



NGÔN NGỮ TRUY VẤN DỮ LIỆU SQL

CƠ SỞ DỮ LIỆU

Nội dung

2

1

Giới thiệu chung

2

Lệnh định nghĩa – khởi tạo

3

Truy vấn cơ bản

4

Sắp xếp – gom nhóm

5

Truy vấn lồng

Giới thiệu chung

3

- ❖ **RDBMS**(Relational database management system) là hệ thống quản lý các cơ sở dữ liệu quan hệ(relational database) như MS SQL, MySQL, Oracle.
- ❖ **Relational database**: là database cho phép liên kết dữ liệu để lưu trữ một lúc nhiều table, những liên kết thành lập mối quan hệ giữa những table cung cấp một cách thuận lợi nhất để lưu trữ dữ liệu, có thể nhập ở một nơi và tham chiếu đến nhiều table khác trong database



Giới thiệu chung

4

- **SQL** : **S**tructured **Q**uery **L**anguage ANSI (American National Standards Institute)
- Tiền thân SEQUEL và SEQUEL-2 và ISO (International Standards Organization)
- Do IBM phát triển (1974-1976).
- Các phiên bản
 - SQL - 86
 - SQL - 92
 - SQL - 99
- Hệ quản trị cơ sở dữ liệu
- Ngôn ngữ truy vấn dữ liệu

Giới thiệu chung

5

- SQL is a language that all commercial RDBMS implementations understand.
- SQL is a non-procedural language

in-memory

```
private static List<Dish> mock() {  
    // menu  
    Dish d1 = new Dish("D01", true, 200, Type.OTHER);  
    Dish d2 = new Dish("D02", false, 220, Type.MEAT);  
    Dish d3 = new Dish("D03", false, 280, Type.MEAT);  
    Dish d4 = new Dish("D04", false, 380, Type.FISH);  
    return Arrays.asList(d1,d2,d3,d4);  
}  
  
// Before (Java07)  
List<Dish> lowCaloricDishes = new ArrayList<>();  
for (Dish d: menu) {  
    if (d.getCalories() < 400) {  
        lowCaloricDishes.add(d);  
    }  
}  
  
// sorting by calories  
menu.sort(new Comparator<Dish>() {  
    @Override  
    public int compare(Dish d1, Dish d2) {  
        return d1.getCalories() - d2.getCalories();  
    }  
});  
  
// get name of dish  
List<String> lowCaloricDishesName = new ArrayList<>();  
for (Dish d: lowCaloricDishes) {  
    lowCaloricDishesName.add(d.getName());  
}
```

How to implement

database

Table: dishes

ID	NAME	CALORIES
1	D01	200
2	D02	220
3	D03	280
4	D04	380

```
SELECT name  
FROM dishes  
WHERE calories < 300  
ORDER BY CALORIES ASC
```

What you expected

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `insertIntoGroupItem`(p_rows INT)  
BEGIN  
  
    DECLARE i INT DEFAULT 1;  
    DECLARE max_group_item_id INT DEFAULT ((SELECT max(MaLoai) FROM LoaiHang));  
    DECLARE group_item_id INT;  
  
