Ngôn Ngữ SQL

Nguyễn Đức Cương cuongnguyenduc@gmail.com/
nguyenduccuong@iuh.edu.vn
Website: http://nguyenduccuong.com

Nội Dung

- Giới thiệu Ngôn Ngữ SQL
  Nhóm lệnh DDL
  Nhóm lệnh DML
  Câu lệnh Select

Giới thiệu Ngôn Ngữ SQL

Microsoft SQL Server là một hệ quản trị cơ sở dữ liệu quan hệ (Relation database management system – RDBMS) chạy trên hệ thống mạng Windows NT 4 hay Windows.

Chuẩn ANSI SQL được công bố vào năm 1989, 1992 và 1999

Đặc điểm của SQL – Structure Query Language:
Là ngôn ngữ tựa tiếng Anh
Ngôn ngữ phi thủ tục

Giới thiệu

- Các lệnh trong SQL có thể phân làm 3 loại:
  - Ngôn ngữ định nghĩa dữ liệu (Data Definition Language commands - DDL)
  - Ngôn ngữ thao tác dữ liệu (Data Manipulation Language commands -DML)
  - Ngôn ngữ điều khiển dữ liệu (Data Control Language commands -DCL)

Giới thiệu SQL

– Data Definition Language Statements (DDL)

create alter drop

– Data Control Language Statements (DCL)

deny grant revoke

– Data Manipulation Language Statements (DML)

select insert update delete

5

Creating a New Database

- Cú pháp lệnh tạo CSDL :

```
CREATE DATABASE database_name
[ ON
[ < filespec > [ ,...n ] ]
[ , < filegroup > [ ,...n ] ]
]
[ LOG ON { < filespec > [ ,...n ] } ]
```

- Cú pháp Filespec:

```
(NAME = logical_name,
FILENAME = 'path\filename',
SIZE = size_in_MB,
MAXSIZE = size_in_MB | UNLIMITED,
FILEGROWTH = % or MB)
```


6

Creating a New Database

- Some arguments:
  - The name of the database
  - The size of the database

```
CREATE DATABASE Sample
ON
   PRIMARY ( NAME=SampleData,
   FILENAME='c:\Program Files\..\..\Data\Sample.mdf',
   SIZE=10MB,
   MAXSIZE=15MB,
   FILEGROWTH=20%)
LOG ON
   ( NAME=SampleLog,
   FILENAME= 'c:\Program Files\..\..\Data\Sample.ldf',
   SIZE=3MB,
   MAXSIZE=5MB,
   FILEGROWTH=1MB)
COLLATE SQL_Latin1_General_Cp1_CI_AS
```

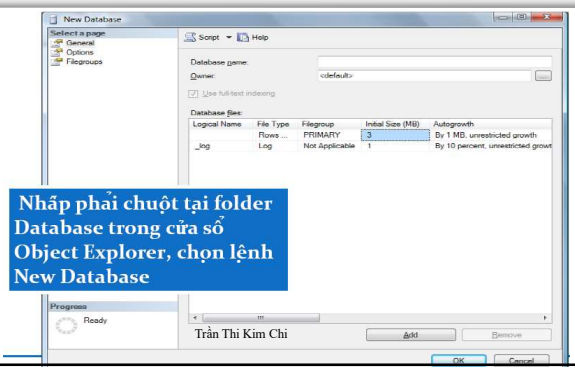

7

Creating a New Database

```
CREATE DATABASE Sales
ON PRIMARY
( NAME = Sales1_dat, FILENAME = 'D:\BTSQL\Sales_dat.mdf',
    SIZE = 10, MAXSIZE = 50, FILEGROWTH = 15% ),
( NAME = Sales2_dat, FILENAME = 'D:\BTSQL\Sales2_dat.ndf',
    SIZE = 10, MAXSIZE = 50, FILEGROWTH = 15% ),
FILEGROUP SalesGroup1
( NAME = Sales3_dat, FILENAME = 'D:\BTSQL\Sales3_dat.ndf',
    SIZE = 10, MAXSIZE = 50, FILEGROWTH = 5 )
( NAME = Sales4_dat,
    FILENAME = 'D:\BTSQL\Sales4_dat.ndf',
    SIZE = 10, MAXSIZE = 50, FILEGROWTH = 5 )
LOG ON
( NAME = 'Sales_log',
    FILENAME = 'D:\BTSQL\salelog.ldf',
    SIZE = 5MB,
    MAXSIZE = 25MB,
```


8

Creating a New Database (từ menu)



Nhấp phải chuột tại folder Database trong cửa sổ Object Explorer, chọn lệnh New Database

