

Chapter 5

# SQL (Structured Query Language)



KHOA CÔNG NGHỆ THÔNG TIN  
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

[fit@hcmus](mailto:fit@hcmus)

# Content

- Introduction
- Data definition
- Data manipulation
  - Query
  - Data update
- View
- Index

# Introduction

- Relational algebra language (ĐSQH )
  - How to execute the query operations
  - Difficult for users
- SQL (Structured Query Language)
  - High level declarative language interface
  - The user only specifies what the result is to be
  - Developed by IBM (1970s)
  - Also pronounced SEQUEL
  - SQL became a standard of American National Standards Institute (ANSI)
    - SQL-86
    - SQL-92
    - SQL-99

# Introduction

## ■ SQL includes

- Data definition language (DDL)
- Data manipulation language (DML)
- View definition
- Integrity constraint
- Authorization and security
- Transaction control

**SQL-92 standard**

## ■ SQL terms

- Table ~ relation
- Column ~ attribute
- Row ~ tuple

**SQL Server**

# Content

- Introduction
- **Data definition**
  - Data type
  - Data definition commands
- Data manipulation
- View
- Index

# Data definition

- Describes the structure of information in the DB
  - Schema for the relation
  - Domain of each attribute
  - Integrity constraint
  - Index on each relation
- Consists of
  - CREATE TABLE
  - DROP TABLE
  - ALTER TABLE
  - CREATE DOMAIN
  - CREATE DATABASE

# Data definition

## ■ Numeric

- INTEGER
- SMALLINT
- NUMERIC, NUMERIC(p), NUMERIC(p,s)
- DECIMAL, DECIMAL(p), DECIMAL(p,s)
- REAL
- DOUBLE PRECISION
- FLOAT, FLOAT(p)

# Data definition

## ■ Character string

- CHARACTER (CHAR)
- CHARACTER(n) (CHAR (n))
- CHARACTER VARYING(n) (VARCHAR(n))
- NATIONAL CHARACTER (n) (NCHAR(n))

## ■ Bit string

- BIT, BIT(x)
- BIT VARYING(x)

## ■ Datetime

- DATE (M/D/YY), DATETIME, TIMESTAMP (M/D/YY hh:mm)
- TIME (hh:mm)

# Create table command

- Define a new table by giving
  - A name
  - Attributes
    - Name
    - Date type
    - Integrity constraints on attribute
- Syntax

```
CREATE TABLE <Table_name> (
    <Column_name> <Data_type> [<Constraint>],
    <Column_name> <Data_type> [<Constraint>],
    ...
    [<Constraint>]
)
```

# Example

- GIAOVIEN table includes:

- **MaGV**: primary key
- **HoTen**: not null
- **Luong**: default value = 1000
- **Phai**: "Nam" or "Nữ"
- **NgaySinh**: date of birth
- **SoNha, Duong, Quan, ThanhPho**
- **GVQLCM**: foreign key
- **MaBM**: foreign key

# Example

```
CREATE TABLE GIAOVIEN (
    MaGV           CHAR(9),
    HoTen          NVARCHAR(50),
    Luong          INT,
    Phai           CHAR(3),
    NgaySinh       DATETIME,
    SoNha          NVARCHAR(10),
    Duong          NVARCHAR(50),
    Quan           NVARCHAR(50),
    ThanhPho        NVARCHAR(50),
    GVQLCM         CHAR(9),
    MaBM           CHAR(9)
)
```

# Create table command

## ■ <Constraint>

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

## ■ Give a name to constraints

```
CONSTRAINT <Constraint_name> <Constraint>
```

# Example

```
CREATE TABLE GIAOVIEN (
    MAGV             CHAR(9) PRIMARY KEY,
    HOTEN            NVARCHAR(50) NOT NULL,
    LUONG             INT DEFAULT (1000),
    PHAI              CHAR(3) CHECK (PHAI IN('Nam', 'Nu')),
    NGAYSINH          DATETIME,
    SONHA             NVARCHAR(10),
    DUONG             NVARCHAR(50),
    QUAN              NVARCHAR(50),
    THANHPHO           NVARCHAR(50),
    GVQLCM            CHAR(9),
    MABM              CHAR(9)
)
```

# Example

```
CREATE TABLE CONGVIEC (
    MADT  VARCHAR(10),
    STT   INT,
    TENCV NVARCHAR(50),
    NGAYBD DATETIME,
    NGAYKT DATETIME,
    PRIMARY KEY(MADT, STT)
)
```

```
CREATE TABLE DETAI (
    MADT  VARCHAR (10) PRIMARY KEY,
    TENDT NVARCHAR (50) UNIQUE,
    KINHPHI INT,
    CAPQL NVARCHAR (50),
    NGAYBD DATETIME,
    NGAYKT DATETIME,
    MACD  VARCHAR (10),
    GVCNDT CHAR (9)
)
```

# Example

```
CREATE TABLE GIAOVIEN (
```

```
    MAGV           CHAR(9) CONSTRAINT PK_GV PRIMARY KEY,  
    HOTEN          NVARCHAR(50) CONSTRAINT NN_HOTEN NOT NULL,  
    LUONG          INT CONSTRAINT DE_LUONG DEFAULT (10000),  
    PHAI           CHAR(3)  CONSTRAINT CK_PHAI CHECK (PHAI IN('Nam', 'Nu'))  
    CONSTRAINT NN_PHAI NOT NULL,  
    NGAYSINH       DATETIME,  
    SONHA          NVARCHAR(10),  
    DUONG          NVARCHAR(50),  
    QUAN           NVARCHAR(50),  
    THANHPHO        NVARCHAR(50),  
    GVQL            CHAR(9),  
    MABM            CHAR(9)
```

# Example

```
CREATE TABLE CONGVIEC(  
    MADT  VARCHAR(10),  
    STT   INT,  
    TENCV NVARCHAR(50),  
    NGAYBD DATETIME,  
    NGAYKT DATETIME,  
  
    CONSTRAINT PK_CV PRIMARY KEY(MADT, STT),  
    CONSTRAINT FK_CONGVIEC_DETAI  
        FOREIGN KEY MADT REFERENCES DETAI(MADT)  
)
```

# Create table command

- Is used for modification
  - The structure of tables
  - Integrity constraints
- Add columns

```
ALTER TABLE <Table_name> ADD  
<Column_name> <Data_type> [<Constraint>]
```

- Drop columns

```
ALTER TABLE <Table_name> DROP COLUMN <Column_name>
```

- Alter columns

```
ALTER TABLE <Table_name> ALTER COLUMN  
<Column_name> <Data_type>
```

# Alter table command

- Add constraints

```
ALTER TABLE <Table_name> ADD  
    CONSTRAINT <Constraint_name> <constraint>,  
    CONSTRAINT <Constraint_name> <constraint>,  
    ...
```

- Drop constraints

```
ALTER TABLE <Table_name> DROP <Constraint_name>
```

# Example

```
ALTER TABLE GIAOVIEN ADD TUOI INT  
CONSTRAINT CK_TUOI CHECK (TUOI >= 23 AND TUOI <=60) NOT NULL
```

```
ALTER TABLE GIAOVIEN DROP COLUMN HOTEN
```

```
ALTER TABLE GIAOVIEN ALTER COLUMN HOTEN  
NVARCHAR(100)
```

# Example

CREATE TABLE BOMON(

MABM	INT NOT NULL,
TENBM	NVARCHAR(50),
PHONG	CHAR(10),
DIENTHOAI	CHAR(15),
TRUONGBM	CHAR(9),
MAKHOA	CHAR(4),
NGAYNHANCHUC	DATETIME

***PRIMARY KEY constraint must be defined as NOT NULL***

)

ALTER TABLE BOMON ADD

CONSTRAINT PK\_BOMON **PRIMARY KEY** (MABM),

CONSTRAINT FK\_TRBOMON **FOREIGN KEY** (TRUONGBM)  
REFERENCES GIAOVIEN(MAGV),

***Must ensure that GIAOVIEN is existed***

CONSTRAINT UNI\_TENBM **UNIQUE**(TENBM),

CONSTRAINT DF\_NGAYNHANCHUC **DEFAULT**(GETDATE()) FOR NGAYNHANCHUC

# Drop table command

- Is used for deleting the structure of tables
  - All the data in a table are also deleted
- Syntax

```
DROP TABLE <Table_name>
```

