SQL – TRUY VẤN LÒNG

❖ Ví dụ lồng phân cấp:

Câu 3: Tìm những hóa đơn có trị giá lớn hơn trị giá của ít nhất một hóa đơn do nhân viên có mã 'NV01' lập

```
SELECT * FROM HoaDon  
WHERE TriGia > Any (select Trigia FROM Hoadon WHERE  
MaNV='NV01')
```

Câu 4: Tìm những hóa đơn có trị giá lớn hơn trị giá của tất cả hóa đơn do nhân viên có mã 'NV01' lập.

```
SELECT * FROM HoaDon  
WHERE TriGia > All (select Trigia FROM Hoadon WHERE  
MaNV='NV01')
```

4.8. SQL – TRUY VẤN LÒNG

❖ Ví dụ lồng tương quan:

Câu 1: Tìm những nhân viên lập hóa đơn cho khách hàng có mã số 'KH01'

```
SELECT MaNV, HoTen FROM NhanVien  
WHERE Exists (SELECT * FROM Hoadon WHERE  
MaKH='KH01' AND HoaDon.MaNV = NhanVien.MaNV)
```

Câu 2: Tìm những nhân viên không lập hóa đơn nào

```
SELECT * FROM NhanVien  
WHERE NOT Exists (SELECT * FROM Hoadon  
WHERE HoaDon.MaNV = NhanVien.MaNV)
```

Câu 3: Tìm những hóa đơn có trị giá lớn hơn trị giá của ít nhất một hóa đơn do nhân viên có mã số 'NV01' lập

```
SELECT * FROM HoaDon as HD1  
WHERE Exists (select * FROM HoaDon as HD2  
WHERE MaNV='NV01' AND HD1.Trigia > HD2.Trigia)
```

4.8. SQL – TRUY VẤN LÒNG

Đối với Exists không nhất thiết liệt kê tên thuộc tính ở mệnh đề SELECT của truy vấn con, chỉ cần SELECT *. Điều kiện EXISTS sẽ trả về TRUE nếu truy vấn con trả về ít nhất một bản ghi, ngược lại trả về FALSE.

Những câu truy vấn có ANY hoặc IN đều có thể chuyển thành câu truy vấn có Exists

4.8. SQL – TRUY VẤN LÒNG

Ví dụ: Liệt kê các mã nhân viên thực hiện cả 2 đề án DA01 và DA02

Cách 1:

```
SELECT MaNV  
FROM PHANCONG WHERE MaDA = 'DA01'  
AND MaNV IN (SELECT MaNV  
              FROM PHANCONG  
              WHERE MaDA = 'DA02')
```

B1: Truy vấn con: Lấy ra danh sách MaNV của những nhân viên tham gia đề án 'DA02'.

B2: Truy vấn chính: Chọn ra những nhân viên có MaNV nằm trong danh sách vừa lấy được ở bước 1, đồng thời cũng tham gia đề án 'DA01'.

4.8. SQL – TRUY VẤN LÒNG

Ví dụ: Liệt kê các mã nhân viên thực hiện cả 2 đề án DA01 và DA02

Cách 2:

```
SELECT MaNV
FROM PHANCONG PC_DA1
WHERE MaDA = 'DA01'
AND EXISTS (SELECT *
             FROM PHANCONG PC_DA2
             WHERE MaDA = 'DA02'
             AND PC_DA2.MaNV = PC_DA1.MaNV)
```

4.8. SQL – TRUY VẤN LÒNG

Ví dụ: Liệt kê các mã nhân viên thực hiện đề án DA01 nhưng không thực hiện đề án DA02

Cách 1:

```
SELECT MaNV
FROM PHANCONG
WHERE MaDA = 'DA01'
AND MaNV NOT IN (SELECT MaNV
                    FROM PHANCONG
                    WHERE MaDA = 'DA02')
```


4.8. SQL – TRUY VẤN LÒNG

Ví dụ: Liệt kê các mã nhân viên thực hiện đề án DA01 nhưng không thực hiện đề án DA02

Cách 2:

SELECT MaNV

FROM PHANCONG PC_DA1

WHERE MaDA = 'DA01'

AND NOT EXISTS (SELECT *

FROM PHANCONG PC_DA2

WHERE MaDA = 'DA02'

AND PC_DA2.MaNV = PC_DA1.MaNV

4.9. Phép chia tập hợp (÷)

- SQL không hỗ trợ một từ khóa đặc thù để thực hiện phép chia, vì thế chúng ta phải thể hiện nó thông qua sự hỗ trợ của các mệnh đề khác như JOIN, EXCEPT, IN, EXISTS.
- Thông thường phép chia thường được dùng với các câu truy vấn có chứa từ khoá **tất cả**, ví dụ như:
 - Tìm những người có tài khoản tại **tất cả** các ngân hàng trong một thành phố?
 - Tìm những sinh viên đã học xong **tất cả** các môn để đủ điều kiện tốt nghiệp.

4.9. Phép chia tập hợp (\div)

Review phép chia trong ĐSQH:

$R^+ = \{A, B, C, D, E\}$

| R | A | B | C | D | E |
|---|----------|---|----------|---|---|
| | α | a | α | a | 1 |
| | α | a | γ | a | 1 |
| | α | a | γ | b | 1 |
| | β | a | γ | a | 1 |
| | β | a | γ | b | 3 |
| | γ | a | γ | a | 1 |
| | γ | a | γ | b | 1 |
| | γ | a | γ | b | 1 |

$S^+ = \{D, E\}$

| S | D | E |
|---|---|---|
| | a | 1 |
| | b | 1 |

$$Y = R^+ - S^+$$

$Q^+ = \{A, B, C\}$

| Q | A | B | C |
|---|----------|---|----------|
| | α | a | γ |
| | γ | a | γ |

$Q = R \div S$

$$\begin{aligned} Q1 &\leftarrow \pi_y(R) \\ Q2 &\leftarrow Q1 \times (S) \\ Q3 &\leftarrow \pi_y(Q2 - R) \\ KQ &\leftarrow Q1 - Q3 \end{aligned}$$

Q: lấy ra những bộ trong R mà có liên kết với tất cả những bộ trong S

$R \div S$ là tập các giá trị a_i trong R sao cho **không có** giá trị b_i nào trong S làm cho bộ (a_i, b_i) **không tồn tại** trong R

4.9. Phép chia tập hợp (\div)

Ví dụ: Bài Quản lý Bán hàng

--- Câu 18: Tìm số hóa đơn đã mua tất cả các sản phẩm do Singapore sản xuất (phép chia)

SANPHAM (MASP, TENS_P, DVT, NUOCSX, GIA)

Tân từ: Mỗi sản phẩm có một mã số, một tên gọi, đơn vị tính, nước sản xuất và một giá bán.

HOADON (SOHD, NGHD, MAKH, MANV, TRIGIA)

Tân từ: Khi mua hàng, mỗi khách hàng sẽ nhận một hóa đơn tính tiền, trong đó sẽ có số hóa đơn, ngày mua, nhân viên nào bán hàng, trị giá của hóa đơn là bao nhiêu và mã số của khách hàng nếu là khách hàng thành viên.

CTHD (SOHD, MASP, SL)

Tân từ: Diễn giải chi tiết trong mỗi hóa đơn gồm có những sản phẩm gì với số lượng là bao nhiêu.

4.9. Phép chia tập hợp (÷)

S: các sản phẩm do Singapore sản xuất
(SANPHAM:NUOCSX='Singapore')[MASP]

R: Tìm số hóa đơn đã **mua** các **sản phẩm**
(HOADON_{SOHD}⋈CTHD) [SOHD, MASP]

R / S

4.9. Phép chia tập hợp (÷)

```
SELECT * FROM SANPHAM  
WHERE NUOCSX = 'Singapore'
```

| MASP | TENSP | DVT | NUOCSX | GIA |
|------|---------|-----|-----------|---------|
| BC01 | But chi | cay | Singapore | 3000.00 |
| BC02 | But chi | cay | Singapore | 5000.00 |

```
SELECT HD.SOHD  
FROM HOADON HD, CTHD CT  
WHERE CT.SOHD=HD.SOHD  
AND MASP = 'BC01'  
INTERSECT  
SELECT HD.SOHD  
FROM HOADON HD, CTHD CT  
WHERE CT.SOHD=HD.SOHD  
AND MASP = 'BC02'
```