    WHILE i <= p_rows DO  
        SET group_item_id = max_group_item_id + i;  
        INSERT INTO LoaiHang(MaLoai, TenLoai)  
        VALUES(group_item_id, concat('Group Item ', group_item_id));  
        SET i = i + 1;  
    END WHILE;  
END
```

Giới thiệu chung

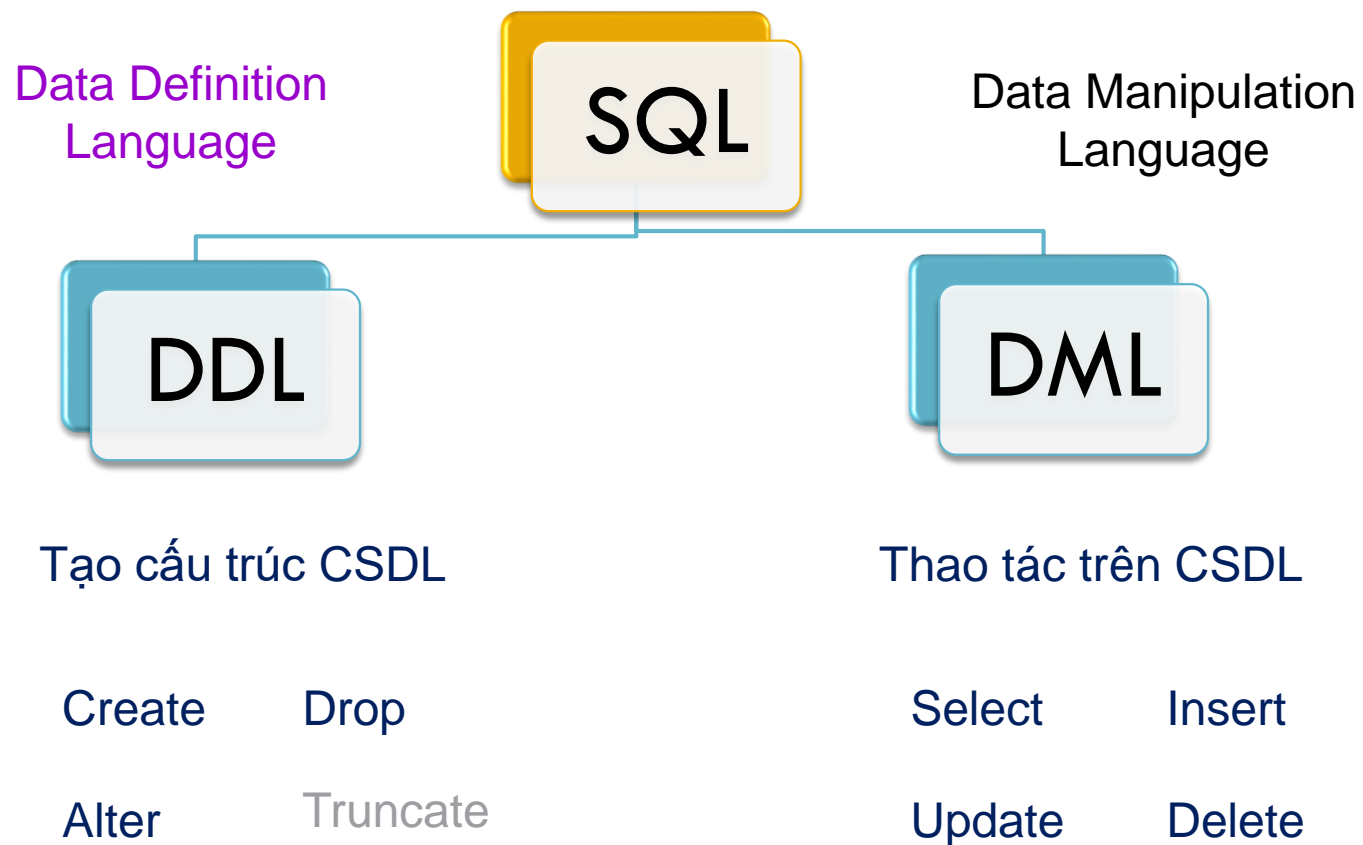
6

- Mỗi hệ quản trị CSDL đều phải có phải có ngôn ngữ giao tiếp giữa người sử dụng với cơ sở dữ liệu
- Ngôn ngữ giao tiếp CSDL gồm các loại sau:
 - Data Definition Language – **DDL**: Ngôn ngữ mô tả dữ liệu
 - Data Manipulation Language – **DML**: Ngôn ngữ thao tác dữ liệu



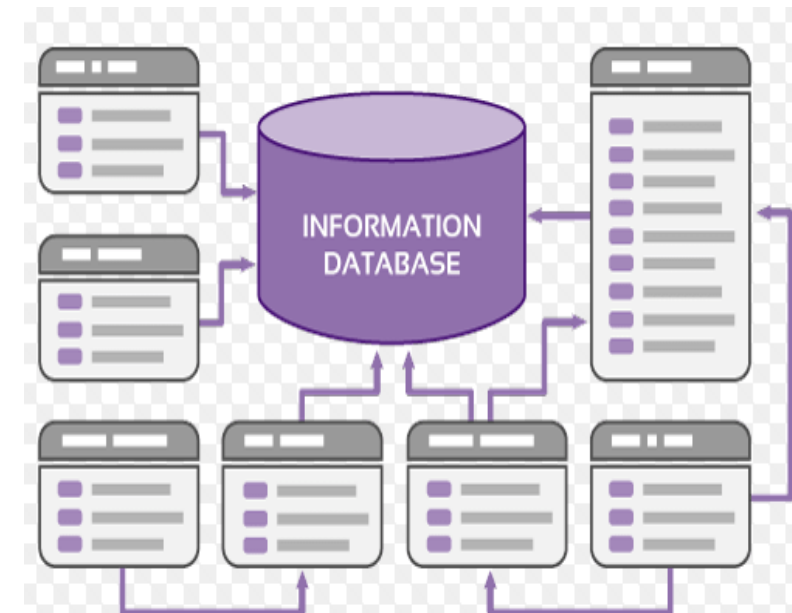
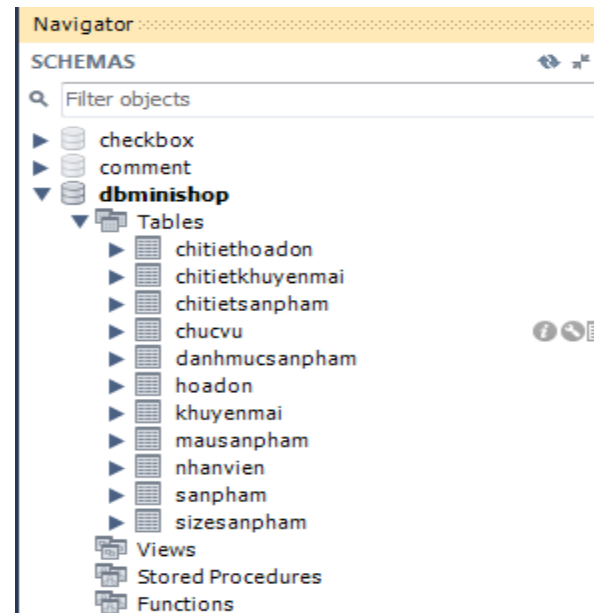
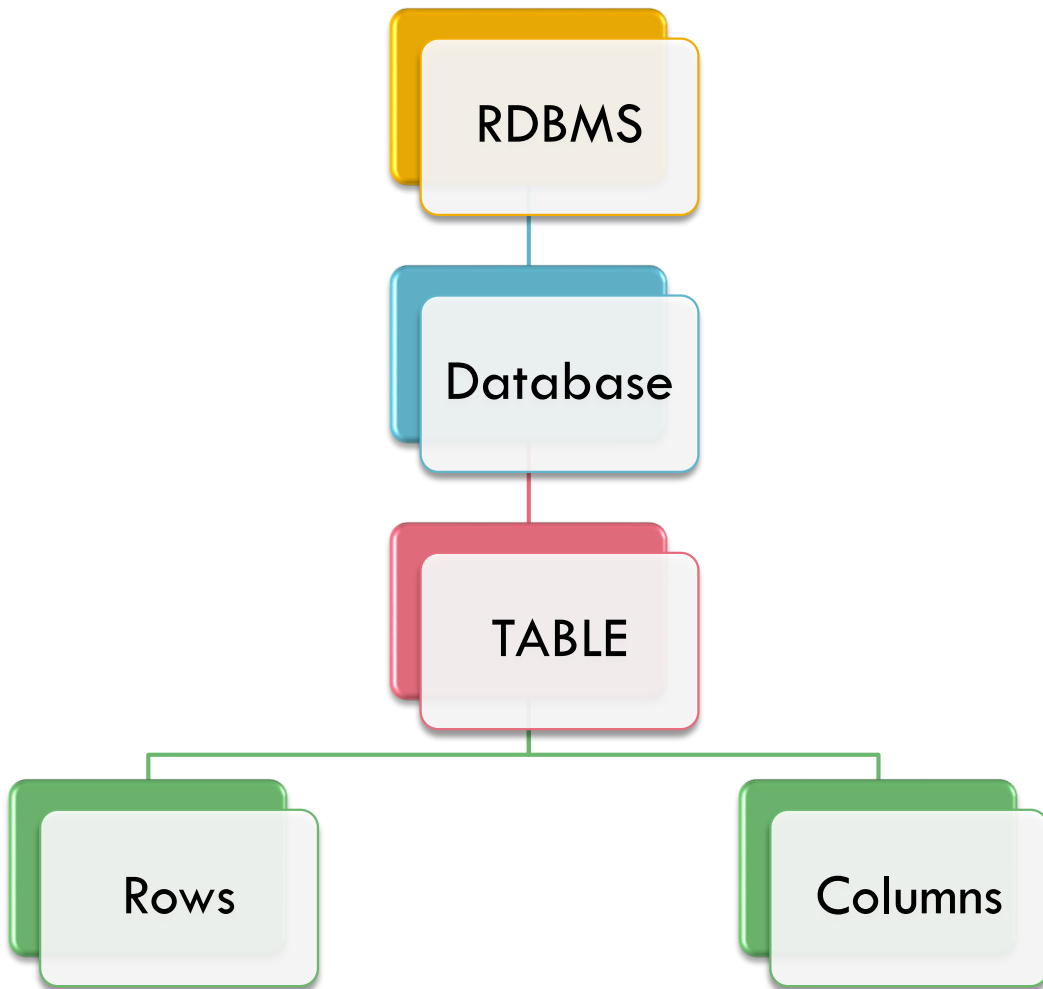
Giới thiệu chung

7



CẤU TRÚC CHUNG

8



Result Grid		Filter Rows:		Edit:	Export/Import:		Wrap Cell Content:	
	masanpham	madanhmuc	tensanpham	giatien	mota	hinhshanpham		
1	1		Sơ Mi Nam No Style TN O01	185,000	- Thiết kế áo sơ mi kiểu dáng basic, dễ dàng mix...	79fb5c06-b673-9300		
2	1		Sơ Mi Nam No Style TD R02	225,000	- Thiết kế áo sơ mi kiểu dáng basic, dễ dàng mix...	b443d8c7-54cc-4500		
3	1		Sơ Mi Nam No Style TN O02	185,000	- Thiết kế áo sơ mi kiểu dáng basic, dễ dàng mix...	7cb8e2db-a600-9c00		
4	1		Sơ Mi Nam No Style TN L01	225,000	- Thiết kế đơn giản và hiện đại với sơ mi tay ngắn...	4364f967-2c49-6600		
5	1		Sơ Mi Nam No Style TD ST01	225,000	- Thiết kế đơn giản và hiện đại với sơ mi tay ngắn...	d5ebdfa6-2945-2300		
6	1		Sơ Mi Nam No Style TN N03	225,000	- Thiết kế đơn giản và hiện đại với sơ mi tay ngắn...	71be5178-0fd9-3300		
7	7		Sơ Mi Adachi / 0012655	225,000	- Thiết kế đơn giản và hiện đại với sơ mi tay ngắn...	d5ebdfa6-2945-2300		
8	7		Sơ Mi Adachi / 0012738	225,000	- Thiết kế đơn giản và hiện đại với sơ mi tay ngắn...	a4f67968-f23e-0100		
9	7		Sơ Mi Adachi / 0012658	225,000	- Thiết kế đơn giản và hiện đại với sơ mi tay ngắn...	deec8c60-24e0-2400		

Kiểu dữ liệu

❖ Kiểu số

- Integer, smallint, **int**
- Numeric, decimal, real, **float**

❖ Boolean

- bit

❖ Kiểu chuỗi ký tự

- char (n), nchar (n)
- varchar(n), nvarchar (n)
- text

❖ Ngày giờ

- date: ngày, tháng, năm
- time: giờ, phút, giây
- datetime: date + time

❖ Bổ sung

- char: fixed length
- varchar: dynamic length
- n: national language

Phân biệt char và varchar

The CHAR and VARCHAR types are similar, but differ in the way they are stored and retrieved. They also differ in **maximum length** and in **whether trailing spaces are retained**.

Lệnh định nghĩa - DDL

11

Lệnh **CREATE** được dùng để:

- Tạo lược đồ (cơ sở dữ liệu)
- Tạo bảng
- Tạo khung nhìn
- Tạo ràng buộc
- Xóa dữ liệu

CREATE DATABASE | SCHEMA

CREATE TABLE

CREATE VIEW

DROP TABLE

ALTER TABLE

TRUNCATE TABLE

DDL

12

Lệnh tạo bảng

Bảng: tên bảng

tập thuộc tính

Tên

Kiểu dữ liệu

Ràng buộc toàn vẹn (RBTV)

```
CREATE TABLE TenBang
```

```
(
```

```
  <TenCot> <kieu du lieu> [RBTV],
```

```
  <TenCot> <kieu du lieu> [RBTV],
```

```
  [RBTV]
```

```
)
```

```
CREATE TABLE PhongBan
```

```
(
```

```
    MaPB    int primary key,
```

```
    TenPB   varchar(50) not null,
```

```
    NgayTao Date not null
```

```
)
```

DDL

13

```
class Input {  
  
    @Min(1)  
    @Max(10)  
    private int numberBetweenOneAndTen;  
  
    @Pattern(regex = "[0-9]{1,3}\\.[0-9]{1,3}\\.[0-9]{1,3}\\.[0-9]{1,3}$")  
    private String ipAddress;  
  
    // ...  
}
```

Tên

Kiểu dữ liệu

Ràng buộc toàn vẹn (RBTV)

CREATE TABLE PhongBan

(

MaPB int **primary key**,

TenPB varchar(50) **not null**,

NgayTao Date

)

Ràng buộc

Not null: Không cho phép thuộc tính chứa giá trị null

Null: Được chứa giá trị null

Khóa chính: Primary key

Khóa ngoại: Foreign key - references

Unique: Không trùng lặp

Default: Giá trị mặc định

Check: Kiểm tra điều kiện nào đó

Đặt tên ràng buộc: [**CONSTRAINT** <tên ràng buộc>] <RBTV>

DDL

15

```
-      CREATE TABLE NHANVIEN
      (
        MANV          CHAR(5) PRIMARY KEY,
        HONV          VARCHAR(30) NOT NULL,
        TENLOT        VARCHAR (30) NOT NULL,
        TENNV         VARCHAR(30) NOT NULL,
        PHAI          CHAR(10) CHECK PHAI IN ('Nam', 'Nu'),
        LUONG         INT DEFAULT (2000000),
        DIACHI        VARCHAR (100),
        NGAYSINH      DATETIME,
        MA_NQL        CHAR(5),
        PHG           CHAR(5)
        FOREIGN KEY (MA_NQL) REFERENCES NHANVIEN (MANV), PRIMARY KEY (MANV)
        FOREIGN KEY (PHG) REFERENCES PHONGBAN (MAPB) )
-      CREATE TABLE PHONGBAN (
        MAPB          CHAR(5) CONSTRAINT PK_PB PRIMARY KEY,
        TENPB         VARCHAR(30),
        TRPHG         CHAR(5),
        NGAYBD        DATETIME
        CONSTRAINT FK_PB FOREIGN KEY (TRPHG) REFERENCES NHANVIEN (MANV)
      )
```

DDL

16

ALTER TABLE : Thay đổi cấu trúc ràng buộc của bảng

❖ Thêm cột

ALTER TABLE <tên bảng> **ADD** <tên cột> <kiểu dữ liệu> [RBTv]

ALTER TABLE PhongBan ADD NgayTao Datetime null

❖ Xóa cột

ALTER TABLE <tên bảng> **DROP COLUMN** <tên cột>

ALTER TABLE PhongBan DROP NgayTao

❖ Thay đổi kiểu dữ liệu

ALTER TABLE <tên bảng> ALTER COLUMN <tên cột> <tên mới mới> <kiểu dữ liệu mới>

ALTER TABLE <tên bảng> **CHANGE** <tên cột> <tên mới mới> <KDL mới> <RBTv>

ALTER TABLE PhongBan CHANGE NgayTao NgayTao Date

DDL

17

Thay đổi cấu trúc ràng buộc của bảng

ALTER TABLE

❖ Thêm ràng buộc