- Example

DROP TABLE GIAOVIEN

DROP TABLE BOMON

DROP TABLE THAMGIADT

# Drop table command

GIÁO VIÊN

<u>MãGV</u>	Họ Tên	Lương	Phái	Ngày Sinh	<del>Số Nhà</del>	Đường	Quận	Thành Phố	<del>GVQL</del>	<del>MãBM</del>
-------------	--------	-------	------	-----------	-------------------	-------	------	-----------	-----------------	-----------------



BỘ MÔN

<u>MãKhoa</u>	<u>Trưởng BM</u>	Ngày Nhận	<del>Chức</del>	Điện Thoại	<u>MãBM</u>	Tên BM	Phòng
---------------	------------------	-----------	-----------------	------------	-------------	--------	-------

KHOA

<u>Trưởng Khoa</u>	<u>Ngày Nhận Chủ</u>	Tên Khoa	Năm TL	Phòng	Điện Thoại	<u>MãKhoa</u>
--------------------	----------------------	----------	--------	-------	------------	---------------

# Create type command

- Is used for creating a new data type
- Syntax

```
CREATE TYPE <New_data_type> AS <Data_type>
```

- Ví dụ

```
CREATE TYPE MyString30 AS VARCHAR(30)
```

# Content

- Introduction
- Data definition
- **Data manipulation**
  - **Basic queries**
  - Set, set/multiset comparison and nested queries
  - Aggregate functions and grouping
- View
- Index

# Basic Query

- Is used for retrieving some tuples that often satisfy a certain condition
- Is formed of the three clauses

```
SELECT <list_of_columns>
```

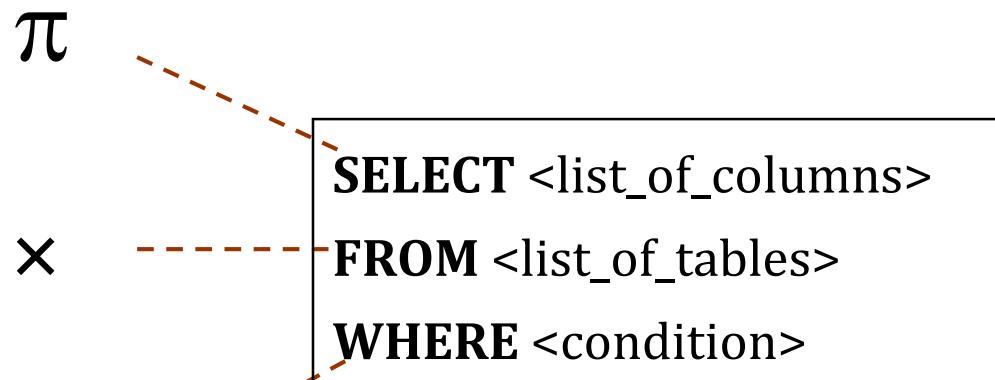
```
FROM <list_of_tables>
```

```
WHERE <condition>
```

- < list\_of\_columns >
  - Column names showed in the result of the query
- < list\_of\_tables >
  - Table names required to process the query
- < condition >
  - Boolean expression that identifies the rows to be retrieved
  - Expression's connection : AND, OR, and NOT
  - Operations: < , > , <=, >=, <>, =, LIKE and BETWEEN

# Basic query

- SQL and Relational Algebra



**SELECT L  
FROM R  
WHERE C**

$\dashrightarrow \pi_L(\sigma_C(R))$

# SELECT-clause

List all columns

*The entire tuple is produced*

```
SELECT *
  FROM KHOA
 WHERE PHONG='I53'
   AND NAMTL = '1995'
```

MaKhoa	TenKhoa	Phong	NamTL	DienThoai	TruongKhoa	NgayNhanChuc
CNTT	Công nghệ thông tin	I53	1995	08313964145	GV130	01/01/2007
SH	Sinh học	B32	1975	08313123545	GV250	01/01/1990

$\sigma_{\text{PHG}=\text{'I53'} \wedge \text{NamTL}=\text{'1995'}}(\text{KHOA})$

# SELECT-clause

List some specific columns

```
SELECT MAKHOA, TENKHOA, PHONG  
FROM KHOA  
WHERE PHONG='I53'  
AND NAMTL = '1995'
```

MaKhoa	TenKhoa	Phong
CNTT	Công nghệ thông tin	I53

$$\pi_{MAKHOA, TENKHOA, PHONG}(\sigma_{PHG='I53' \wedge NamTL='1995'}(KHOA))$$

# SELECT-clause

## Alias name

```
SELECT MAKHOA AS 'Mã khoa', TENKHOA AS 'Tên khoa', PHONG AS 'Mã phòng'  
FROM KHOA  
WHERE PHONG='I53' AND NAMTL = '1995'
```

Mã khoa	Tên khoa	Mã phòng
CNTT	Công nghệ thông tin	I53

$$\rho_{\text{Mã khoa, Tên khoa, Mã phòng}}(\pi_{\text{MANV, HONV, TENLOT, TENNV}}(\sigma_{\text{PHG='I53'} \wedge \text{NamTL='1995'}}(\text{KHOA})))$$

# SELECT-clause

## Extension

```

SELECT MAGV, HOTEN, SONHA + ',' + DUONG + ',' + ',' + QUAN + ',' +
THANHPHO AS 'DIA CHI'
FROM GIAOVIEN
WHERE PHAI='Nam'

```

MAGV	HOTEN	DIA CHI
GV001	Nguyễn Văn A	123 Phan Đăng Lưu, Q.Phú Nhuận, TP.Hồ Chí Minh

$$\rho_{MAGV, HOTEN, DIA\ CHI}(\pi_{MAGV, HOTEN, SONHA + DUONG + QUAN + THANHPHO}(\sigma_{PHAI = 'Nam'}(GIAOVIEN)))$$

# SELECT-clause

## Extension

```
SELECT MAGV, LUONG*1.1 AS 'LUONG10%'  
FROM GIAOVIEN  
WHERE PHAI='Nam'
```

MAGV	LUONG10%
GV001	550000

$$\rho_{MAGV, LUONG10\%}(\pi_{MAGV, LUONG*1.1}(\sigma_{PHAI = 'Nam'}(GIAOVIEN)))$$

# SELECT-clause

Duplicate tuples are eliminated

```
SELECT LUONG  
FROM GIAOVIEN  
WHERE PHAI= 'Nam'
```

LUONG
30000
25000
25000
38000



```
SELECT DISTINCT LUONG  
FROM GIAOVIEN  
WHERE PHAI= 'Nam'
```

LUONG
30000
25000
38000

- Cost
- Users want to see all tuples

# Example

- Find the MAGV and TENGV of teachers who work for the department ‘Hệ thống thông tin’

$R1 \leftarrow GIAOVIEN \bowtie GIAOVIEN.MABM=BOMON.MABM BOMON$

$KQ \leftarrow \pi_{MAGV, HOTEN} (\sigma_{TENBM='Hệ thống thông tin'} (R1))$

```

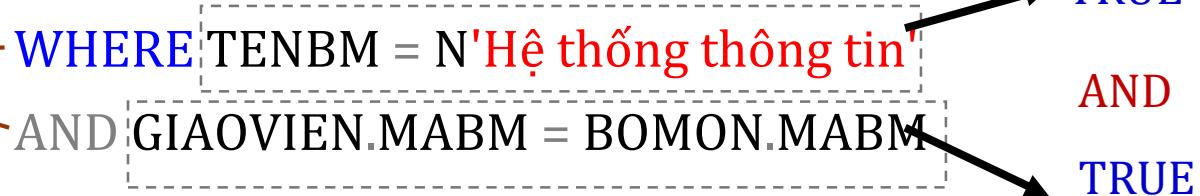
SELECT      MAGV, HOTEN
FROM        GIAOVIEN, BOMON
WHERE       TENBM= N'Hệ thống thông tin'
            AND GIAOVIEN.MABM=BOMON.MABM
    
```

# WHERE-clause

- Use logical operators (AND, OR) to combine two or more boolean expressions

Boolean  
expressions

```
SELECT MAGV, HOTEN  
FROM GIAOVIEN, BOMON  
WHERE TENBM = N'Hệ thống thông tin'  
AND GIAOVIEN.MABM = BOMON.MABM
```



TRUE  
AND  
TRUE

# WHERE-clause

## Priority

```
SELECT MAGV, HOTEN  
FROM GIAOVIEN, BOMON  
WHERE (TENBM = N'Hệ thống thông tin' OR TENBM = N'Mạng máy tính')  
AND GIAOVIEN.MABM = BOMON.MABM
```

The default priority of logical operators: left to right.