SOHD

1001

```
SELECT HD.SOHD  
FROM HOADON HD, CTHD CT  
WHERE CT.SOHD=HD.SOHD  
AND MASP = 'BC01'  
UNION  
SELECT HD.SOHD  
FROM HOADON HD, CTHD CT  
WHERE CT.SOHD=HD.SOHD  
AND MASP = 'BC02'
```

SOHD

1001

1014

4.9. Phép chia tập hợp (÷)

Biểu diễn phép chia trong SQL:

- 1. Sử dụng NOT EXISTS + NOT EXISTS**
- 2. Sử dụng Sử dụng NOT EXISTS + EXCEPT**
- 3. Sử dụng GROUP BY + HAVING**
- 4. Một số cách khác như NOT IN + NOT IN, NOT IN + EXCEPT, NOT EXISTS + NOT IN, CROSS JOIN + EXCEPT... sinh viên tự tìm hiểu thêm**

4.9. Phép chia tập hợp (\div)

1. Sử dụng NOT EXISTS + NOT EXISTS để biểu diễn

- **Từ ý nghĩa:** $R \div S$ là tập các giá trị a_i trong R sao cho **không** có giá trị b_i nào trong S làm cho bộ (a_i, b_i) **không** tồn tại trong R
- **Ý tưởng:** Kiểm tra xem có **bất kỳ** giá trị nào trong tập hợp cần **kiểm tra** mà **không có liên kết** với đối tượng cần tìm hay không. Nếu **không** có thì đối tượng đó thỏa mãn điều kiện phép chia.
- Hiệu quả tốt, tối ưu cho các hệ thống lớn

4.9. Phép chia tập hợp (\div)

Cú pháp:

```
SELECT (DISTINCT) R1.A
FROM R R1
WHERE NOT EXISTS (
    SELECT *
    FROM S
    WHERE NOT EXISTS (
        SELECT *
        FROM R R2
        WHERE R2.B=S.B
        AND R1.A=R2.A)
```

4.9. Phép chia tập hợp (\div)

Cách thực hiện ví dụ câu 18:

Kiểm tra từng hóa đơn, nếu hóa đơn đó mua tất cả sản phẩm do Singapore sản xuất thì hóa đơn đó thỏa mãn điều kiện.

(tương đương ~) Kiểm tra từng hóa đơn, nếu không tồn tại sản phẩm nào của Singapore mà hóa đơn đó chưa mua, thì hóa đơn đó thỏa mãn điều kiện.

(tương đương ~) Kiểm tra từng hóa đơn, nếu danh sách các sản phẩm do Singapore sản xuất mà hóa đơn đó chưa mua là rỗng, thì hóa đơn đó thỏa mãn điều kiện.

(tương đương ~) Kiểm tra từng hóa đơn, nếu có sản phẩm nào do Singapore sản xuất mà **không** có trong hóa đơn, thì hóa đơn đó **không** thỏa mãn điều kiện.

4.9. Phép chia tập hợp (÷)

Ví dụ: câu 18

```
SELECT DISTINCT HD.SOHD
FROM HOADON HD
WHERE NOT EXISTS (
    SELECT *
    FROM SANPHAM SP
    WHERE NUOCSX='SINGAPORE'
    AND NOT EXISTS (
        SELECT *
        FROM CTHD CT
        WHERE CT.MASP=SP.MASP
        AND CT.SOHD=HD.SOHD))
```