```
ALTER TABLE <tên bảng> ADD  
CONSTRAINT <tên ràng buộc>  
<RBTV> (referenced_columns)  
[REFERENCES PARENT_TABLE  
(referenced_columns)],
```

❖ Xóa ràng buộc

```
ALTER TABLE <tên bảng>  
DROP <kiểu ràng buộc> <tên  
ràng buộc>
```

Lưu ý: Khi tạo khóa chính bên ngoài lệnh tạo bảng, thì các thuộc tính của khóa chính phải được khai báo là not null trong câu lệnh tạo.

Lệnh thao tác dữ liệu - DML - Insert

18

Thêm 1 dòng dữ liệu

```
INSERT INTO table_name (column1, column2, column3, ...)  
VALUES (value1, value2, value3, ...);
```

```
INSERT INTO customer  
VALUES(1, 20, "leonado")
```

```
INSERT INTO table_name  
VALUES (value1, value2, value3, ...);
```



Các giá trị theo đúng thứ tự
các cột trong table

DML - Insert

19

Thêm nhiều dòng dữ liệu

```
INSERT INTO table_name (column1, colname_2, .... )
```

```
VALUES ( values_1, values_2,...,values_n),
```

```
      ( values_1, values_2,...,values_n)
```

Note: Dòng cuối không có dấu phẩy “,”

```
INSERT INTO KhachHang (LoaiKH, Ho, Ten, HoTen, NgaySinh, Email) VALUES
(1, N'Phạm', N'Uyên', N'Phạm uyên', '01/25/1991', 'p.uyenh@gmail.com'),
(1, N'Nguyễn', N'Tùng', N'Nguyễn Tùng', '10/10/1992', 'n.tung@gmail.com')
(1, N'Trần', N'Hoàng', N'TranThanh', '09/10/1993', 't.hoang@gmail.com')
```

```
INSERT INTO table_name (column1, colname_2, .... )
```

```
VALUES ( values_1, values_2,...,values_n),
```

```
|
INSERT INTO KhachHang VALUES
(1, N'Lê', N'Thanh', N'Tran Thanh Thành', '10/10/1990', 'tranthanh@gmail.com', '01234560098', N'37 Hoàng Văn Thụ'),
(1, N'Trần', N'Tuấn', N'Trần Tuấn', '10/10/1990', 'tranthanh@gmail.com', '01234560098', N'37 Hoàng Văn Thụ'),
(1, N'Nguyễn', N'Lan', N'Nguyễn Lan', '10/10/1990', 'tranthanh@gmail.com', '01234560098', N'37 Hoàng Văn Thụ')
```

DML - Insert

20

Lời khuyên khi tạo database và nhập liệu

Nhập liệu

- + Nhập các bảng không có khóa ngoại trước
- + Nếu nhập bảng có khóa ngoại, cho giá trị bằng NULL
- + Nhập các bảng có khóa ngoại và dữ liệu đúng với dữ liệu ở referenced table
- + **DIS/EN-ABLE FOREIGN_KEY_CHECKS**

DML - DELETE

21

1. Xóa toàn bộ dữ liệu

```
DELETE FROM TABLE
```

```
DELETE FROM KháchHang
```

2. Xóa theo điều kiện

```
DELETE FROM table_name  
WHERE <condition>
```

```
DELETE FROM KháchHang  
WHERE LoạiKH = 1
```

Foreign Key Option

Foreign Key Options

On Update: NO ACTION

On Delete: NO ACTION

There are three types of on delete associated with foreign key

- ❖ **On Delete Cascade:** when data is removed from a parent table, automatically data deleted from child table (foreign key table).
- ❖ **On Delete set Null:** when data is removed from a parent table, the foreign key associated cell will be null in a child table.
- ❖ **On Delete Restrict:** when data is removed from a parent table, and there is a foreign key associated with child table it gives error, you can not delete the record.

Khởi tạo dữ liệu - UPDATE

23

Cập nhật dữ liệu

UPDATE table_name

SET table_column = value

WHERE <condition>

```
UPDATE KháchHang  
SET LoaiKH = 2
```

```
UPDATE KháchHang  
SET LoaiKH = 2  
WHERE MaKH = 15
```

DML - SELECT

24

Syntax

SELECT [ALL/DISTINCT] <Column name1>, <Column name2>,

...

FROM <Table name>

[WHERE <Search condition>]

[GROUP BY grouping columns]

[HAVING search condition]

[ORDER BY sort specification]

Exp: SELECT SNAME, CITY FROM S



Thứ tự hoạt động của câu lệnh

Tối thiểu có SELECT-FROM

Không thay đổi thứ tự các mệnh đề trong câu truy vấn

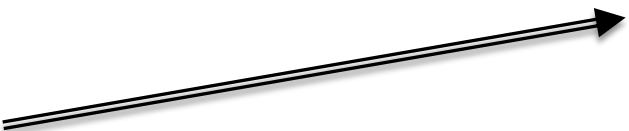
Không phân biệt hoa thường

DML - SELECT

25

Syntax

7 Filter columns
1 FROM TABLE
2 [WHERE] Filter rows
3 [GROUP BY] Then filter rows, Filter group return one data row
4 [HAVING] Filter groups
5 [ORDER BY] Sort by columns
[LIMIT offset, row_count] restrict amount of data, pagination
6



Sau khi lọc bảng theo điều kiện where, lọc **các nhóm** trả về dữ liệu theo nhóm thỏa điều kiện

Example

Thống kê số lượng học sinh của mỗi lớp trong khoa CNTT

Thống kê số lượng tin tức của mỗi loại danh mục tin

Đếm số lượng sản phẩm của mỗi loại mặt hàng

DML - SELECT

26

Truy vấn tất cả các dòng trong bảng

SELECT { tất cả các cột }
FROM <Tên bảng>

SELECT *
FROM <table_name>

```
SELECT* FROM KhachHang
```

SELECT {danh sách cột cần lấy}
FROM <Tên bảng>

SELECT columnA, columnB
FROM < table_name >

```
SELECT Maloai, TenLoai,  
FROM LoaiHang
```

```
SELECT 'Mã lớp' = malop,  
       tenlop 'Tên lớp',  
       khoa AS 'Khoá'  
FROM lop
```

DML - SELECT

27

DISTINCT & TOP - LIMIT

Loại trừ giá trị trùng nhau trong kết quả trả về

SELECT DISTINCT <Danh sách các thuộc tính> **FROM**

SELECT DISTINCT khoa **FROM** lop

SELECT TOP 5 hodem,ten,ngaysinh

FROM sinhvien

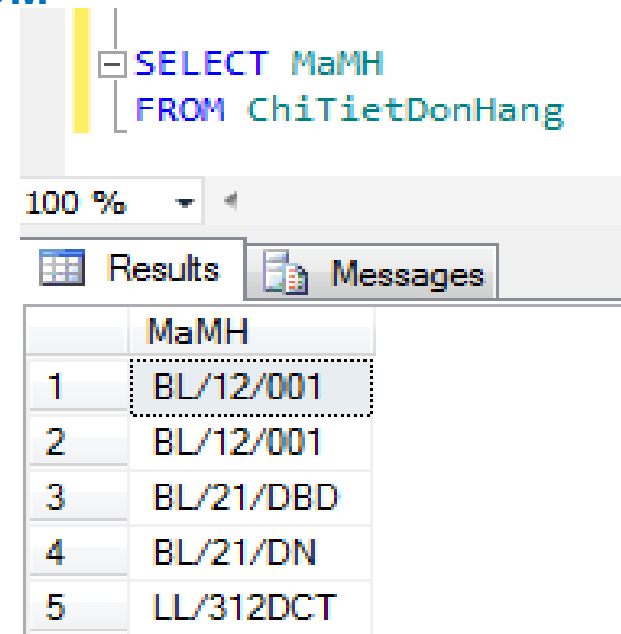
SELECT TOP 10 PERCENT hodem,ten,ngaysinh

FROM sinhvien

SELECT maHS, tenHS

FROM hocsinh

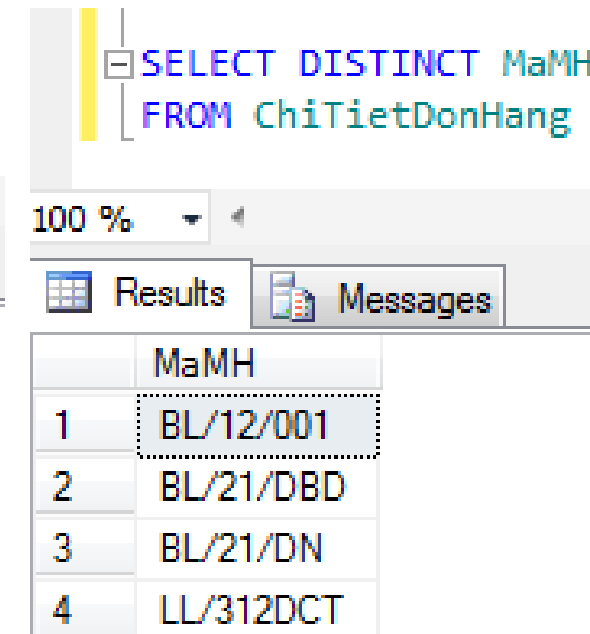
LIMIT offset, rowcount



100 %

Results Messages

	MaMH
1	BL/12/001
2	BL/12/001
3	BL/21/DBD
4	BL/21/DN
5	LL/312DCT



100 %

Results Messages

	MaMH
1	BL/12/001
2	BL/21/DBD
3	BL/21/DN
4	LL/312DCT

DML - SELECT

28

PHÉP TÍNH TRÊN THUỘC TÍNH

Lấy thông tin tất cả mặt hàng kèm theo giá bán mới sau khi đã giảm 10 %