Use parentheses to explicitly specify the intended precedence of the operator

# WHERE-clause

**BETWEEN**

```
SELECT MAGV, HOTEN  
FROM GIAOVIEN  
WHERE LUONG >= 20000 AND LUONG <= 30000
```

```
SELECT MAGV, HOTEN  
FROM GIAOVIEN  
WHERE LUONG BETWEEN 20000 AND 30000
```

# WHERE-clause

**NOT BETWEEN**

```
SELECT MAGV, HOTEN  
FROM GIAOVIEN  
WHERE LUONG NOT BETWEEN 20000 AND  
30000
```



```
SELECT MAGV, HOTEN  
FROM GIAOVIEN  
WHERE LUONG < 20000 OR LUONG > 30000
```

# WHERE-clause

**LIKE**

```
SELECT MAGV, HOTEN
```

```
FROM GIAOVIEN
```

```
WHERE HOTEN LIKE 'Nguyen ___'
```



HOTEN LIKE 'nguyen \_\_\_'

Arbitrary characters

```
SELECT MAGV, HOTEN
```

```
FROM GIAOVIEN
```

```
WHERE HOTEN LIKE 'Nguyen %'
```

Arbitrary strings

# WHERE-clause

**NOT LIKE**

```
SELECT MAGV, HOTEN  
FROM GIAOVIEN  
WHERE HOTEN LIKE 'Nguyen'
```

↓  
negative

```
SELECT MAGV, HOTEN  
FROM GIAOVIEN  
WHERE HOTEN NOT LIKE 'Nguyen'
```

# WHERE-clause

**charlist**

MAGV	HOTEN
GV001	Nguyễn Văn A
GV002	Hoàng Thị C

```
SELECT *
FROM GIAOVIEN GV
WHERE GV.HOTEN LIKE N'[n]%'
```



MAGV	HOTEN
GV001	Nguyễn Văn A

```
SELECT *
FROM GIAOVIEN GV
WHERE GV.HOTEN LIKE N'[nh]%'
```



MAGV	HOTEN
GV001	Nguyễn Văn A
GV002	Hoàng Thị C

# WHERE-clause

**charlist**

MAGV	HOTEN
GV001	Nguyễn Văn A
GV002	Hoàng Thị C

```
SELECT *
FROM GIAOVIEN GV
WHERE GV.HOTEN LIKE N'[^h]%'
```



MAGV	HOTEN
GV001	Nguyễn Văn A

```
SELECT *
FROM GIAOVIEN GV
WHERE GV.HOTEN LIKE N'[^nh]%'
```



MAGV	HOTEN

# WHERE-clause

Ngày giờ

```
SELECT MAGV, HOTEN  
FROM GIAOVIEN  
WHERE NGAYSINH BETWEEN '1955-12-08' AND '1966-07-19'
```

'1955-12-08' YYYY-MM-DD

'17:30:00' HH:MI:SS

'12/08/1955' MM/DD/YYYY

'05:30 PM'

'December 8, 1955'

'1955-12-08 17:30:00'

# WHERE-clause

## NULL

- SQL allows attributes to have value NULL
  - Value unknown
  - Value inapplicable
  - Value withheld
- Operation on a NULL and any value, the result is NULL
  - $x$  has a value NULL
  - $x + 3$  is also NULL
- Comparison on a NULL value and any value, the result is UNKNOWN
  - The value of  $x = 3$  is UNKNOWN
  - The comparison  $x = 3$  is not correct SQL

# WHERE-clause

**NULL**

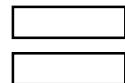
```
SELECT MAGV, HOTEN  
FROM GIAOVIEN  
WHERE GVQL IS NULL
```

```
SELECT MAGV, HOTEN  
FROM GIAOVIEN  
WHERE GVQL IS NOT NULL
```

# FROM-clause

From clause ~ **x** in Relational algebra

GIAOVIEN x BOMON



SELECT \*

FROM GIAOVIEN, BOMON

SELECT \*

FROM GIAOVIEN, BOMON

WHERE TRUE



Sử dụng thêm điều  
kiện ở WHERE để biểu  
diễn phép kết

GIAOVIEN  $\bowtie$  BOMON

**C**

SELECT \*

FROM GIAOVIEN, BOMON

WHERE **C**

MAGV	MAB	MABM	TENBM
001	MHTT	HTTT	Hệ thống thông tin
001	HTTT	MTT	Mạng máy tính
001	HTTT	CNPM	Công nghệ PM
002	MMT	HTTT	Hệ thống thông tin
002	MMT	MTT	Mạng máy tính
002	MMT	CNPM	Công nghệ PM
...	...	...	...

Alias name

# FROM-clause

Ambiguous ???

MAGV	MAB	MABM	TENBM
001	M HTTT	HTTT	Hệ thống thông tin
001	HTTT	MTT	Mạng máy tính
001	HTTT	CNPM	Công nghệ PM
002	MMT	HTTT	Hệ thống thông tin
002	MMT	MTT	Mạng máy tính
002	MMT	CNPM	Công nghệ PM
...	...	...	...

MAGV	MABM	TENBM
001	HTTT	Hệ thống thông tin
002	MTT	Mạng máy tính
...	...	...

SELECT MAGV, M~~A~~BM, TENBM  
 FROM GIAOVIEN, BOMON  
 WHERE MABM = ~~MABM~~

SELECT G.MAGV, G.MABM, B.TENBM  
 FROM GIAOVIEN **G**, BOMON **AS B**  
 WHERE G.MABM = B.MABM

use spaces or "AS" when you are aliasing

Điều kiện kết để tìm ra bộ môn của giáo viên

# Example

- Cho biết tên của bộ môn và tên của trưởng bộ môn của những bộ môn thuộc khoa CNTT (mã khoa)

```
SELECT BM.TENBM, GV.HOTEN AS TEN_TRUONGBM  
FROM BOMON BM, GIAOVIEN GV  
WHERE BM.TRUONGBM = GV.MAGV AND  
BM.MAKHOA= 'CNTT'
```

Điều kiện kết để tìm ra giáo viên làm trưởng bộ môn

# Example

- Với những đề tài thuộc cấp quản lý ‘Thành phố’, cho biết mã đề tài, đề tài thuộc về chủ đề nào, họ tên người chủ nghiêm đề tài cùng với ngày sinh và địa chỉ của người ấy

```
SELECT D.MADT, C.TENCD, G.MAGV, G.HOTEN, G.DIACHI  
FROM DETAI D, CHUDE C, GIAOVIEN G  
WHERE D.CapQL = 'Thanh Pho' AND D.MACD = C.MACD AND  
D.GVCNDT = G.MAGV
```

# Example

- Tìm họ tên của giáo viên viên thuộc bộ môn “HTTT” có tham gia vào đề tài “ Mobile Database” với số tiền phụ cấp cho mỗi công việc trên 10 triệu.

```
SELECT GV.HOTEN  
FROM GIAOVIEN GV, THAMGIADT TG, DETAI DT  
WHERE GV.MAGV = TG.MAGV AND  
TG.MADT = DT.MADT AND  
GV.MABM='HTTT' AND  
DT.TENDT='Mobile Database' AND TG.PHUCAP>10
```

# Example

- Tìm họ tên của từng giáo viên và người phụ trách chuyên môn trực tiếp của nhân viên đó.

# Example

- Tìm họ tên của những giáo viên được “Trần Trà Hương” phụ trách quản lý chuyên môn.