4.9. Phép chia tập hợp (÷)

- `SELECT * FROM CTHD CT WHERE CT.MASP=SP.MASP AND CT.SOHD=HD.SOHD`: Lấy tất cả các chi tiết hóa đơn, đảm bảo rằng sản phẩm từ Singapore (`SP.MASP=CT.MASP`) tồn tại trong chi tiết hóa đơn của hóa đơn hiện tại (`HD.SOHD=CT.SOHD`)
- `SELECT * FROM SANPHAM SP WHERE NUOCSX='Singapore' AND NOT EXISTS (...)`: các sản phẩm từ Singapore không nằm trong hóa đơn hiện tại
- `SELECT DISTINCT HD.SOHD FROM HOADON HD WHERE NOT EXISTS (...)`: kiểm tra để đảm bảo rằng không có sản phẩm nào từ Singapore bị thiếu trong hóa đơn hiện tại (`HD.SOHD`)

4.9. Phép chia tập hợp (\div)

2. Sử dụng NOT EXISTS + EXCEPT

- **Ý tưởng:** Lấy hiệu giữa tập hợp các giá trị thuộc tính trong R và tập hợp các giá trị thuộc tính tương ứng trong S, sử dụng EXCEPT để thực hiện phép trừ tập hợp.
- **Tính chất:** Cú pháp ngắn gọn hơn so với NOT EXISTS & NOT EXISTS. Hiệu năng: EXCEPT có thể chậm hơn NOT EXISTS. Logic phức tạp hơn, có thể khó hiểu đối với người mới học.

4.9. Phép chia tập hợp (\div)

Cú pháp:

SELECT R1.A

FROM R R1

WHERE NOT EXISTS (

(SELECT S.D FROM S WHERE)

EXCEPT

(SELECT R2.D FROM R R2

WHERE R1.A=R2.A)

)

4.9. Phép chia tập hợp (\div)

Tìm các sản phẩm bị thiếu từ Singapore và loại bỏ hóa đơn nếu có sản phẩm thiếu

4.9. Phép chia tập hợp (÷)

Ví dụ: câu 18

```
SELECT H.SOHD
FROM HOADON H
WHERE NOT EXISTS (
    SELECT MASP
    FROM (SELECT MASP
          FROM SANPHAM
          WHERE NUOCSX = 'Singapore'
          EXCEPT
          SELECT MASP
          FROM CTHD C
          WHERE C.SOHD = H.SOHD
         ) AS MissingProducts
);
```


4.9. Phép chia tập hợp (÷)

- `SELECT MASP FROM CTHD C WHERE C.SOHD = H.SOHD`: kiểm tra xem sản phẩm nào từ Singapore có trong chi tiết hóa đơn hiện tại.
- `SELECT MASP FROM (SELECT MASP FROM SANPHAM WHERE NUOCSX = 'Singapore' EXCEPT ...`: danh sách các sản phẩm từ Singapore mà hóa đơn cần chứa, trừ đi kết quả câu trên.
- `AS MissingProducts`: tập sản phẩm gồm các sản phẩm từ Singapore mà không có trong CTHD
- `SELECT H.SOHD FROM HOADON H WHERE NOT EXISTS`: Chọn những số HD mà không chứa bất kỳ sản phẩm nào từ Singapore bị thiếu sẽ được chọn

4.9. Phép chia tập hợp (\div)

3. Sử dụng GROUP BY + HAVING

- **Ý tưởng:** Sử dụng GROUP BY để nhóm các đối tượng cần kiểm tra trong R và sử dụng HAVING để lọc ra những đối tượng thỏa mãn điều kiện có số lượng phần tử bằng với số lượng phần tử trong S

Đếm số lượng đối tượng B mà A liên kết và so sánh với tổng số lượng B trong bảng S. Nếu số lượng này bằng nhau thì A thỏa mãn điều kiện phép chia.

Tính chất: Cú pháp tương đối đơn giản, dễ hiểu. Ít hiệu quả hơn các phương pháp khác, đặc biệt với cơ sở dữ liệu lớn.