```
| SELECT *, GiaBan *0.9 as GiaMoi  
FROM MatHang
```

DML - SELECT

29

Chọn ra một số dòng - WHERE

SELECT {danh sách tất cả các tên cột}

FROM <Tên bảng>

WHERE <điều kiện>

Lấy thông tin chi tiết của mặt hàng có mã số “BL/12/001”

```
SELECT *  
FROM MatHang  
WHERE MaMH = 'BL/12/001'
```

Use CASE structure in SELECT

30

Syntax:

```
CASE biểu_thức  
WHEN biểu_thức_kiểm_tra THEN kết_quả  
[ ... ]  
[ELSE kết_quả_của_else]  
END
```

```
SELECT masv,hodem,ten,  
       CASE gioitinh  
         WHEN 1 THEN 'Nam'  
         ELSE 'Nữ'  
       END AS gioitinh  
FROM sinhvien
```

Logical Conditions

Operator	Meaning
=	Equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
<>	Not equal to
BETWEEN ...AND...	Between two values (inclusive)
IN (set)	Match any of a list of values
LIKE	Match a character pattern
IS NULL	Is a null value
AND	Returns TRUE if both component conditions are true
OR	Returns TRUE if either component condition is true
NOT	Returns TRUE if the following condition is false

Truy vấn dữ liệu - WHERE

32

Các điều kiện tìm kiếm

❖ So sánh : =, >, <, >=, <=

❖ Miền giá trị : BETWEEN, NOT BETWEEN

❖ Tập hợp : IN, NOT IN

❖ Điều kiện tìm kiếm chuỗi

❖ Null, Not null

❖ Điều kiện phức : AND, OR, NOT

Truy vấn dữ liệu - WHERE

33

Phép toán so sánh : =, >, <, >=, <=

SELECT {danh sách tất cả các tên cột}

FROM <Tên bảng>

WHERE <Tên cột> <Phép toán so sánh> <giá trị>

↓
Kiểu dữ liệu



↓
Kiểu dữ liệu

```
SELECT *  
FROM DonHang  
WHERE SoLuong = 1
```

```
SELECT *  
FROM LoaiHang  
WHERE TenLoai = N'đầm dài'
```

Truy vấn dữ liệu - WHERE

34

Phép toán so sánh : kiểu ngày giờ

```
] SELECT*  
FROM DonHang  
WHERE NgayDat = '01-02-2016'
```

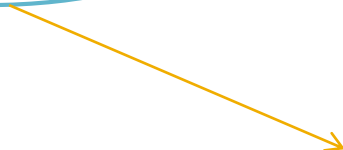
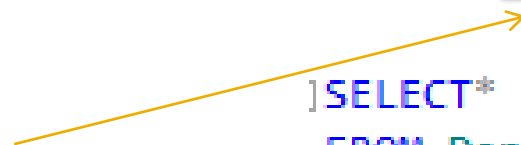
```
] SELECT*  
FROM DonHang  
WHERE NgayDat = '01/02/2016'
```

```
] SELECT*  
FROM DonHang  
WHERE NgayDat = '2016-01-02'
```

```
] SELECT*  
FROM DonHang  
WHERE NgayDat = '2016/01/02'
```

MM-DD-YYYY

YYYY-MM-DD



Truy vấn dữ liệu - WHERE

35

Miền giá trị : BETWEEN , NOT BETWEEN

SELECT {danh sách tất cả các tên cột}

FROM <Tên bảng>

WHERE <Tên cột> **BETWEEN** <giá trị 1> **AND** <giá trị 2>

SELECT*

FROM DonHang

WHERE SoLuong **BETWEEN** 1 **AND** 3

SELECT*

FROM DonHang

WHERE SoLuong **>=** 1 **AND** SoLuong **<=** 3

SELECT {danh sách tất cả các tên cột}

FROM <Tên bảng>

WHERE <Tên cột> **>=** <giá trị 1> **AND** <Tên cột> **<=** <giá trị 2>

Truy vấn dữ liệu - WHERE

36

Tập hợp: IN, NOT IN

SELECT {danh sách tất cả các tên cột}

FROM <Tên bảng>

WHERE <Tên cột> **IN** (giá trị 1, giá trị 2,..., giá trị n)

SELECT*

FROM DonHang

WHERE KháchHang **IN** (40, 41)

SoDH	NgàyDat	Lo...	KháchHang	Hình...	SoLu...	DCGiaoHang
1	2016-01-02 00...	1	40	1	2	37 Hoàng V...
2	2016-02-02 00...	2	41	2	1	15 Phan Xíc...

SELECT*

FROM MatHang

WHERE MaMH **IN** ('BL/21/DN', 'BL/12/001')

Results

Messages

MaMH	TenMH	NgàyTao	SoLuong	GiaMua	GiaBan	Loai	NhaSX
BL/12/001	Áo sơ mi caro	2016-01-01 00:00:00.000	20	250000.00	280000.00	12	BL
BL/21/DN	Đầm xòe cổ tim	2016-01-01 00:00:00.000	15	450000.00	500000.00	21	BL

Truy vấn dữ liệu - WHERE

37

Tập hợp: IN, NOT IN

SELECT {danh sách tất cả các tên cột}

FROM <Tên bảng>

WHERE <Tên cột> **NOT IN** (giá trị 1, giá trị 2,..., giá trị n)

```
SELECT*  
FROM MatHang  
WHERE MaMH NOT IN ('BL/21/DN', 'BL/12/001')
```

MaMH	TenMH	NgayTao	SoLuong	GiaMua	GiaBan	Loai	NhaSX
BL	Đầm cánh tiên	2016-02-02 00:00:00.000	16	130000.00	150000.00	3	LL
BL/12/002	Áo sơ mi jean	2016-01-01 00:00:00.000	5	320000.00	380000.00	12	BL
BL/21/DBD	Đầm body	2016-01-01 00:00:00.000	10	350000.00	390000.00	21	BL
BL/26/001	Áo thun cách điệu	2016-01-01 00:00:00.000	15	200000.00	250000.00	26	BL
LL/312DCT	Đầm cánh tiên	2016-01-01 00:00:00.000	19	130000.00	150000.00	312	LL
LL/312DH	Đầm hồng	2016-01-01 00:00:00.000	19	130000.00	150000.00	312	LL

Truy vấn dữ liệu - WHERE

38

Chuỗi ký tự

SELECT {danh sách tất cả các tên cột}

FROM <Tên bảng>

WHERE <Tên cột> = 'giatri'

Tìm loại hàng có tên là đầm dài

```
] SELECT *  
FROM LoaiHang  
WHERE TenLoai = N'đầm dài'
```

Truy vấn dữ liệu - WHERE

39

Chuỗi ký tự - LIKE , NOT LIKE

SELECT {danh sách tất cả các tên cột}

FROM <Tên bảng>

WHERE <Tên cột> **LIKE** 'giatri'

SELECT*

FROM DonHang

WHERE DCGiaoHang **LIKE** N'37%'

SELECT*

FROM LoaiHang

WHERE TenLoai **LIKE** N'N__'

?: chuỗi ký tự có thể rỗng
hoặc n ký tự
- : 1 ký tự bất kỳ

Truy vấn dữ liệu - WHERE

40

Chuỗi ký tự - LIKE , NOT LIKE

SELECT {danh sách tất cả các tên cột}

FROM <Tên bảng>

WHERE <Tên cột> **NOT LIKE** '*giatri*'

SELECT*

FROM LoaiHang

WHERE TenLoai **NOT LIKE** N'Nam'

Truy vấn dữ liệu - WHERE

41

Kiểu dữ liệu null

```
SELECT {danh sách tất cả các tên cột}  
FROM   <Tên bảng>  
WHERE  <Tên cột> IS null
```

```
SELECT*  
FROM LoaiHang  
WHERE Nhom IS null
```

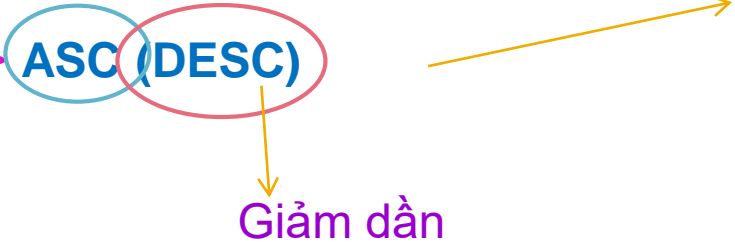
```
SELECT {danh sách tất cả các tên cột}  
FROM   <Tên bảng>  
WHERE  <Tên cột> IS NOT null
```

```
SELECT*  
FROM LoaiHang  
WHERE Nhom IS NOT null
```

ORDER BY

42

SELECT {danh sách tất cả các tên cột} Tăng dần. Mặc định
FROM <Tên bảng>
ORDER BY <tên cột> **ASC** (**DESC**)



Giảm dần

SELECT {danh sách tất cả các tên cột}
FROM <Tên bảng> Trái → Phải
ORDER BY <cột1> <đk sắp xếp> <cột 2> <đk sắp xếp>...

Khi mệnh đề có **GROUP BY**. Dữ liệu sắp khi gom nhóm sẽ được sắp xếp.

ORDER BY

43

Ví dụ:

Sắp xếp mặt hàng theo thứ tự giá bán tăng dần và số lượng theo thứ tự từ nhiều đến ít