# ORDER BY-clause

- Is used for presenting a query in sorted order
- Syntax

```
SELECT <<List_of_columns>
FROM <List_of_tables>
WHERE <Conditions>
ORDER BY <List_of columns>
```

- ASC: ascending order(default)
- DESC:descending order

# ORDER BY-clause

- Example

```
SELECT *
FROM THAMGIADT
ORDER BY MAGV DESC, MADT ASC, STT DESC
```

MAGV	MADT	STT
GV01	DT01	1
GV01	DT01	2
GV01	DT02	1
GV02	DT01	2
GV02	DT01	3
GV02	DT03	1
GV02	DT03	4



MAGV	MADT	STT
GV02	DT01	3
GV02	DT01	2
GV02	DT03	4
GV02	DT03	1
GV01	DT01	2
GV01	DT01	1
GV01	DT02	1



# Content

- Introduction
- Data definition
- **Data manipulation**
  - Basic queries
  - **Set, set/multiset comparison and nested queries**
  - Aggregate functions and grouping
- View
- Index

# Phép toán tập hợp trong SQL

- SQL has implemented set operators
  - UNION
  - INTERSECT
  - EXCEPT
- The result is a set
  - Eliminate identical tuples
  - To keep identical tuples
    - UNION ALL
    - INTERSECT ALL
    - EXCEPT ALL

# Set operations in SQL

## ■ Syntax

```
SELECT <ColList> FROM <TabList> WHERE <Condition>
```

**UNION [ALL]**

```
SELECT <ColList> FROM <TabList> WHERE < Condition >
```

```
SELECT < ColList > FROM < TabList > WHERE < Condition >
```

**INTERSECT [ALL]**

```
SELECT < ColList > FROM < TabList > WHERE < Condition >
```

```
SELECT < ColList > FROM < TabList > WHERE < Condition >
```

**EXCEPT [ALL]**

```
SELECT < ColList > FROM < TabList > WHERE < Condition >
```

# Example

- Cho biết mã của các giáo viên có họ tên bắt đầu là '**Nguyễn**' và lương trên 200000 **hoặc**, giáo viên là trưởng bộ môn nhận chức sau năm 1995

```
SELECT MAGV  
FROM GIAOVIEN  
WHERE HOTEN LIKE N'Nguyễn%'  
AND LUONG > 200000  
UNION  
SELECT TRUONGBM  
FROM BOMON  
WHERE YEAR(NGAYNHANCHUC)>=1995
```

# Example

- Tìm những giáo viên vừa là trưởng bộ môn vừa chủ nhiệm đề tài

```
SELECT TRUONGBM
```

```
FROM BOMON
```

```
INTERSECT
```

```
SELECT GVCNDT
```

```
FROM DETAI
```

```
SELECT BM.TRUONGBM
```

```
FROM BOMON BM, DETAI DT
```

```
WHERE BM.TRUONGBM = DT.GVCNDT
```

# Example

- Tìm những giáo viên không tham gia bất kỳ đề tài nào

```
SELECT MAGV  
FROM GIAOVIEN  
EXCEPT  
SELECT MAGV  
FROM THAMGIADT
```

# Nested query

```
SELECT MAGV, HOTEN  
FROM GIAOVIEN, BOMON  
WHERE TENBM = N'Hệ thống thông tin'  
AND GIAOVIEN.MABM = BOMON.MABM
```

Outer query

```
SELECT <danh sách các cột>  
FROM <danh sách các bảng>  
WHERE <so sánh tập hợp> (  
    SELECT <danh sách các cột>  
    FROM <danh sách các bảng>  
    WHERE <điều kiện>)
```

Subquery

# Nested query

- Queries can have several nested levels
- Subqueries of a WHERE clause are connected by logical connective
  - OR, AND
- Subqueries will return
  - A single attribute and a single tuple (a single value)
  - A table (a set or multiset of tuples)

# Nested query

- WHERE clause of the outer query
  - <Expression> <set operation> <subquery>
  - Set comparison includes many operators
    - IN, NOT IN
    - ALL
    - ANY hoặc SOME
  - Check whether the result of subqueries is empty or not
    - EXISTS
    - NOT EXISTS

# Nested query

## ■ Categories

- Non-correlated subqueries
  - WHERE clause of subqueries do not refer to attributes of relations in FROM clause of the outer query
  - Subqueries will be performed before the outer query, and be executed just one time
- Correlated subqueries
  - WHERE clause of subqueries refer to at least one attribute of relations in FROM clause of the outer query
  - Subqueries will be executed many times, each time will correlate to one tuple of the outer query

# Example

- Tìm những giáo viên là trưởng bộ môn

```
SELECT MAGV, HOTEN
FROM GIAOVIEN, BOMON
WHERE GIAOVIEN.MAGV = BOMON.TRUONGBM
```

*Cartesian product*

```
SELECT MAGV, HOTEN
FROM GIAOVIEN
WHERE MAGV IN (SELECT TRUONGBM
                FROM BOMON)
```

```
SELECT MAGV, HOTEN
FROM GIAOVIEN
WHERE MABM IN ('001',
'002', '004', '005', '007')
```

*Nested query with IN*

Subquery **does not use** attributes from relations in the outer query

# Example

```
SELECT HOTEN  
FROM GIAOVIEN  
WHERE MAGV IN (SELECT MAGV  
                FROM GIAOVIEN  
                WHERE HOTEN LIKE N'Nguyễn%'  
                AND LUONG > 200000)  
  
OR MAGV IN (SELECT TRUONGBM  
              FROM BOMON  
              WHERE YEAR(NGAYNHANCHUC)>=1995)
```

# Example

- Tìm những giáo viên không tham gia đề tài nào

```
SELECT *
FROM GIAOVIEN
WHERE MAGV NOT IN(SELECT MAGV
                   FROM THAMGIADT)
```

```
SELECT *
FROM GIAOVIEN
WHERE MAGV <> ALL(SELECT MAGV
                     FROM THAMGIADT)
```

# Example

- Tìm những giáo viên có lương lớn hơn lương của ít nhất một giáo viên bộ môn ‘**Công nghệ phần mềm**’

```
SELECT *
FROM GIAOVIEN
WHERE LUONG > ANY (SELECT GV.LUONG
                     FROM GIAOVIEN GV, BOMON BM
                     WHERE GV.MABM = BM.MABM
                     AND BM.TENBM = N'Công nghệ phần mềm')
```

```
SELECT GV1.*
FROM GIAOVIEN GV1, GIAOVIEN GV2, BOMON BM
WHERE GV2.MABM = BM.MABM
AND BM.TENBM = N'Công nghệ phần mềm' AND GV1.LUONG > GV2.LUONG
```

# Example

- Tìm những giáo viên có lương lớn hơn lương của tất cả giáo viên thuộc bộ môn ‘**Hệ thống thông tin**’

```
SELECT *
FROM GIAOVIEN
WHERE LUONG > ALL (SELECT LUONG
                     FROM GIAOVIEN GV, BOMON BM
                     WHERE GV.MABM = BM.MABM
                     AND BM.TENBM = N'Hệ thống thông tin')
```

# Example

- Tìm những trưởng bộ môn tham gia tối thiểu 1 đề tài

```
SELECT *
FROM GIAOVIEN
WHERE MAGV IN (SELECT TRUONGBM
                 FROM BOMON)
AND MAGV IN (SELECT MAGV
              FROM THAMGIADT)
```

# Example

- Tìm những giáo viên là trưởng bộ môn

```
SELECT MAGV, HOTEN
```

```
FROM GIAOVIEN
```

*Nested query with IN*

```
WHERE MAGV IN (SELECT TRUONGBM FROM BOMON)
```

```
SELECT MAGV, HOTEN
```

```
FROM GIAOVIEN GV
```

```
WHERE EXISTS (SELECT *
```

*Nested query with EXISTS*

```
FROM BOMON BM
```

```
WHERE BM.TRUONGBM = GV.MAGV)
```

Giáo viên là trưởng bộ môn khi **tồn tại** một bộ môn có TRUONGBM = MAGV của giáo viên đó

Mệnh đề WHERE của truy vấn con tham chiếu ít nhất một thuộc tính của các quan hệ trong mệnh đề FROM ở truy vấn cha