4.9. Phép chia tập hợp (÷)

Cú pháp

SELECT R.A

FROM R

[WHERE R.B IN (SELECT S.B FROM S [WHERE <Điều kiện>])]

GROUP BY R.A

HAVING COUNT(DISTINCT R.B) = (SELECT COUNT(S.B)

FROM S [WHERE <Điều kiện>])

4.9. Phép chia tập hợp (\div)

Đếm số sản phẩm Singapore trong từng hóa đơn và so sánh với tổng số sản phẩm Singapore để tìm hóa đơn đủ sản phẩm

4.9. Phép chia tập hợp (÷)

Ví dụ: câu 18

```
SELECT H.SOHD
FROM HOADON H
JOIN CTHD C ON H.SOHD = C.SOHD
JOIN SANPHAM S ON C.MASP = S.MASP
WHERE S.NUOCSX = 'Singapore'
GROUP BY H.SOHD
HAVING COUNT(DISTINCT S.MASP) = (SELECT COUNT(*)
                                FROM SANPHAM
                                WHERE NUOCSX = 'Singapore');
```

4.9. Phép chia tập hợp (÷)

- SELECT H.SOHD, COUNT(DISTINCT S.MASP) FROM HOADON H JOIN CTHD C ON H.SOHD = C.SOHD JOIN SANPHAM S ON C.MASP = S.MASP WHERE S.NUOCSX = 'Singapore' GROUP BY H.SOHD

| SOHD | No column name |
|------|----------------|
| 1001 | 2 |
| 1014 | 1 |

- SELECT COUNT(*) FROM SANPHAM WHERE NUOCSX = 'Singapore': Truy vấn con này đếm tổng số sản phẩm từ Singapore trong bảng SANPHAM

No column name

2

- HAVING COUNT(DISTINCT S.MASP) = ...: Đếm số lượng mã sản phẩm khác nhau (MASP) từ Singapore có trong hóa đơn hiện tại

- SELECT H.SOHD FROM HOADON H JOIN CTHD C ON H.SOHD = C.SOHD JOIN SANPHAM S ON C.MASP = S.MASP WHERE S.NUOCSX = 'Singapore' GROUP BY H.SOHD: Nhóm kết quả theo SOHD để tính toán trên mỗi hóa đơn riêng biệt, tính số lượng sản phẩm từ Singapore trong mỗi hóa đơn.

4.10. HÀM COUNT,SUM,MAX,MIN,AVG

a) Sử dụng các hàm COUNT, SUM, MIN, MAX, AVG trên 1 nhóm lớn (trên toàn bộ quan hệ):

– Câu hỏi 1: Số lượng nhân viên của công ty.

```
SELECT COUNT(MaNV) as SoNV FROM  
NhanVien
```

– Câu hỏi 2: Số lượng nhân viên quản lý trực tiếp nhân viên khác.

```
SELECT COUNT (DISTINCT Ma_NQL) FROM  
NhanVien
```

– Câu hỏi 3: Tìm mức lương lớn nhất, mức lương trung bình, tổng lương của công ty.

```
SELECT MAX(Luong), AVG(Luong), SUM(Luong) FROM  
NhanVien
```

– Câu hỏi 4: Cho biết nhân viên có mức lương lớn nhất.

```
SELECT * FROM NhanVien  
WHERE Luong = (SELECT MAX(Luong) FROM  
NhanVien )
```

4.11. SQL – MỆNH ĐỀ GROUP BY

Quan hệ NV

nhóm

| Q | S |
|---|----|
| a | 10 |
| a | 2 |
| b | 9 |
| b | 5 |
| c | 10 |
| c | 8 |
| c | 6 |
| c | 4 |
| c | 10 |
| d | 16 |
| d | 18 |
| d | 50 |

Chia các dòng thành các nhóm dựa trên tập thuộc tính chia nhóm

| Q | Count(S) |
|---|----------|
| a | 2 |
| b | 2 |
| c | 5 |
| d | 3 |

Tương tự cho các hàm SUM, MIN, MAX, AVG

Các thuộc tính GROUP BY: Q

Câu SQL:
SELECT Q, **count**(S)
FROM NV
Group by Q

4.11. SQL – MỆNH ĐỀ GROUP BY

Câu hỏi 5: Cho biết nhân viên có mức lương trên mức lương trung bình của công ty.

```
SELECT HoTen FROM NhanVien WHERE Luong >  
(SELECT AVG(Luong) FROM NhanVien )
```

b) Sử dụng các hàm COUNT, SUM, MIN, MAX, AVG trên từng nhóm nhỏ: mệnh đề GROUP BY

- Chia các dòng thành các nhóm nhỏ dựa trên tập thuộc tính chia nhóm.
- Thực hiện các phép toán trên nhóm như: Count (thực hiện phép đếm), Sum (tính tổng), Min(lấy giá trị nhỏ nhất), Max(lấy giá trị lớn nhất), AVG (lấy giá trị trung bình).