```
SELECT *  
FROM   MatHang  
ORDER BY GiaBan ASC, SoLuong DESC
```

```
SELECT *  
FROM   MatHang  
ORDER BY GiaBan, SoLuong DESC
```

GROUP BY

44

Tạo ra các **nhóm dữ liệu** có **cùng giá trị** của **thuộc tính gom nhóm**

Thường được dùng với các hàm min, max, avg, sum, count, group_concat

SELECT {danh sách các tên cột}

FROM <Tên bảng>

GROUP BY <tên cột>  Thuộc tính gom nhóm

Ví dụ: Liệt kê các mặt hàng và phân theo nhóm loại hàng

SELECT MaMH, TenMH, MaLH

FROM MatHang

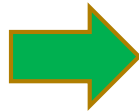
GROUP BY MaLH

GROUP BY

45

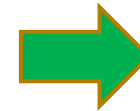
```
50 • SELECT MaMH,  
51      TenMH,  
52      MaLH  
53 FROM MatHang;  
54
```

	MaMH	TenMH	MaLH
▶	1	Áo sơ mi Nam	1
	2	Áo sơ mi Nữ	1
	3	Quần tây Nam	2
	4	Quần jean Nam	2
	5	Quần jean Nữ	2
	6	Giày da Nam	3
	7	Giày thể thao Nữ	3
	8	Thắt lưng Nam	4
	9	Thắt lưng Nữ	4
	10	Mũ thể thao Nam	5



```
50 • SELECT MaMH,  
51      TenMH,  
52      MaLH  
53 FROM MatHang  
54 GROUP BY MaLH;  
55
```

	MaMH	TenMH	MaLH
▶	1	Áo sơ mi Nam	1
	3	Quần tây Nam	2
	6	Giày da Nam	3
	8	Thắt lưng Nam	4
	10	Mũ thể thao Nam	5



```
51 • SELECT MaLH,  
52      COUNT(MaMH) SoLuong  
53 FROM MatHang  
54 GROUP BY MaLH;  
55  
56
```

	MaLH	SoLuong
▶	1	2
	2	3
	3	2
	4	2
	5	1

HAVING

46

Điều kiện trên nhóm

```
SELECT {danh sách tất cả các tên cột}  
FROM <Tên bảng>  
GROUP BY <tên thuộc tính gom nhóm>  
HAVING <điều kiện>
```

Ví dụ: Tìm tất cả các loại hàng có số lượng mặt hàng lớn hơn 1

```
SELECT Loai, count(SoLuong)  
FROM MatHang  
GROUP BY Loai  
HAVING count(soluong) > 1
```

HAVING

47

Điều kiện trên nhóm

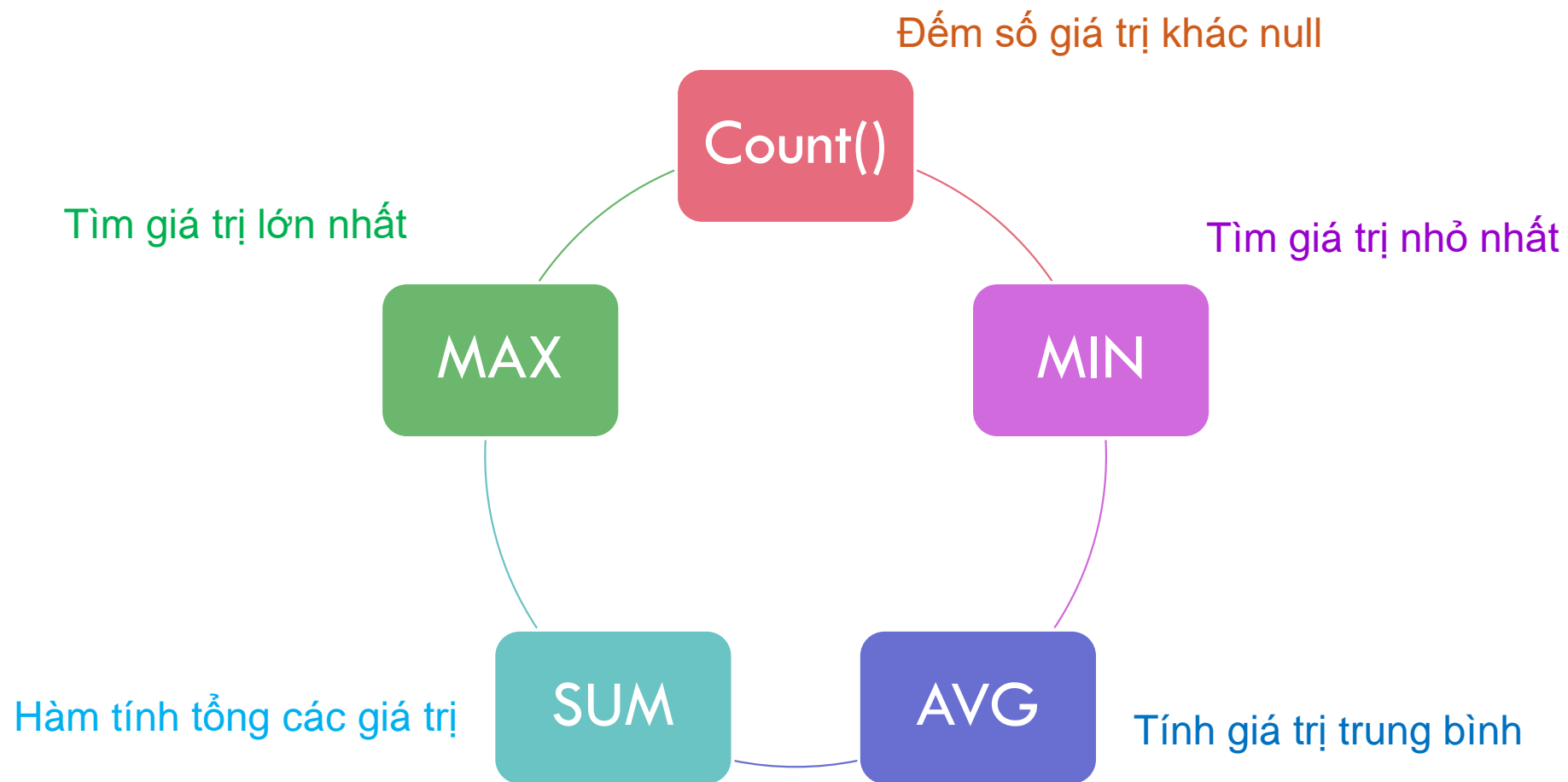
Chỉ kiểm tra điều kiện trên nhóm, không là điều kiện lọc trên tất cả dòng dữ liệu của bảng

Chỉ thực hiện khi có GROUP BY

Sử dụng các hàm kết hợp trong mệnh đề SELECT để gom nhóm

CÁC HÀM TÍNH TOÁN

48



CÁC HÀM TÍNH TOÁN

49

Đặc điểm

1. Nhận tên một cột Trả về **Một giá trị**
2. Hàm sum, avg: chỉ áp dụng cho trường kiểu số
3. Hàm count, min, max có thể áp dụng cho trường kiểu số và kiểu dữ liệu khác
4. Chỉ có hàm **count(*)** thực hiện được trên giá trị null, đếm số dòng
5. Mệnh đề SELECT chứa hàm tính toán nếu
Có GROUP BY → Chỉ được liệt kê thuộc tính đơn
trong group by

CÁC HÀM TÍNH TOÁN

50

Count star vs Count column vs Count 1

```
SELECT COUNT(*) FROM TABLE1
```

COUNT(*): Đếm số lượng dòng của table = COUNT(1)

COUNT(column): Đếm số lượng giá trị not-null của column trong table

Nếu run count(column) với column là non-null thì RDBMS(depends) sẽ convert nó thành count(*). Bởi vì performance cao hơn. Nó không phải kiểm tra giá trị của mỗi dòng, chỉ cần đếm số dòng.

COUNT(distinct column): đếm số giá trị UNIQUE-NOT NULL trong column.

WHERE – OR - AND

51

OR = hoặc : hoặc thỏa điều kiện này hoặc thỏa điều kiện khác

AND = và : Thỏa cùng lúc điều kiện này và điều kiện khác

Ví dụ: Tìm những mặt hàng có số lượng lớn hơn 20 hoặc có giá bán là 150.000

SELECT*

FROM MatHang

WHERE SoLuong >= 20 **OR** GiaBan = 150000

	MaMH	TenMH	NgayTao	SoLuong	GiaMua	GiaBan	Loai	NhaSX
1	BL	Đầm cánh tiên	2016-01-02 00:00:00.000	16	130000.00	150000.00	3	LL
2	BL/12/001	Áo sơ mi caro	2016-01-01 00:00:00.000	20	250000.00	280000.00	12	BL
3	LL/312DCT	Đầm cánh tiên	2016-01-01 00:00:00.000	19	130000.00	150000.00	312	LL
4	LL/312DH	Đầm hồng	2016-01-02 00:00:00.000	19	130000.00	150000.00	312	LL

WHERE – OR - AND

52

Ví dụ: Tìm những chiếc đầm có số bán là dưới 250.000

SELECT *

FROM MatHang

WHERE TenMH like N'%đầm%' **AND** GiaBan < 250000

	MaMH	TenMH	NgayTao	SoLuong	GiaMua	GiaBan	Loai	NhaSX
1	BL	Đầm cánh tiên	2016-01-02 00:00:00.000	16	130000.00	150000.00	3	LL
2	LL/312DCT	Đầm cánh tiên	2016-01-01 00:00:00.000	19	130000.00	150000.00	312	LL
3	LL/312DH	Đầm hồng	2016-01-02 00:00:00.000	19	130000.00	150000.00	312	LL

UNION - INTERSECT

53

UNION : phép hội

```
SELECT <danh sách tên cột>  
FROM <Tên bảng 1>  
WHERE <điều kiện1>  
UNION  
SELECT <danh sách tên cột>  
FROM <Tên bảng 2>  
WHERE <điều kiện2>
```