9

Managing Database

Hiển thị thông tin DB

- Mở CSDL
  USE TenCSDl
  Ví dụ:
  use Sales
- Kiểm tra sự tồn tại của CSDL
  sp_helpdb TenCSDL
- Kiểm tra không gian sử dụng của CSDL
  sp_spaceused

10

Managing Databases

Cú pháp lệnh thay đổi cấu trúc CSDL

ALTER DATABASE *database_name*
ADD FILE *filespec* [**TO FILEGROUP** *filegroup_name*]
ADD LOG FILE *filespec*
| **REMOVE FILE** *logical_filename*
| **ADD FILEGROUP** *filegroup_name*
| **REMOVE FILEGROUP** *filegroup_name*
| **MODIFY FILE** *filespec*
| **MODIFY FILEGROUP** *filegroup_name*
filegroup_property
[**SET** *optionspec* [**WITH** *termination*]

11

Managing Databases

Ví dụ thay đổi cấu trúc CSDL

a) Chỉnh sửa Size của tập tin
ALTER DATABASE Sales
MODIFY FILE (NAME = 'Sales_log', size =10MB)

b) Bổ sung thêm một tập tin dữ liệu
ALTER DATABASE Sales
ADD File (Name =Sales_data2, Filename ='D:\BTSQL\Sales_data2.mdf,SIZE =10 MB, Maxsize =20MB)

12

Managing Data and Log File Growth

```
ALTER DATABASE Sample
   MODIFY FILE ( NAME = 'SampleLog',
   SIZE = 15MB)
GO

ALTER DATABASE Sample
ADD FILE
  (NAME = SampleData2,
   FILENAME='c:\Program Files\..\..\
        Data\Sample2.ndf',
   SIZE=15MB,
   MAXSIZE=20MB)
GO
```


Managing Database

- Xem các thuộc tính của CSDL

  SELECT DATABASEPROPERTYEX('databasename', 'property')

  Property: IsAutoShrink, IsCloseCursorsOnCommitEnabled, Recovery, Updateability, UserAccess

Managing Data and Log File Growth

Thay đổi thuộc tính DB

ALTER DATABASE database_name
SET option [, status]

Option

AUTO_SHRINK
CURSOR_CLOSE_ON_COMMIT
RECOVERY FULL | BULK_LOGGED | SIMPLE
SINGLE_USER | RESTRICTED_USER | ULTI_USER
READ_ONLY | READ_WRITE

Example:

ALTER DATABASE Sales
SET Read_Only

Managing Data and Log File Growth

- Đổi tên cơ sở dữ liệu:

  sp_renamedb [@dbname =] 'old_name', [@newname =] 'new_name'

  VD: Sp_ReNamedb 'Sales', 'Banhang'

Managing Data and Log File Growth

Xóa cơ sở dữ liệu:

- Khi 1 CSDL bị xóa thì tất cả các file vật lý của nó sẽ bị xóa
- **Cú pháp:**

  DROP DATABASE *database_name*
- **Ví dụ:**

  Drop database Banhang

Chú ý: Không thể xóa các CSDL master, model, tempdb

System Data Types

Có 2 nhóm:

- **System-Supplied datatype:** Các kiểu dữ liệu cơ bản được hỗ trợ bởi SQL Server.
- **User-defined datatype:** Các kiểu dữ liệu của người dùng tự định nghĩa dựa trên các kiểu dữ liệu cơ bản.

System Data Types

Các kiểu dữ liệu cơ bản:

Loại	Kiểu dữ liệu cơ sở	Kích cỡ	Vùng giá trị	Mô tả
Binary	Binary	8 KB	"0"…"9", "a"..."f", "A"..."F"	Chứa các bit thông tin
	Varbinary	8 KB	"0"…"9", "a"..."f", "A"..."F"	
	Image	2^31 -1 bytes	2^31 –1 bytes	Dữ liệu hình ảnh
Character	Char	255 bytes	1..8000 ký tự	Ký tự hoặc chuỗi
	Varchar	255 bytes	1..8000 ký tự	Ký tự hoặc chuỗi
	Text	2147483647 bytes	2^31-1 ký tự (2147483647)	Ký tự hoặc chuỗi
Date and Time	Datetime	8 bytes	01/01/1753->31/12/9999	Chuỗi biểu diễn ngày giờ
	Smalldatetime	4 bytes	1/1/1900 -> 6/6/2079	Chuỗi biểu diễn ngày giờ
Decimal	Decimal	17 bytes	-10^38-1 -> 10^38-1	Số thực
	Numeric	17 bytes	-10^38-1 -> 10^38-1	Số thực
Foating point	Float	8 bytes	-1.79E+308 -> 1.79E+308	Số thực
	Real	4 bytes	-3.40E+38 ->3.40E+38	Số thực
Integer	Bigint	8 bytes	-2^63 -> 2^63	Số nguyên