4.11. SQL – MỆNH ĐỀ GROUP BY

Câu hỏi 6: Cho biết số lượng nhân viên theo từng phái (giới tính)?

Do cột phái có 2 giá trị “nam” và “nữ”, trường hợp này ta chia bảng NhanVien thành 2 nhóm nhỏ. Thuộc tính chia nhóm là thuộc tính “Phai”.

➡ **SELECT Phai, count(MaNV) as SoNV FROM NhanVien
Group by Phai**

Câu hỏi 7: Cho biết số lượng nhân viên theo từng phòng?

Do cột MaPH có 3 giá trị “NC” và “DH” và “QL”, trường hợp này ta chia bảng nhân viên thành 3 nhóm nhỏ. Thuộc tính chia nhóm là thuộc tính “MaPH”.

➡ **SELECT MaPH, count(Manv) FROM NhanVien Group by
MaPH**

Tương tự: cho biết tổng lương của mỗi phòng, cho biết mức lương thấp nhất của từng phòng, mức lương cao nhất, mức lương trung bình của từng phòng

4.11. SQL – MỆNH ĐỀ GROUP BY

Câu hỏi 8: Cho biết tên phòng và số lượng nhân viên theo từng phòng?

Thuộc tính chia nhóm là (TenPH) thay cho MaPH.

➔ **SELECT** TenPH, count(Manv) **as** SoLuongNV
FROM NhanVien n, PhongBan p **WHERE** n.MaPh=p.MaPH
Group by TenPH

Câu hỏi 9: Với mỗi phòng, cho biết số lượng nhân viên theo từng phái?

Do cột MaPH có 3 giá trị “NC” và “DH” và “QL”, mỗi phòng chia nhỏ theo từng phái: 2 nhóm “Nam” và “Nữ”, trường hợp này ta chia bảng nhân viên thành 6 nhóm nhỏ. Như vậy, tập thuộc tính chia nhóm cho câu truy vấn là (Phong, Phai).

➔ **SELECT** MaPH, Phai, count(MaNV) **FROM** NhanVien
Group by MaPH, Phai

4.11. SQL – MỆNH ĐỀ GROUP BY

Câu hỏi 10: Đếm số đề án của từng nhân viên tham gia?

- Do cột MaNV có 7 giá trị “NV001”,...”NV008” (không có nhân viên “005”), trường hợp này ta chia bảng PhanCong thành 7 nhóm nhỏ. Với mỗi nhóm nhỏ (MaNV), ta đếm số đề án (count(MADA)) tham gia. Thuộc tính chia nhóm là thuộc tính “MaNV”.

➔ **SELECT** MaNV, count(MaDA) **as** SoDATG **FROM**
PhanCong
Group by MaNV

- Tương tự: tính tổng số giờ làm việc của mỗi nhân viên (SUM), thời gian làm việc thấp nhất của mỗi nhân viên (MIN), thời gian làm việc lớn nhất của mỗi nhân viên (MAX), thời gian làm việc trung bình,...

4.11. SQL – MỆNH ĐỀ GROUP BY

Câu hỏi 11: Cho biết mã, tên nhân viên và số đề án mà n/v đã tham gia?



```
SELECT n.MaNV, HoTen, count(MaDA) as SoDATG  
FROM PhanCong pc, NhanVien n WHERE  
pc.manv=n.manv Group by MaNV, HoTen
```

Lưu ý: Các thuộc tính trong mệnh đề SELECT (trừ những thuộc tính trong các hàm kết hợp) phải xuất hiện trong mệnh đề GROUP BY.