# Example

- Tìm những giáo viên có lương lớn nhất

```
SELECT MAGV, HOTEN  
FROM GIAOVIEN GV  
WHERE NOT EXISTS (SELECT *  
                   FROM GIAOVIEN GV2  
                   WHERE GV2.LUONG > GV.LUONG)
```

Giáo viên là có lương lớn nhất khi **không tồn tại** một giáo viên nào mà có lương lớn hơn giáo viên đó

```
SELECT MAGV, HOTEN  
FROM GIAOVIEN GV  
WHERE LUONG >= ALL (SELECT LUONG  
                     FROM GIAOVIEN GV2)
```

Giáo viên là có lương lớn nhất khi lương của giáo viên **lớn hơn hoặc bằng tất cả** lương của các giáo viên (lồng phân cấp)

# Example

- Tìm giáo viên trùng tên và cùng giới tính với giáo viên khác trong cùng bộ môn

```
SELECT *
FROM GIAOVIEN GV1
WHERE EXISTS (SELECT *
               FROM GIAOVIEN GV2
               WHERE GV1.HOTEN LIKE GV2.HOTEN
                 AND GV1.PHAI = GV2.PHAI
                 AND GV1.MABM = GV2.MABM
                 AND GV1.MAGV <> GV2.MAGV)
```

# Example

- Tìm những giáo viên không tham gia đề tài nào

```
SELECT *
FROM GIAOVIEN GV
WHERE NOT EXISTS (SELECT *
                   FROM THAMGIADT PC
                   WHERE PC.MAGV = GV.MAGV)
```

Giáo viên **GV** không tham gia đề tài khi **không tồn tại** một dòng nào trong THAMGIADT mà có  $MAGV = GV.MAGV$

# Example

- Tìm những giáo viên có lương lớn hơn lương của ít nhất một giáo viên bộ môn ‘Công nghệ phần mềm’

```
SELECT *
FROM GIAOVIEN GV1
WHERE EXISTS (SELECT *
               FROM GIAOVIEN GV2, BOMON BM
               WHERE GV2.MABM = BM.MABM
               AND BM.TENBM = N'Công nghệ phần
mềm'
               AND GV1.LUONG > GV2.LUONG)
```

# Example

- Tìm những trưởng bộ môn tham gia tối thiểu 1 đề tài

```
SELECT *
```

```
FROM GIAOVIEN GV
```

```
WHERE EXISTS (SELECT *
```

```
    FROM BOMON BM
```

```
    WHERE GV.MAGV = BM.TRUONGBM)
```

```
AND EXISTS (SELECT * FROM THAMGIADT PC
```

```
    WHERE PC.MAGV = GV.MAGV)
```

# Discussion IN and EXISTS

## ■ IN

- <Column\_name> IN <Subquery>
- Attributes in the subquery's SELECT clause have the same data types as attributes in the outer query's WHERE clause

## ■ EXISTS

- Do not need attributes, constants or any expressions before it
- Do not need to specify column names in the subquery's SELECT clause
- Queries containing “= ANY” or IN can be converted queries containing EXISTS

# Discussion IN and EXISTS

## ■ EXISTS:

- Is used for checking an existing row
- Syntax
- ... EXISTS (Select ... From ... Where...)

*Sub query*

If the subquery returns at least one row => EXISTS (...) = TRUE

If the sub query returns no row => EXISTS (...) = FALSE

# Example

- Tìm những giáo viên có tham gia đề tài

```
SELECT *
FROM GIAOVIEN GV
WHERE EXISTS (SELECT *
                  FROM THAMGIADT PC
                  WHERE PC.MAGV = GV.MAGV)
```

Nếu giáo viên **GV** có tham gia đề tài →  
câu truy vấn bên trong sẽ có dữ liệu  
( $\geq 1$  dòng) → mệnh EXISTS (S .. F...  
W) có giá trị **TRUE**

# Example

- Tìm những giáo viên không tham gia đề tài

```
SELECT *
```

```
FROM GIAOVIEN GV
```

```
WHERE NOT EXISTS (SELECT *
```

```
    FROM THAMGIADT PC
```

```
    WHERE PC.MAGV = GV.MAGV)
```

Nếu giáo viên **GV** không tham gia đề tài → câu truy vấn bên trong sẽ rỗng (0 dòng) → NOT EXISTS (S...F..W) có giá trị TRUE

# Divide operation in SQL

R	A	B	C	D	E
$\alpha$	a	$\alpha$	a	a	1
$\alpha$	a	$\gamma$	a	a	1
$\alpha$	a	$\gamma$	b	b	1
$\beta$	a	$\gamma$	a	a	1
$\beta$	a	$\gamma$	b	b	3
$\gamma$	a	$\gamma$	a	a	1
$\gamma$	a	$\gamma$	b	b	1
$\gamma$	a	$\beta$	b	b	1

S	D	E
$b_i$	a	1
	b	1

$R \div S$	A	B	C
$a_i$	$\alpha$	a	$\gamma$
	$\gamma$	a	$\gamma$

- $R \div S$  is a set of values  $a_i$  in  $R$  such that there is no values  $b_i$  in  $S$  that makes the tuple  $(a_i, b_i)$  does not exist in  $R$

# Divide operation in SQL

## ■ Using EXCEPT

```
SELECT R1.A, R1.B, R1.C
```

```
FROM R R1
```

```
WHERE NOT EXISTS (
```

```
    ( SELECT S.D, S.E FROM S)
```

```
    EXCEPT
```

```
    ( SELECT R2.D, R2.E
```

```
        FROM R R2
```

```
        WHERE R1.A=R2.A AND R1.B=R2.B
```

```
        AND R1.C=R2.C )
```

```
)
```

# Divide operation in SQL

- Using NOT EXISTS

```
SELECT R1.A, R1.B, R1.C
```

```
FROM R R1
```

```
WHERE NOT EXISTS (
```

```
    SELECT *
```

```
    FROM S
```

```
    WHERE NOT EXISTS (
```

```
        SELECT *
```

```
        FROM R R2
```

```
        WHERE R2.D=S.D AND R2.E=S.E
```

```
        AND R1.A=R2.A AND R1.B=R2.B AND R1.C=R2.C ))
```

# Example

- Tìm tên các giáo viên được phân công làm tất cả các đề tài
  - Tìm tên các **nhân viên** mà không có **đề án** nào là không được **phân công làm**
  - Tập bị chia: THAMGIADT(MAGV, MADT)
  - Tập chia: DETAI(MADT)
  - Tập kết quả: KQ(MAGV)
  - Kết KQ với GIAOVIEN để lấy ra TENGV

# Example

```
SELECT DISTINCT GV.MAGV, GV.HOTEN
FROM GIAOVIEN GV, THAMGIADT PC1
WHERE GV.MAGV = PC1.MAGV
```

AND NOT EXISTS

( (SELECT DT.MADT FROM DETAI DT) *Toàn bộ đề tài*  
 EXCEPT  
 (SELECT PC2.MADT  
 FROM THAMGIADT PC2  
 WHERE PC2.MAGV = **PC1.MAGV**) )

Những đề tài mà giáo viên PC1.MAGV không tham gia

*Những đề tài mà PC1.MaGV đã tham gia*

Nếu danh sáchs đề tài chưa tham gia = rỗng →  
 Giáo viên đã tham gia tất cả các đề tài

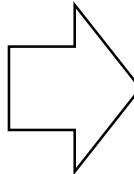
# Example

```
SELECT DISTINCT GV.MAGV, GV.HOTEN  
FROM GIAOVIEN GV, THAMGIADT PC1  
WHERE GV.MAGV = PC1.MAGV
```

AND NOT EXISTS (

```
    SELECT *  
    FROM DETAI DT  
    WHERE NOT EXISTS (SELECT *  
        FROM THAMGIADT PC2  
        WHERE PC2.MAGV = PC1.MAGV  
        AND DT.MADT = PC2.MADT))
```