System Data Types

	Int	4 bytes	-2^31 -> 2^31-1	Số nguyên
	Smallint	2 bytes	-2^15 -> 2^15-1	Số nguyên
	Tinyint	1 bytes	0..255	Số nguyên
Monetary	Money	8 bytes	-2^63 -> 2^63-1	Dữ liệu tiền tệ
	Smalmoney	4 bytes	-214748.3648 -> 214748.3648	Dữ liệu tiền tệ
Special	Bit	1 bytes	0 hoặc 1	Dữ liệu có một trong hai trạng thái 0 hoặc 1
	Cursor	Kiểu DL cho biến hoặc giá trị trả về của procedure, tham chiếu đến 1 mẫu tin		
	Timestamp	8 bytes	Chuỗi có dạng: 0x0000000100000a90	Theo dõi mẫu tin nào bị thay đổi dữ liệu
	Uniqueidentifier	16 bytes	Số thập lục phân	
	SQL_variant	Là kiểu dữ liệu có thể chứa bất kỳ loại dữ tùy ý của SQL Server ngoại trừ **text**, **ntext**, **image**, and the **timestamp** data type		
	Table			
Unicode	Nchar		4000 ký tự	Ký tự hoặc chuỗi
	Nvarchar		4000 ký tự	Ký tự hoặc chuỗi
	Ntext		2^30-1 ký tự	Ký tự hoặc chuỗi

Bảng dữ liệu - Table

- Các bước tạo một bảng
  - **Bước 1:** Xác định kiểu dữ liệu của các cột.
  - **Bước 2:** Xác định các cột có thể hoặc không thể có giá trị rỗng (*null value*).
  - **Bước 3:** Xác định các cột phải có các giá trị duy nhất.
  - **Bước 4:** Xác định khóa chính – khóa ngoại.
  - **Bước 5:** Xác định các giá trị mặc định.
  - **Bước 6:** Xác định các ràng buộc trên các cột (mô tả miền trị).
  - **Bước 7:** Tạo bảng và các chỉ mục của bảng.

Tạo bảng - CREATE TABLE

```
CREATE TABLE
  [ database_name.[ owner ] .|owner.] table_name
  ({ < column_definition >
        |column_name AS computed_column_expression
        | < table_constraint > ::= [ CONSTRAINT
        constraint_name ] }
        [ { PRIMARY KEY | UNIQUE } [ ,...n ]
  )
[ ON { filegroup | DEFAULT } ]
[ TEXTIMAGE_ON { filegroup | DEFAULT } ]
```


Tạo bảng - CREATE TABLE

Cú pháp

```
CREATE TABLE <Table_Name>
(<Column_Name>  <Data_Type>,….)
```

Ví dụ

```
CREATE TABLE Sanpham
( Masp CHAR(5),
  Tensp VARCHAR(15), Dvt VARCHAR(10), Dongia
SMALLMONEY, SlTon INT)
```


Tạo bảng - CREATE TABLE

IDENTITY [(*seed* , *increment*)]

- Tạo giá trị gia tăng duy nhất cho 1 cột, và cột này thường được dùng khoá chính cho bảng.
- Giá trị được gán thường là các kiểu dữ liệu sau: tinyint, smallint, int, bigint, decimal(p,0), hay numeric(p,0).
- Trong mỗi bảng chỉ cho phép 1 cột là identity mà thôi.
- *Seed: là giá trị đầu tiên được tạo.*
- *Increment:là bước tăng để tạo ra giá trị kế tiếp.*
- Giá trị mặc định thường là (1,1).

Tạo bảng - CREATE TABLE

Cú pháp : Tạo cột có giá trị phát sinh tự động

```
CREATE TABLE <Table_Name>
(<Column_Name>  <Data_Type>
INDENTITY(seed[, Increment]) NOT NULL….)
```

Ví dụ

```
CREATE TABLE NhaCungCap
(MaNCC int Identity NOT NULL Primary key, TenNCC
VarChar(25))
```


25

Tạo bảng - CREATE TABLE

Cột tính toán - Computed column

- *Cú pháp:*
  column_name AS *computed_column_expression*
- Là một cột ảo không được lưu trữ vật lý trong bảng. Nó được tính toán dựa vào các cột khác trong cùng bảng thông qua 1 biểu thức. Ví dụ : cost AS price * qty.
- Được dùng trong mệnh đề SELECT, WHERE, hay ORDER BY. Không thể dùng trong lệnh INSERT hay UPDATE
- Được dùng như giá trị khóa trong chỉ mục hay 1 phần của các ràng buộc PRIMARY KEY hay UNIQUE nếu giá trị của nó được định nghĩa bởi 1 biểu thức xác định và kiểu dữ liệu của giá trị trả về hợp lệ.
- Ví dụ: Cột tính toán a+b có thể được dùng làm chỉ mục nhưng a+DATEPART(dd, GETDATE()) không thể dùng làm chỉ mục