```
SELECT MaKH  
FROM KhachHang  
WHERE LoaiKH = 2  
UNION  
SELECT KhachHang  
FROM DonHang
```

Số column được chọn từ
những table phải bằng nhau
và nên cùng là một column

Tìm khách hàng có mã loại = 2 hoặc khách hàng đã mua hàng

UNION - INTERSECT

54

INTERSECT: phép giao

```
SELECT <danh sách tên cột>  
FROM <Tên bảng1>  
WHERE <điều kiện1>  
INTERSECT  
SELECT <danh sách tên cột>  
FROM <Tên bảng 2>  
WHERE <điều kiện2>
```

```
SELECT MaKH  
FROM KhachHang  
WHERE LoaiKH = 2  
INTERSECT  
SELECT KhachHang  
FROM DonHang
```

Tìm khách hàng có mã loại = 2 và đã mua hàng

UNION - INTERSECT

55

Notes:

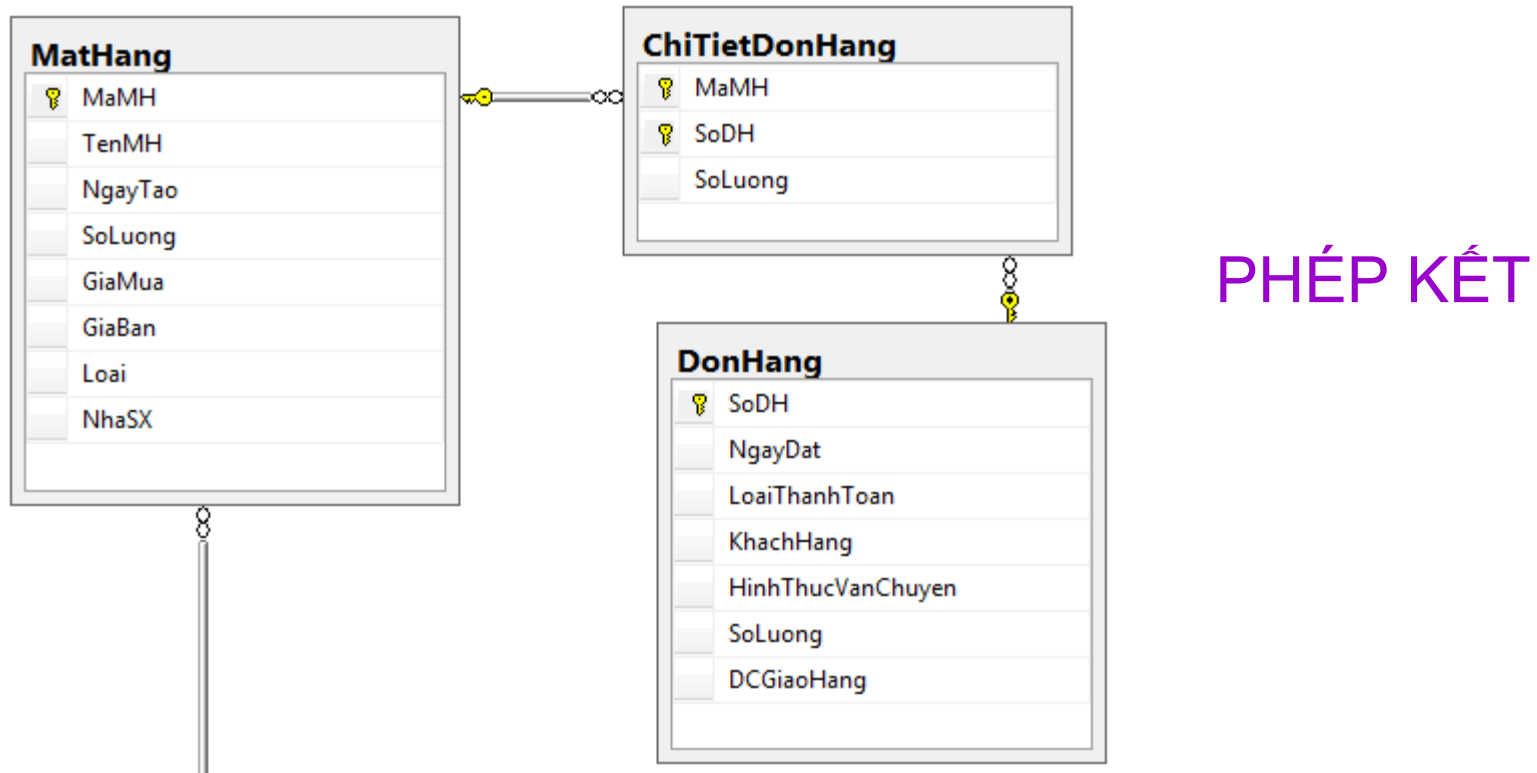
Trả về kết quả không trùng nhau

Dễ lấy tất cả các kết quả, thêm từ khóa ALL vào

- **UNION ALL**
- **INTERSECT ALL**
- **ALREADY DEFAULT**

PHÉP KẾT

56



Làm thế nào để lấy thông tin chi tiết (tên mặt hàng, số lượng, giá bán) các mặt hàng trong đơn hàng?

PHÉP KẾT

57

- Dùng khi truy xuất dữ liệu từ nhiều bảng
- Liên kết 2 hay nhiều bảng với nhau dựa trên 1 thuộc tính(khóa ngoại)

Các loại phép kết

Phép kết tự nhiên
JOIN

Phép kết trong
INNER JOIN

Phép kết trái
LEFT JOIN

Phép kết phải
RIGHT JOIN

Phép kết ngoài
OUTER JOIN

PHÉP KẾT

58

SELECT <danh sách thuộc tính>

FROM Table1 tbl1 **JOIN** Table2 tbl2 **ON** tbl1.ThuocTinh = tbl2.ThuocTinh

OUTER JOIN

INNER JOIN

LEFT JOIN

RIGHT JOIN

MaMH	TenMH	Loai
001	Áo sơ mi	A
002	Đầm	B
003	Jean	C
004	Áo thun	A

MaMH	SoHD	SoLuong
001	001	3
003	002	2

JOIN

59

SELECT *

INNER JOIN

FROM Mathang a **JOIN** ChiTietDonHang b **ON** a.maMH = b.MaMH

MaMH	TenMH	Loai
001	Áo sơ mi	A
002	Đầm	B
003	Jean	C
004	Áo thun	A

MaMH	SoHD	SoLuong
001	001	3
003	002	2

MaMH	TenMH	Loai	MaMH	SoHD	SoLuong
001	Áo sơ mi	A	001	001	3

LEFT JOIN

60

SELECT *

FROM Mathang a **LEFT JOIN** ChiTietDonHang b **ON** a.maMH = b.MaMH

MaMH	TenMH	Loai
001	Áo sơ mi	A
002	Đầm	B
003	Jean	C
004	Áo thun	A

MaMH	SoHD	SoLuong
001	001	3
003	002	2



MaMH	TenMH	Loai	MaMH	SoHD	SoLuong
001	Áo sơ mi	A	001	001	3
002	Đầm	B	Null	Null	Null
003	Jean	C	003	002	2
004	Áo thun	A	Null	Null	null

RIGHT JOIN

61

SELECT *

FROM ChiTietDonHang a **RIGHT JOIN** Mathang b **ON** a.maMH = b.MaMH

MaMH	SoHD	SoLuong
001	001	3
003	002	2

MaMH	TenMH	Loai
001	Áo sơ mi	A
002	Đầm	B
003	Jean	C
004	Áo thun	A

MaMH	SoHD	SoLuong	MaMH	TenMH	Loai
001	001	3	001	Áo sơ mi	A
Null	Null	Null	002	Đầm	B
003	002	2	003	Jean	C
Null	Null	null	004	Áo thun	A

OUTER JOIN

62

SELECT *

FROM ChiTietDonHang a **OUTER JOIN** Mathang b **ON** a.maMH = b.MaMH

MaMH	SoHD	SoLuong
001	001	3
003	002	2
005	004	1

MaMH	TenMH	Loai
001	Áo sơ mi	A
002	Đầm	B
003	Jean	C
004	Áo thun	A

MaMH	SoHD	SoLuong	MaMH	TenMH	Loai
001	001	3	001	Áo sơ mi	A
Null	Null	Null	002	Đầm	B
003	002	2	003	Jean	C
Null	Null	null	004	Áo thun	A
005	004	1	Null	Null	Null

TRUY VẤN CON

63

Là câu **truy vấn** xuất hiện trong một câu truy vấn khác

Có thể xuất hiện trong các mệnh đề WHERE, HAVING, INSERT, UPDATE, DELETE

Tìm khách hàng có mã loại = 2 đã mua hàng (Xuất hiện trong mệnh đề WHERE)

SELECT MaKH

FROM KhachHang

WHERE LoaiKH = 2 **AND** MaKH **IN** (**SELECT**

KhachHang

FROM DonHang)

TRUY VẤN CON

64

Cập nhật tất cả các sản phẩm thuộc loại hàng "T-Shirt" với giá bán là 99 nghìn
(Xuất hiện trong mệnh đề WHERE của câu lệnh UPDATE)

UPDATE mathang

SET GiaBan = 99

WHERE MaLoai = (SELECT MaLoai

FROM loaihang

WHERE TenLoai = "T-Shirt");

TRUY VẤN CON

65

*Tạo mới table LoaiHangGoc và sao chép dữ liệu từ table LoaiHang sang LoaiHangGoc
(Xuất hiện trong mệnh đề INSERT)*