Những đề tài  
mà giáo viên  
PC1.MAGV  
không tham gia



Tìm những giáo viên mà danh sách đề tài không tham gia = rỗng

# Example

```
SELECT DISTINCT GV.MAGV, GV.HOTEN  
FROM GIAOVIEN GV, THAMGIADT PC1  
WHERE GV.MAGV = PC1.MAGV  
AND 0 = (SELECT COUNT(*)  
        FROM DETAI DT  
        WHERE NOT EXISTS (SELECT *  
                           FROM THAMGIADT PC2  
                           WHERE PC2.MAGV = PC1.MAGV  
                           AND DT.MADT = PC2.MADT))
```

Tìm những giáo viên mà số lượng đề tài chưa tham gia bằng 0

# Divide operation in SQL

- Cho  $R(\underline{A}, \underline{B})$ ,  $S(\underline{B})$ , thực hiện  $R \div S$

```
SELECT R.A  
FROM R  
[WHERE R.B IN (SELECT S.B FROM S [WHERE <ĐK>])]  
GROUP BY R.A  
HAVING COUNT(DISTINCT R.B) = ( SELECT COUNT(S.B)  
                                FROM S  
                                [WHERE <ĐK>])
```

# Example

```
SELECT DISTINCT GV.MAGV, GV.HOTEN  
FROM GIAOVIEN GV, THAMGIADT PC1  
WHERE GV.MAGV = PC1.MAGV  
GROUP BY GV.MAGV, GV.HOTEN  
HAVING COUNT(DISTINCT PC1.MADT) = (SELECT COUNT(MADT)  
                                    FROM DETAI DT)
```

# Example

- Tìm tên các giáo viên được phân công làm **tất cả** các đề tài có kinh phí trên 100 triệu?

```
SELECT DISTINCT GV.MAGV, GV.HOTEN  
FROM GIAOVIEN GV, THAMGIADT PC1  
WHERE GV.MAGV = PC1.MAGV  
AND PC1.MADT IN (SELECT DT.MADT FROM DETAI WHERE KINHPHI > 100)  
GROUP BY GV.MAGV, GV.HOTEN  
HAVING COUNT(DISTINCT PC1.MADT) = (SELECT COUNT(MADT)  
                                    FROM DETAI DT  
                                    WHERE KINHPHI > 100)
```

# Exercise

1. Cho biết mã số, họ tên, ngày sinh của giáo viên tham gia tất cả các công việc của đề tài ‘Ứng dụng hóa học xanh’.
2. Cho biết mã số, họ tên, tên bộ môn và tên người quản lý chuyên môn của giáo viên tham gia tất cả các đề tài thuộc chủ đề ‘Nghiên cứu phát triển’.
3. Cho biết họ tên, ngày sinh, tên khoa, tên trưởng khoa của giáo viên tham gia tất cả các đề tài có giáo viên ‘Nguyễn Hoài An’ tham gia.
4. Cho biết họ tên giáo viên khoa ‘Công nghệ thông tin’ tham gia tất cả các công việc của đề tài có trưởng bộ môn của bộ môn đông nhất khoa ‘Công nghệ thông tin’ làm chủ nhiệm.

# Content

- Introduction
- Data definition
- **Data manipulation**
  - Basic queries
  - Set, set/multiset comparison and nested queries
  - **Aggregate functions and grouping**
  - Other
- View
- Index

# Aggregate functions

## ■ COUNT

- COUNT(\*) : the number of rows
- COUNT(<Column\_name>): the number of non-zero values of the column
- COUNT(DISTINCT <Column\_name>): the number of different and non-zero values of the column

## ■ MAX

## ■ SUM

## ■ AVG

## ■ These function is in SELECT and HAVING clause

# Example

- Tìm tổng lương, lương cao nhất, lương thấp nhất và lương trung bình của các giáo viên

```
SELECT SUM(LUONG), MAX(LUONG), MIN(LUONG), AVG(LUONG)  
FROM GIAOVIEN
```

# Example

- Cho biết số lượng giáo viên của bộ môn ‘Mạng máy tính’

```
SELECT COUNT(*) AS SL_GV  
FROM GIAOVIEN GV, BOMON BM  
WHERE GV.MABM = BM.MABM  
AND TENBM=N'Mạng máy tính'
```

# Example

- Tìm những giáo viên có lương thuộc 3 mức lương cao nhất

```
SELECT *
FROM GIAOVIEN GV1
WHERE 2 >= (SELECT COUNT(*)
              FROM GIAOVIEN GV2
              WHERE GV2.LUONG > GV1.LUONG)
```

# Example

- Cho biết số lượng giáo viên của từng bộ môn

Bộ môn	Số lượng
HTTT	2
CNPM	1
MMT	1

MANV	HOTEN	...	MABM
GV001	Nguyễn Văn A	...	HTTT
GV002	Trần Văn B	...	HTTT
GV003	Trần Thị C	...	CNPM
GV004	Đặng Thị D	...	MMT

# Grouping

## ■ Syntax

```
SELECT <ColumnList>  
FROM <TableList>  
WHERE <Conditions>  
GROUP BY <List_of_grouping_columns>
```

## ■ After grouping

- Each group will have identical values at grouping attributes

# Example

- Cho biết số lượng giáo viên của từng bộ môn

```
SELECT MABM, COUNT(*) 'Số lượng giáo viên'  
FROM GIAOVIEN  
GROUP BY MABM
```

```
SELECT GV.MABM, COUNT(*) 'Số lượng giáo viên'  
FROM GIAOVIEN GV, BOMON BM  
WHERE GV.MABM = BM.MABM  
GROUP BY GV.MABM
```

# Example

- Với mỗi giáo viên cho biết mã số, mã đề tài và số công việc mà họ tham gia ứng với mỗi đề tài

MAGV	MADT	STT
GV001	DT001	1
GV001	DT001	2
GV001	DT002	1
GV002	DT002	2
GV003	DT001	3
GV003	DT002	3

```

SELECT PC.MAGV, PC.MADT, COUNT(*) AS 'Số lượng công việc'
FROM THAMGIADT PC
GROUP BY PC.MAGV, PC.MADT
    
```

# Example

- Cho biết những giáo viên tham gia từ 2 công việc trở lên cho mỗi đề tài?

MAGV	MADT	STT
GV001	DT001	1
GV001	DT001	2
GV001	DT002	1
GV002	DT002	2
GV003	DT001	3
GV003	DT002	3

# Conditions on groups

## ■ Syntax

```
SELECT <ColumnList>  
FROM <TableList>  
WHERE <Conditions>  
GROUP BY <List_of_grouping_columns>  
HAVING <Conditions>
```

# Example

- Cho biết những giáo viên tham gia từ 2 công việc trở lên cho mỗi đề tài?

```
SELECT PC.MAGV, PC.MADT, COUNT(*) AS 'Số lượng công việc'  
FROM THAMGIADT PC  
GROUP BY PC.MAGV, PC.MADT  
HAVING COUNT(*) >= 2
```

# Example

- Cho biết những giáo viên tham gia từ 2 đề tài trở lên

MAGV	MADT	STT
GV001	DT001	1
GV001	DT001	2
GV001	DT002	1
GV002	DT002	2
GV003	DT001	3
GV003	DT002	3

```

SELECT PC.MAGV, COUNT(DISTINCT MADT) AS 'Số lượng đề tài'
FROM THAMGIADT PC
GROUP BY PC.MAGV
HAVING COUNT(DISTINCT MADT) >= 2

```

# Example

- Cho biết những bộ môn (TENBM) có lương trung bình của các giáo viên lớn hơn 20000

```
SELECT GV.MABM, AVG(GV.LUONG) AS 'Lương trung  
bình'
```

```
FROM GIAOVIEN GV
```

```
GROUP BY GV.MABM
```

```
HAVING AVG(GV.LUONG)>20000
```

```
SELECT BM.TENBM, AVG(GV.LUONG) AS 'Lương trung bình'
```

```
FROM GIAOVIEN GV, BOMON BM
```

```
WHERE GV.MABM = BM.MABM
```

```
GROUP BY BM.MABM, BM.TENBM
```

```
HAVING AVG(GV.LUONG)>=20000
```

# Discussion

- GROUP BY -clause
  - Attributes in SELECT clause (excepting attributes of aggregate functions) must appear in GROUP BY-clause
- HAVING-clause
  - Use aggregate functions in SELECT-clause to check a certain condition
  - Just validate the conditions for groups, not a condition for filtering rows
  - After grouping, conditions on groups will be performed