4.11. SQL – MỆNH ĐỀ GROUP BY

Bổ sung: các lỗi thường gặp khi sử dụng mệnh đề GROUP BY

```
SELECT MaNV, TenNV, SUM(Luong)
```

```
FROM NhanVien
```

```
GROUP BY MaNV;
```

- Lỗi: TenNV không xuất hiện trong GROUP BY.
- SQL Server sẽ báo lỗi thực thi.
- Cách sửa lỗi ?

4.11. SQL – MỆNH ĐỀ GROUP BY

Bổ sung: các lỗi thường gặp khi sử dụng mệnh đề **GROUP BY**

```
SELECT MaNV, SUM(Luong)
```

```
FROM NhanVien;
```

- Lỗi: không có GROUP BY
- SQL Server sẽ báo lỗi thực thi.

```
SELECT MaNV, SUM(Luong)
```

```
FROM NhanVien
```

```
HAVING SUM(Luong) > 10000000;
```

- Lỗi: không có GROUP BY mà có HAVING
- SQL Server sẽ báo lỗi thực thi.

4.11. SQL – MỆNH ĐỀ GROUP BY

Bổ sung: các lỗi thường gặp khi sử dụng mệnh đề GROUP BY

```
SELECT MaNV, SUM(Luong), MAX(NgayLamViec)
```

```
FROM NhanVien
```

```
GROUP BY MaNV, SUM(Luong);
```

- Lỗi: dư ở GROUP BY hàm SUM

```
SELECT MaNV, SUM(Luong)
```

```
FROM NhanVien
```

```
GROUP BY MaNV
```

```
ORDER BY Luong;
```

- Lỗi: cột Luong không tồn tại trong danh sách SELECT

4.12. SQL – MỆNH ĐỀ HAVING

- Lọc kết quả theo điều kiện, sau khi đã gom nhóm
- Điều kiện của HAVING là điều kiện về các hàm tính toán trên nhóm (Count, Sum, Min, Max, AVG) và các thuộc tính trong danh sách GROUP BY.

Câu hỏi 12: Cho biết những nhân viên tham gia từ 2 đề án trở lên?

```
SELECT MaNV, count(MaDA) as SoDATG FROM  
PhanCong  
Group by MaNV  
Having count(MaDA) >=2
```

Câu hỏi 13: Cho biết mã phòng ban có trên 4 nhân viên?

```
SELECT MaPH, count(Manv) FROM NhanVien Group by  
MaPH  
Having count(Manv)>4
```

4.12. SQL – MỆNH ĐỀ HAVING

- Sử dụng các hàm kết hợp trong mệnh đề SELECT để kiểm tra một số điều kiện nào đó
- Chỉ kiểm tra điều kiện trên nhóm, không là điều kiện lọc trên từng bộ
- Sau khi gom nhóm, điều kiện trên nhóm mới được thực hiện

4.12. SQL – MỆNH ĐỀ HAVING

- ❖ Thứ tự thực hiện câu truy vấn có mệnh đề GROUP BY VÀ HAVING:
 - Chọn ra những dòng thỏa điều kiện trong mệnh đề WHERE
 - Những dòng này sẽ được gom thành nhiều nhóm tương ứng với mệnh đề GROUP BY
 - Áp dụng các hàm kết hợp cho mỗi nhóm
 - Bỏ qua những nhóm không thỏa điều kiện trong mệnh đề HAVING
 - Rút trích các giá trị của các cột và hàm kết hợp trong mệnh đề SELECT

4.13. MỘT SỐ VÍ DỤ KHÁC

Câu 1: Cho biết giá trị hóa đơn cao nhất, thấp nhất

```
SELECT Max(Trigia) as Trigiacaonhat, Min(Trigia) as  
Trigiathapnhap FROM HOADON
```

Câu 2: Cho biết số lượng hóa đơn do nhân viên “Nguyen Van A” lập.

```
SELECT count(*) as soluonghoadon  
FROM HOADON HD, NHANVIEN NV  
WHERE HD.MANV=NHANVIEN.MANV AND HOTEN=“Nguyen Van  
A ”
```