```
INSERT INTO LoaiHangGoc(MaLHGoc, TenLHGoc)
```

```
SELECT MaLoai,TenLoai
```

```
FROM LoaiHang
```

TRUY VẤN LÒNG

66

Là câu truy vấn có chứa “truy vấn con” ở mệnh đề WHERE

SELECT <danh sách tên cột>

FROM <Tên bảng>

WHERE <biểu thức so sánh> (**SELECT** <danh sách thuộc tính>

FROM Tên bảng

WHERE <điều kiện>)

Biểu thức so sánh thường có: **IN**, **NOT IN**, **ALL**, **ANY**

EXISTS, **NOT EXISTS**

TRUY VẤN LỒNG

67

Có 2 loại truy vấn lồng

Lồng phân cấp = Truy vấn con

Lồng tương quan

❖ Lồng phân cấp

SELECT MaKH

FROM KhachHang

WHERE LoaiKH = 2 **AND** MaKH **IN** (**SELECT** KhachHang

FROM DonHang

WHERE SoLuong > 1)

Sử dụng toán tử
IN, NOT IN, =

Mệnh đề WHERE của câu truy vấn con **không tham chiếu** đến giá trị thuộc tính của các bảng ở mệnh đề FROM của truy vấn cha

Thực thi câu truy
vấn con trước

TRUY VẤN LỒNG

68

❖ Lồng tương quan

```
SELECT MaKH
FROM KhachHang
WHERE LoaiKH = 2 AND EXISTS (SELECT MaKH || 1 || *
                              FROM DonHang
                              WHERE MaKH = MaKH)
```

Sử dụng toán tử
EXIST, NOT EXIST

Mệnh đề **WHERE** của câu truy vấn **con** tham
chiếu đến ít nhất 1 thuộc tính của các bảng ở
mệnh đề **FROM** của truy vấn cha

Thực hiện song song
theo cơ chế
“at least one found”

TRUY VẤN LÒNG

69

IN và EXISTS

IN

<tên cột> IN <câu truy vấn con>

Thuộc tính trong SELECT của truy vấn con CÙNG kiểu dữ liệu thuộc tính trong WHERE của truy vấn cha

EXISTS

EXISTS <câu truy vấn con>

Không nhất thiết liệt kê thuộc tính ở mệnh đề SELECT của câu truy vấn con

TRUY VẤN LÒNG

70

The query that uses the **EXISTS** operator is much faster than the one that uses the **IN** operator.

The reason is that the **EXISTS** operator works based on the "at least found" principle. It returns true and stops scanning table once at least one matching row found.

On the other hands, when the **IN** operator is combined with a subquery, MySQL must process the subquery first and then uses the result of the subquery to process the whole query.

The general rule of thumb is that if the subquery contains a large volume of data, the **EXISTS** operator provides better performance.

However, the query that uses the **IN** operator will perform faster if the result set returned from the subquery is very small.

For example, the following statement uses the **IN** operator selects all employees who work at the office in San Francisco.

INDEX in SQL

71

An index is used to speed up searching in the database. MySQL have some good documentation on the subject (which is relevant for other SQL servers as well):

<http://dev.mysql.com/doc/refman/5.0/en/mysql-indexes.html>

An index can be used to efficiently find all rows matching some column in your query and then walk through only that subset of the table to find exact matches. If you don't have indexes on any column in the `WHERE` clause, the `SQL` server has to walk through *the whole table* and check every row to see if it matches, which may be a slow operation on big tables.

The index can also be a `UNIQUE` index, which means that you cannot have duplicate values in that column, or a `PRIMARY KEY` which in some storage engines defines where in the database file the value is stored.

In MySQL you can use `EXPLAIN` in front of your `SELECT` statement to see if your query will make use of any index. This is a good start for troubleshooting performance problems.

Read more here: <http://dev.mysql.com/doc/refman/5.0/en/explain.html>

INDEX in SQL

72

The best way to improve the performance of `SELECT` operations is to create indexes on one or more of the columns that are tested in the query. The index entries act like pointers to the table rows, allowing the query to quickly determine which rows match a condition in the `WHERE` clause, and retrieve the other column values for those rows. All MySQL data types can be indexed.

Although it can be tempting to create an indexes for every possible column used in a query, unnecessary indexes waste space and waste time for MySQL to determine which indexes to use. Indexes also add to the cost of inserts, updates, and deletes because each index must be updated. You must find the right balance to achieve fast queries using the optimal set of indexes.

Indexes are used to find rows with specific column values quickly. Without an index, MySQL must begin with the first row and then read through the entire table to find the relevant rows. The larger the table, the more this costs. If the table has an index for the columns in question, MySQL can quickly determine the position to seek to in the middle of the data file without having to look at all the data. This is much faster than reading every row sequentially.

Most MySQL indexes (`PRIMARY KEY`, `UNIQUE`, `INDEX`, and `FULLTEXT`) are stored in [B-trees](#). Exceptions: Indexes on spatial data types use R-trees; `MEMORY` tables also support [hash indexes](#); InnoDB uses inverted lists for `FULLTEXT` indexes.

INDEX in SQL

73



	MaMH	TenMH	MauSac	ChatLieu	GiaBan	GiaMua	HinhAnh	MaLoai
1	1	Áo sơ mi Nam	Trắng	UD	199	160	ud.png	1
2	2	Áo sơ mi Nữ	Hồng	UD	199	220	ud.png	1
3	3	Quần tây Nam	Xanh	UD	300	800	ud.png	2
4	4	Quần jean Nam	Nâu	UD	220	600	ud.png	2
5	5	Quần jean Nữ	Trắng	UD	140	280	ud.png	2
6	6	Giày da Nam	Đen	UD	220	750	ud.png	3
7	7	Giày thể thao Nữ	Vàng	UD	240	780	ud.png	3
8	8	Thắt lưng Nam	Hồng	UD	40	260	ud.png	4
9	9	Thắt lưng Nữ	Xanh	UD	42	220	ud.png	4

```
SELECT *  
FROM MatHang  
WHERE MaLoai = 2
```

```
ALTER TABLE TBNAME  
ADD INDEX IDXNAME(COLNAME ...)
```

```
CREATE INDEX IDXNAME  
ON TABLE(COLNAME ...)
```

INDEX in SQL

74

```
CREATE TABLE test (  
  id          INT NOT NULL,  
  last_name   CHAR(30) NOT NULL,  
  first_name  CHAR(30) NOT NULL,  
  PRIMARY KEY (id),  
  INDEX name  (last_name, first_name)  
);
```

```
SELECT * FROM test WHERE last_name='Jones';  
  
SELECT * FROM test  
  WHERE last_name='Jones' AND first_name='John';  
  
SELECT * FROM test  
  WHERE last_name='Jones'  
    AND (first_name='John' OR first_name='Jon');  
  
SELECT * FROM test  
  WHERE last_name='Jones'  
    AND first_name >='M' AND first_name < 'N';
```

INDEX in SQL

75

Verifying Index Usage

Table 8.1 EXPLAIN Output Columns

Column	JSON Name	Meaning
<u>id</u>	select_id	The <code>SELECT</code> identifier
<u>select_type</u>	None	The <code>SELECT</code> type
<u>table</u>	table_name	The table for the output row
<u>partitions</u>	partitions	The matching partitions
<u>type</u>	access_type	The join type
<u>possible_keys</u>	possible_keys	The possible indexes to choose
<u>key</u>	key	The index actually chosen
<u>key_len</u>	key_length	The length of the chosen key
<u>ref</u>	ref	The columns compared to the index
<u>rows</u>	rows	Estimate of rows to be examined
<u>filtered</u>	filtered	Percentage of rows filtered by table condition
<u>Extra</u>	None	Additional information

INDEX in SC

76

```
4 • EXPLAIN SELECT * FROM web_customer_tracker.customer
5 WHERE id = 3;
```

Result Grid Filter Rows: Export: Wrap Cell Content:											
id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	customer	NULL	const	PRIMARY	PRIMARY	4	const	1	100.00	NULL

```
7 • EXPLAIN SELECT *
8 FROM MatHang mh
9 JOIN LoaiHang lh
10 ON mh.MaLH = lh.MaLH
11 WHERE mh.MaLH = 2;
```

	MaLH	TenLH
▶	1	Áo
	2	Quần
	3	Giày dép
	4	Thắt lưng
	5	Mũ
	6	Tất

Result Grid Filter Rows: Export: Wrap Cell Content:											
id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	lh	NULL	const	PRIMARY	PRIMARY	4	const	1	100.00	NULL
1	SIMPLE	mh	NULL	ref	fk_MatHang_LoaiHang1_idx	fk_MatHang_LoaiHang1_idx	4	const	3	100.00	NULL

	MaMH	TenMH	MauSac	MaLH	MaLH	TenLH
▶	3	Quần tây Nam	Xanh	2	2	Quần
	4	Quần jean Nam	Nâu	2	2	Quần
	5	Quần jean Nữ	NULL	2	2	Quần

```
7 • EXPLAIN SELECT *
8 FROM MatHang mh
9 JOIN LoaiHang lh
10 ON mh.MaLH = lh.MaLH
11 WHERE lh.TenLH = 'Quần';
```

Result Grid Filter Rows: Export: Wrap Cell Content:											
id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	lh	NULL	ALL	PRIMARY	NULL	NULL	NULL	6	16.67	Using where
1	SIMPLE	mh	NULL	ref	fk_MatHang_LoaiHang1_idx	fk_MatHang_LoaiHang1_idx	4	java11_shopping.lh.MaLH	2	100.00	NULL

PROCEDURE

77

```
-- SYNTAX
```

```
DELIMITER $$
```

```
CREATE PROCEDURE function_name(params,...)
```

```
BEGIN
```

```
    -- declaration_section
```

```
    -- executable_section
```

```
END $$
```

```
DELIMITER;
```

PROCEDURE

78

```
DELIMITER $$
CREATE PROCEDURE procedure_name(p_rows INT)
BEGIN
    DECLARE i INT DEFAULT default_value;
    DECLARE i INT;