# Discussion

- The order of the query execution
  - (1) Pick out rows that satisfy conditions in the WHERE clause
  - (2) Group these rows into many groups in GROUP BY-clause
  - (3) Apply aggregate functions for each group
  - (4) Eliminate groups that do not satisfy conditions in the HAVING-clause
  - (5) Retrieve values from columns and aggregate functions in SELECT clause

# Example

- Tìm những phòng ban có lương trung bình cao nhất

```
SELECT GV.MABM, AVG(GV.LUONG) AS 'Lương trung bình'  
FROM GIAOVIEN GV  
GROUP BY GV.MABM  
HAVING AVG(GV.LUONG) = (SELECT MAX(AVG(GV.LUONG))  
                           FROM GIAOVIEN GV  
                           GROUP BY GV.MABM)
```

# Example

- Tìm những phòng ban có lương trung bình cao nhất

```
SELECT GV.MABM, AVG(GV.LUONG) AS 'Lương trung bình'  
FROM GIAOVIEN GV  
GROUP BY GV.MABM  
HAVING AVG(GV.LUONG)>= ALL(SELECT AVG(GV.LUONG)  
                                FROM GIAOVIEN GV  
                                GROUP BY GV.MABM)
```

# Example

- Tìm tên các giáo viên được phân công làm tất cả các đề tài

```
SELECT PC.MAGV, COUNT(DISTINCT PC.MADT) AS 'Số lượng đề tài'  
FROM THAMGIADT PC  
GROUP BY PC.MAGV  
HAVING COUNT(DISTINCT PC.MADT) = (SELECT COUNT(MADT)  
                                    FROM DETAI)
```

# Content

- Introduction
- Data definition
- **Data manipulation**
  - Basic queries
  - Set, set/multiset comparison and nested queries
  - Aggregate functions and grouping
  - **Others**
- View
- Index

# Other queries

- Subquery in FROM clause
- Joining conditions in FROM clause
  - Natural join
  - Outer join
- CASE structure

# Subquery in FROM clause

- The result of a subquery is a table
  - Intermediate table in the process of query execution
  - Do not store this result into the database
- Syntax

```
SELECT <ColumnList>
FROM R1, R2, (<Subquery>) AS Table_name
WHERE <Conditions>
```

# Example

- Cho biết những bộ môn (TENBM) có lương trung bình của các giáo viên lớn hơn 20000

```
SELECT BM.TENBM, AVG(GV.LUONG) AS LUONG_TB  
FROM GIAOVIEN GV, BOMON BM  
WHERE GV.MABM = BM.MABM  
GROUP BY BM.MABM, BM.TENBM  
HAVING AVG(GV.LUONG)>=20000
```

# Example

- Cho biết những bộ môn (TENBM) có lương trung bình của các giáo viên lớn hơn 20000

```
SELECT BM.TENBM, LUONG_GV.LUONG_TB  
FROM BOMON BM, (SELECT MABM, AVG(LUONG) LUONG_TB  
                  FROM GIAOVIEN  
                  GROUP BY MABM) AS LUONG_GV  
WHERE BM.MABM = LUONG_GV.MABM
```

# Join conditions in FROM clause

- Equijoin

```
SELECT <ColumnList>  
FROM R1 [INNER] JOIN R2 ON <Expression>  
WHERE <Conditions>
```

- Outer join

```
SELECT <ColumnList>  
FROM R1 LEFT|RIGHT [OUTER] JOIN R2 ON <Expression>  
WHERE <Conditions>
```

# Example

- Tìm mã và tên các giáo viên làm việc tại bộ môn ‘Hệ thống thông tin’

```
SELECT MAGV, HOTEN  
FROM GIAOVIEN, BOMON  
WHERE TENBM = N'Hệ thống thông tin'  
AND GIAOVIEN.MABM = BOMON.MABM
```

```
SELECT MAGV, HOTEN  
FROM GIAOVIEN GV INNER JOIN BOMON BM ON GV.MABM = BM.MABM  
WHERE TENBM = N'Hệ thống thông tin'
```

# Example

- Tìm họ tên các giáo viên và tên các đề tài giáo viên tham gia nếu có

```
SELECT DISTINCT GV.*
```

```
FROM GIAOVIEN GV LEFT JOIN (THAMGIADT PC JOIN DETAI DT ON  
PC.MADT = DT.MADT) ON GV.MAGV = PC.MAGV
```

GIAOVIEN

GIAOVIEN JOIN THAMGIADT

GV.MAGV= PC.MAGV

extend



# CASE structure

- Allow us to check conditions or output the information in each case
- Syntax

```
CASE <Column_name>
    WHEN <value> THEN <Expression>
    WHEN < value > THEN <Expression>
    ...
    [ELSE < Expression >]
END
```

# Example

- Cho biết họ tên các giáo viên đã đến tuổi về hưu (nam 60 tuổi, nữ 55 tuổi)

```
SELECT HOTEN  
FROM GIAOVIEN  
WHERE YEAR(GETDATE()) - YEAR(NGAYSINH) >= ( CASE PHAI  
                                              WHEN 'Nam' THEN 60  
                                              WHEN 'Nu' THEN 55  
                                              END)
```

# Example

- Cho biết họ tên các giáo viên và năm về hưu

```
SELECT GV.HOTEN, YEAR(GV.NGAYSINH) + ( CASE PHAI  
                                              WHEN 'Nam' THEN 60  
                                              WHEN 'Nu' THEN 55  
                                          END) AS NAMVEHUU  
FROM GIAOVIEN GV
```

```
SELECT GV.HOTEN, ( CASE PHAI  
                                              WHEN 'Nam' THEN YEAR(NGAYSINH) + 60  
                                              WHEN 'Nu' THEN YEAR(NGAYSINH) + 55  
                                          END) AS NAMVEHUU  
FROM GIAOVIEN GV
```

# Summary

```
SELECT <List_of_columns>
FROM <List_of_tables>
[WHERE <Conditions>]
[GROUP BY <List_grouping_columns>]
[HAVING <Conditions>]
[ORDER BY <List_of_ordering_columns>]
```

# Content

- Introduction
- Data definition
- Data manipulation
- **Data update**
- View
- Index

# INSERT command

- Is used to add 1 or more tuple(s) to a relation
- In order to add a tuple
  - Relation name
  - List of column names
  - List of values for the tuple

# INSERT command

- Syntax (one tuple)

```
INSERT INTO <Table_name>(<List_of_columns>)  
VALUES (<List_of_values>)
```

# Example

```
INSERT INTO THAMGIADT(MAGV, MADT, STT, PHUCAP, KETQUA)
VALUES('002', '001', 1, 1.0, N'DAT')
```

```
INSERT INTO THAMGIADT(MAGV, MADT, STT, PHUCAP, KETQUA)
VALUES('002', '001', 2, 1.2, NULL)
```

# INSERT command

## ■ Discussion

- The order of values is the same to the order of columns
- The NULL value can be used for non-primary-key attributes
- INSERT command will raise errors if the integrity constraint is violated
  - Primary key
  - Reference
  - NOT NULL constraint



# INSERT command

- Syntax (many tuples)

```
INSERT INTO <Table_name>(<List_of_columns>)  
    <Query>
```

# Example

```
CREATE TABLE THONGKE_BM (
    TENBM NVARCHAR(50),
    SL_GV INT,
    LUONG_TC INT
    PRIMARY KEY(TENBM)
)
```

```
INSERT INTO THONGKE_BM
SELECT BM.TENBM, COUNT(GV.MAGV), SUM(GV.LUONG)
FROM GIAOVIEN GV, BOMON BM
WHERE GV.MABM = BM.MABM
GROUP BY BM.MABM, BM.TENBM
```

# DELETE command

- Is used to remove tuples from a relation
- Syntax

```
DELETE FROM <Table_name>  
[WHERE <Conditions>]
```

# Example

```
DELETE FROM GIAOVIEN  
WHERE HOTEN LIKE N'Trần%'
```

```
DELETE FROM GIAOVIEN  
WHERE MAGV = 'GV001'
```

```
DELETE FROM GIAOVIEN
```