4.13. MỘT SỐ VÍ DỤ KHÁC

Câu 3: Tính doanh thu bán hàng trong năm 2022

```
SELECT SUM(Trigia) as Doanhthu_Nam2022  
FROM HOADON  
WHERE YEAR(NGHD)='2022'
```

Câu 4: Cho biết trị giá trung bình của tất cả các hóa đơn trong năm 2022

```
SELECT AVG(Trigia) as TrigiaTrungbinh_Nam2022  
FROM HOADON  
WHERE YEAR(NGHD)='2022'
```

4.13. MỘT SỐ VÍ DỤ KHÁC

Câu 5: Cho biết số lượng sản phẩm của từng nước?

```
SELECT NuocSX, count(*) as SoluongSP  
FROM SANPHAM  
GROUP BY NuocSX
```

Câu 6: Với mỗi nhân viên cho biết mã số, số lượng hóa đơn và tổng trị giá mà họ đã lập

```
SELECT MaNV, count (*) as SL_HD, sum (Trigia) as Tong_Trigia_HD  
FROM HOADON  
GROUP BY MANV
```

4.13. MỘT SỐ VÍ DỤ KHÁC

Câu 7: Cho biết những nhân viên lập từ 2 hóa đơn trở lên?

```
SELECT MaNV FROM HOADON  
GROUP BY MANV  
HAVING Count(*) >=2
```

Câu 8: Cho biết những nhân viên (HOTEN) có trị giá hóa đơn đã lập trung bình lớn hơn 1 triệu đồng

```
SELECT Hoten, avg(Trigia) as Trigia_TB  
FROM HOADON AS HD, NHANVIEN AS NV  
WHERE NHANVIEN.MANV=HOADON.MANV  
GROUP BY HOTEN  
HAVING AVG(Trigia)>1000000
```

4.13. MỘT SỐ VÍ DỤ KHÁC

Câu 9: Tìm những nhân viên có trị giá hóa đơn đã lập trung bình cao nhất

```
SELECT MaNV, AVG(TriGia) as Trigia_TB
```

```
FROM HOADON
```

```
GROUP BY MANV
```

```
HAVING Trigia_TB >=All
```

```
(SELECT AVG(TriGia)
```

```
FROM HOADON
```

```
GROUP BY MANV)
```


4.13. MỘT SỐ VÍ DỤ KHÁC

Trở lại bài lược đồ CSDL “quản lý đề án công ty”:

NHANVIEN (MaNV, HoTen, Phai, Luong, NTNS, Ma_NQL, MaPH)

PHONGBAN (MaPH, TenPH, TRPH)

DEAN (MaDA, TenDA, Phong, NamThucHien)

PHANCONG (MaNV, MaDA, ThoiGian)

Tìm nhân viên **tham gia tất cả các đề án** của công ty

4.13. MỘT SỐ VÍ DỤ KHÁC

Tìm nhân viên tham gia tất cả các đề án của công ty

→ Tìm nhân viên **không** thuộc danh sách những nhân viên **không** thực hiện ít nhất 1 đề án?

SELECT MaNV, HoTen

FROM NHANVIEN NV

WHERE NOT EXISTS (SELECT *

FROM DEAN DA

WHERE NOT EXISTS (SELECT *


FROM PHANCONG PC

WHERE PC.MaDA = DA.MaDA

AND PC.MaNV = NV.MaNV))

Tài liệu tham khảo

1. Slides bài giảng Quản lý dự án CNTT, Khoa HTTT, Trường Đại học CNTT.
2. ThS. Nguyễn Thị Kim Phụng, Slides bài giảng Cơ sở dữ liệu, Khoa HTTT, Trường Đại học CNTT
3. ThS. Nguyễn Hải Châu, Slides bài giảng Cơ sở dữ liệu, Đại học Công nghệ, ĐH Quốc gia Hà Nội
4. Ramez Elmasri, Shamkant B. Navathe, Fundamentals of Database Systems, Seven Edition, 2016



THANK YOU!

Q & A

ThS. TẠ VIỆT PHƯƠNG
phuongtv@uit.edu.vn