    -- DO WHILE
    WHILE i < n DO
        -- DO SOMETHING
        SET i = i + 1;
    END WHILE;
END
```

```
DELIMITER $$
CREATE PROCEDURE procedure_name(p_rows INT)
BEGIN
    DECLARE i INT DEFAULT default_value;
    DECLARE i INT;

    -- FOR LOOP
    label_name: LOOP
        IF i > n THEN
            LEAVE label_name;
        END IF;

        -- DO SOMETHING
        SET i = i + 1;
    END LOOP;
END
```

FUNCTION

79

-- SYNTAX

DELIMITER \$\$

CREATE FUNCTION function_name(params,...)

RETURNS data_type

BEGIN

-- declaration_section

-- executable_section

RETURN ...

END \$\$

CREATE DEFINER=`root`@`localhost` FUNCTION `calculation`(n int)

RETURNS int(11)

BEGIN

DECLARE result INT DEFAULT 0;

DECLARE i INT DEFAULT 0;

WHILE i < n DO

SET result = result + i;

SET i = i + 1;

END WHILE;

RETURN result;

END

FUNCTION

80

Built-in Function Determinism

DETERMINISTIC	NONDETERMINISTIC
Returns same result for same input.	Returns different result for same input.
deterministic giving significant time for execution if it is giving same result.	Executing method definition again and again for same input. Take more execution time compare than determinisitic
When using nondeterministic for deterministic type of functions will take unwanted execution time. Because unwantly executing again and again for the same output.	When using deterministic for nondeterministic methods might return wrong results. Because not executing for getting different outside at all time for the same input.

PRACTICE

81

PROCEDURE

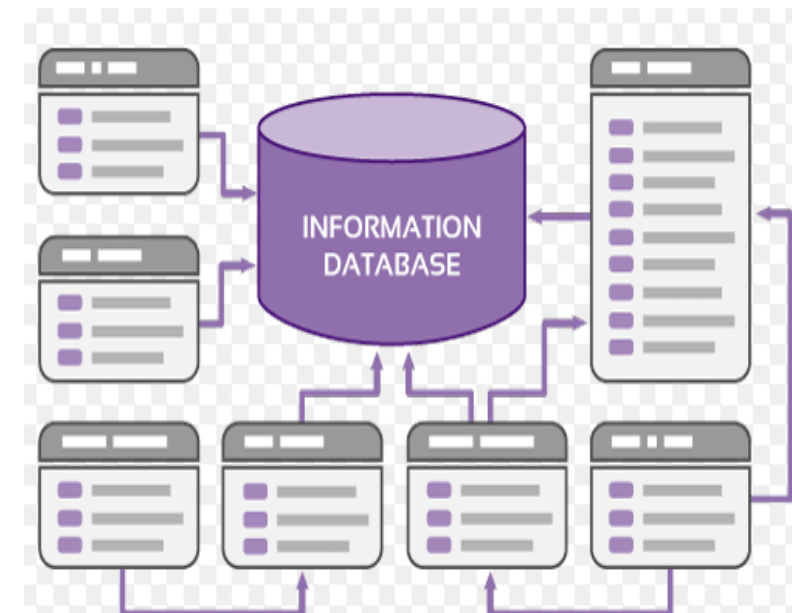
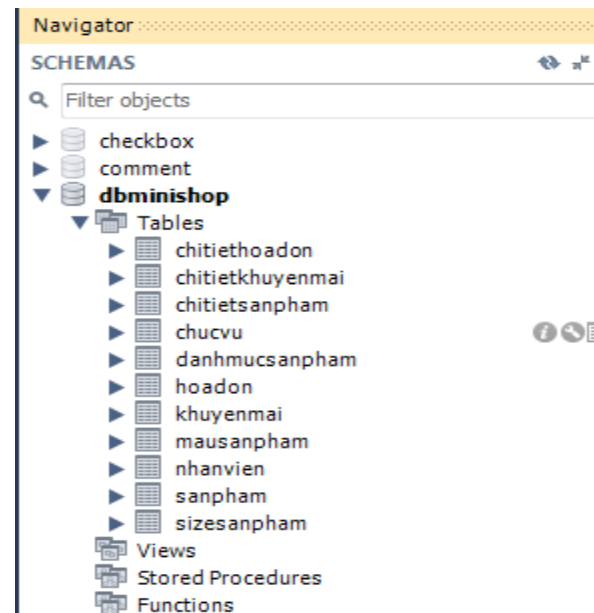
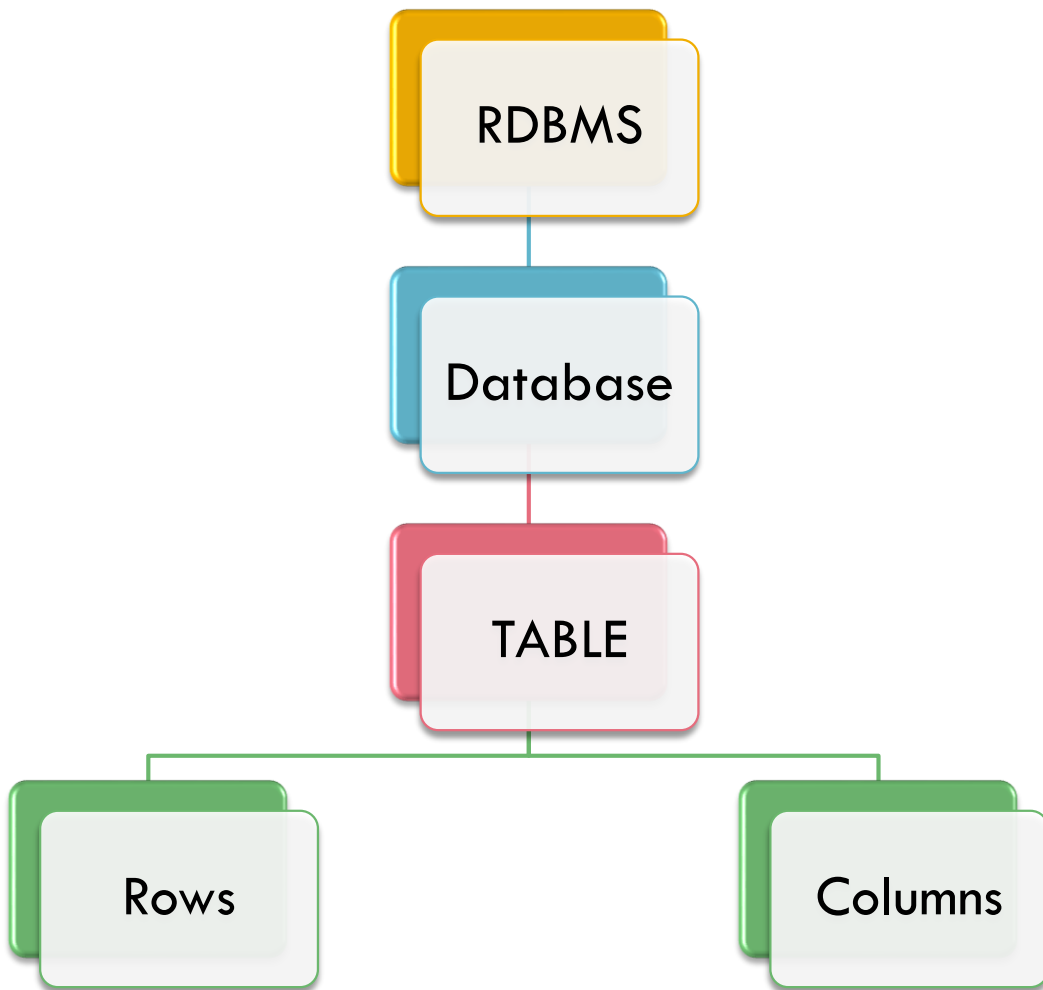
1. Viết phương thức liệt kê các mặt hàng
2. Viết phương thức liệt kê các mặt hàng theo loại hàng
3. Viết phương thức thêm N dòng dữ liệu cho bảng loại hàng
4. Viết phương thức liệt kê các phần tử chẵn nhỏ hơn N

FUNCTION

1. Tính tổng các phần tử chẵn nhỏ hơn N

CẤU TRÚC CHUNG

82



Result Grid

Filter Rows:

Edit:

Export/Import:

Wrap Cell Content:

	masanpham	madanhmuc	tensanpham	giatien	mota	hinhshanpham
1	1	Sơ Mi Nam No Style TN O01	185,000	- Thiết kế áo sơ mi kiểu dáng basic, dễ dàng mix...	79fb5c06-b673-9300	
2	1	Sơ Mi Nam No Style TD R02	225,000	- Thiết kế áo sơ mi kiểu dáng basic, dễ dàng mix...	b443d8c7-54cc-4500	
3	1	Sơ Mi Nam No Style TN O02	185,000	- Thiết kế áo sơ mi kiểu dáng basic, dễ dàng mix...	7cb8e2db-a600-9c00	
4	1	Sơ Mi Nam No Style TN L01	225,000	- Thiết kế đơn giản và hiện đại với sơ mi tay ngắn...	4364f967-2c49-6600	
5	1	Sơ Mi Nam No Style TD ST01	225,000	- Thiết kế đơn giản và hiện đại với sơ mi tay ngắn...	d5ebdfa6-2945-2300	
6	1	Sơ Mi Nam No Style TN N03	225,000	- Thiết kế đơn giản và hiện đại với sơ mi tay ngắn...	71be5178-0fd9-3300	
7	7	Sơ Mi Adachi / 0012655	225,000	- Thiết kế đơn giản và hiện đại với sơ mi tay ngắn...	d5ebdfa6-2945-2300	
8	7	Sơ Mi Adachi / 0012738	225,000	- Thiết kế đơn giản và hiện đại với sơ mi tay ngắn...	a4f67968-f23e-0100	
9	7	Sơ Mi Adachi / 0012658	225,000	- Thiết kế đơn giản và hiện đại với sơ mi tay ngắn...	deec8c60-24e0-2400	