# Example

- Xóa đi những giáo viên ở bộ môn ‘Hệ thống thông tin’

```
DELETE FROM GIAOVIEN  
WHERE MABM IN (SELECT MABM  
                FROM BOMON  
                WHERE TENBM = N'Hệ thống thông tin')
```

```
DELETE FROM GIAOVIEN  
        FROM BOMON BM  
WHERE GIAOVIEN.MABM = BM.MABM  
AND BM.TENBM = N'Hệ thống thông tin'
```

# DELETE command

## ■ Discussion

- The number of removed tuples depends on the condition in WHERE clause
- A missing WHERE clause specifies that all tuples can be deleted
- DELETE command can cause the violation of reference constraints
  - Do not permit to remove
  - Remove tuples whose value is being referred
    - CASCADE
  - Set the NULL value to reference values

# DELETE command

MAGV	HOTEN	...	MABM
CV001	Nguyễn Văn A	...	HTTT
GV002	Trần Văn B	...	HTTT
GV003	Trần Thị C	...	CNPM
GV004	Đặng Thị D	...	MMT

MAGV	MADT	STT	PHUCAP	KETQUA
CV001	001	1	...	...
CV001	001	3	...	...
GV003	002	1	...	
GV004	003	1	...	...
...	...	...	...	...

# DELETE command

MABM	TENBM
HTTT	Hệ thống thông tin
CNPM	Công nghệ phần mềm
MMT	Mạng máy tính
KHMT	Khoa học máy tính

MANV	HOTEN	...	MABM
GV001	Nguyễn Văn A	...	NULL
GV002	Trần Văn B	...	NULL
GV003	Trần Thị C	...	CNPM
GV004	Đặng Thị D	...	MMT

# UPDATE command

- Is used to change the value of attributes
- Syntax

```
UPDATE <Table_name>
SET   <Attribute_name>=<The_new_value>,
        <Attribute_name>=<The_new_value>,
        ...
[WHERE <Conditions>]
```

# Example

```
UPDATE GIAOVIEN  
SET NGAYSINH='08/12/1965'  
WHERE MAGV='GV001'
```

```
UPDATE GIAOVIEN  
SET LUONG=LUONG*1.1
```

# Example

- Với mỗi giáo viên của bộ môn ‘Hệ thống thông tin’, nâng lương của các giáo viên gấp 1.5 lần và gán giáo viên quản lý (GVQL) thành null

```
UPDATE GIAOVIEN  
SET LUONG = LUONG*1.5, GVQL = NULL  
WHERE MABM = (SELECT MABM  
                FROM BOMON  
                WHERE TENBM = N'Hệ thống thông tin')
```

# Example

- Tăng 10% lương cho giáo viên có tham gia đề tài

```
UPDATE GIAOVIEN  
SET LUONG = LUONG * 1.1  
FROM THAMGIADT TG  
WHERE TG.MAGV = GIAOVIEN.MAGV
```

```
UPDATE GIAOVIEN  
SET LUONG = LUONG * 1.1  
WHERE EXISTS (  
SELECT *  
FROM THAMGIADT TG  
WHERE TG.MAGV = GIAOVIEN.MAGV )
```

```
UPDATE GIAOVIEN  
SET LUONG = LUONG * 1.1  
WHERE MAGV IN (  
SELECT TG.MAGV  
FROM THAMGIADT TG  
WHERE TG.MAGV = GIAOVIEN.MAGV )
```

# Example

- Tăng lương 10% cho tất cả các giáo viên làm cho đề tài 'DT001' nhiều hơn 3 công việc.

```
UPDATE GIAOVIEN  
SET LUONG = LUONG * 1.1  
WHERE MAGV IN (SELECT PC.MAGV  
FROM THAMGIADT TG  
WHERE TG.MADT = 'DT001'  
GROUP BY TG.MAGV, TG.MADT  
HAVING COUNT(*) >= 3)
```

# UPDATE command

## ■ Discussion

- Tuples that satisfy conditions in WHERE clause will be modified to the new value
- A missing WHERE clause specifies that all tuples can be modified
- UPDATE command can cause violations of the reference constraint
  - Do not allow to modify
  - Modify the values of tuples that are being referred
    - CASCADE

# Content

- Introduction
- Data definition
- Data manipulation
- Data update
- **View**
- Index

# View

- Table is a relation that exist actually the database
  - Stored in some physical organization
  - Persistent
- View is also a relation
  - Do not exist physically (virtual table)
  - Do not contain the data
  - Is derived from other tables
  - Can query or even modify the data through views

# View

## ■ Why do we use views?

- Hide the complexity of data
- Simplify queries
- Present data with the purpose “easy to use”
- Mechanism of data safety

# View definition

## ■ Syntax

```
CREATE VIEW <View_name> AS  
    <Query>
```

```
DROP VIEW <View_name>
```

## ■ View contains

- A list of attributes that are the same as attributes in SELECT clause
- The number of tuples depending on the conditions in WHERE clause
- Data derived from tables in FROM clause

# Example

```
CREATE VIEW GV_HTTT AS  
    SELECT GV.*  
    FROM GIAOVIEN GV  
    WHERE BM.MABM = 'HTTT'
```

```
CREATE VIEW THONGKE_BM AS  
    SELECT BM.TENBM, COUNT(GV.MAGV) SL_GV,  
          SUM(GV.LUONG) TONG_LUONG  
    FROM GIAOVIEN GV, BOMON BM  
    WHERE GV.MABM = BM.MABM  
    GROUP BY BM.MABM, BM.TENBM
```

# Querying views

- Although views do not contain data, we can do the query on views

```
SELECT GV.HOTEN  
FROM GV_HTTT GV  
WHERE GV.MAGV = 'GV003'
```

$GV\_HTTT \leftarrow \sigma_{MABM='HTTT'}(GIAOVIEN)$

$\pi_{HOTEN}(\sigma_{MAGV='GV003'}(GV\_HTTT))$

# Querying views

- Can query data from both tables and views

```
SELECT DISTINCT GV.*  
FROM GV_HTTT GV, THAMGIADT PC  
WHERE GV.MAGV = PC.MAGV
```

$GV\_HTTT \leftarrow \sigma_{MABM='HTTT'}(GIAOVIEN)$

$KQ \leftarrow GV\_HTTT \bowtie_{GV\_HTTT.MAGV=THAMGIADT.MAGV} THAMGIADT$

# Modifying views

- Can apply INSERT, DELETE, and UPDATE commands to simple views
  - Views built on one table and having the key attribute of that table
- Cannot modify views
  - Views have a key word DISTINCT
  - Views use aggregate functions
  - Views have extended SELECT clause
  - Views are derived from table containing constraints on columns
  - Views are derived from many tables

# Modifying views

- Sửa lại lương cho giáo viên mã 'GV003' ở bộ môn 'Hệ thống thông tin' tăng lên 10%

```
UPDATE GV_HTTT
```

```
SET LUONG = LUONG * 1.1
```

```
WHERE MAGV = 'GV003'
```

# Content

- Introduction
- Data definition
- Data manipulation
- Data update
- View
- Index

# Index

- The index on an attribute A is the data structure that makes it efficient to find tuples having a fixed value for attribute A

```
SELECT *  
FROM GIAOVIEN  
WHERE MABM='HTTT' AND PHAI= 'Nu'
```

Đọc 10.000 bộ

Bảng GIAOVIEN có 10.000 bộ  
Có 200 giáo viên làm việc cho bộ môn ‘HTTT’

Đọc 200 bộ

Đọc 70 bộ

# Index

- Syntax

```
CREATE INDEX <Index_name> ON <Table_name>(<Column_name>)
```

```
DROP INDEX <Index_name>
```

- Example

```
CREATE INDEX MABM_IND ON GIAOVIEN(MABM)
```

```
CREATE INDEX MABM_PHAI_IND ON GIAOVIEN(MABM, PHAI)
```

## ■ Discussion

- Speed up queries in which a value for an attribute is specified, join operations
- Make insertion, deletion, and update more complex and time-consuming
- Cost
  - Index storage
  - Disk access (HDD)

## ■ Selection of indexes

- One of the principal factors that influence a database
- One of the hardest parts of database design