26

Tạo bảng - CREATE TABLE

Cột tính toán - Computed column

- Ví dụ 1

```
CREATE TABLE cthoadon
(   sohd int NOT NULL,
    MaHang char(5) NOT NULL,
    SoLuong int NOT NULL,
    DonGia money,
    ThanhTien AS SoLuong*DonGia
)
```


27

Toàn vẹn dữ liệu (Data integrity)

- Tính toàn vẹn dữ liệu để đảm bảo chất lượng của dữ liệu trong cơ sở dữ liệu.
- Có hai cách để đảm bảo tính toàn vẹn dữ liệu:
  - *Toàn vẹn thủ tục (Procedural integrity:* trigger, stored procedure…)
  - *Toàn vẹn khai báo (Declarative integrity): khai báo các thuộc tính của cơ sở dữ liệu hoặc table* (contraint, default, rule)

28

Toàn vẹn dữ liệu (Data integrity)

- **Có 4 loại bảo toàn dữ liệu**
  - Domain integrity (Bảo toàn miền)
  - Entity integrity (Bảo toàn thực thể)
  - Referential integrity (Bảo toàn tham chiếu)
  - User – defined integrity (Bảo toàn do người dùng qui định)

Constrans – Các ràng buộc

- **SQL hỗ trợ các loại ràng buộc sau:**
  - Default
  - Check
  - Unique
  - Primary key
  - Foregin key

Domain integrity

Entity integrity

Referential integrity

Ràng Buộc mức cột

Ràng Buộc mức Bảng

Các ràng buộc dùng để đảm bảo các giá trị của dữ liệu không vi phạm qui luật bảo toàn dữ liệu.

Bảo toàn thực thể

- **Toàn vẹn thực thể (Entity Integrity):** xác định một dòng như là một thực thể duy nhất trong một bảng cụ thể.
  - Tính toàn vẹn thực thể thể hiện tính toàn vẹn bằng cột định danh hoặc khóa chính của bảng.

Bảo toàn thực thể

- **Ràng buộc khóa chính (Primary Key Constraints)**: khóa chính trong một bảng là một cột hoặc một tập các cột mà nó được sử dụng để xác định một dòng duy nhất trong một bảng.
  - **Cú pháp**:

```
CREATE TABLE table_name
(    column_name  data_type NOT NULL
     [CONSTRAINT constraintname]
     PRIMARY KEY
)
```

Bảo toàn thực thể

- Nếu khóa chính là một tập nhiều cột:

```
CREATE TABLE table_name
(     column_name  data_type[,...]
      [CONSTRAINT constraintname]
      PRIMARY
      KEY{(column1[ASC|DESC][,...columnN])}
)
```


Bảo toàn thực thể

- Ví dụ:

```
CREATE TABLE Monhoc
  (
  MaMH char(10) NOT NULL CONSTRAINT
  PK_MonHoc PRIMARY KEY
  );
CREATE TABLE Ketqua
(
  masv char(10) not null,
  mamh varchar(40) not null,
  Diem float  not null,
  Primary key (masv, mamh)
);
```


Bảo toàn thực thể

- **Ví dụ 3**

```
CREATE TABLE DEAN
(
      MADA smallint PRIMARY KEY,
      TENDA varchar(50) NOT NULL
)
```


Bảo toàn thực thể

Ví dụ 4:

```
CREATE TABLE PHANCONG (
    Manv int NOT NULL, Mada smallint, Sonc int,
    primary key (Manv,Mada))
```

Bảo toàn thực thể

- **Unique Constraints**: để duy trì các giá trị riêng biệt trong một cột hay tập hợp các cột không tham gia vào khóa chính.
  - Có thể chỉ định nhiều Unique constraint trên một bảng
  - Có thể chỉ định Unique constraint trên một hoặc nhiều cột chấp nhận giá trị NULL.
  - Tuy nhiên nếu chỉ định Unique constraint trên một cột thì cột đó chỉ chấp nhận một giá trị NULL.

Bảo toàn thực thể

- **Cú pháp khai báo unique constraint**

```
CREATE TABLE table_name
(     column_name  data_type
      [CONSTRAINT constraint_name] UNIQUE
)
```

Ví dụ:

```
CREATE TABLE HoaDon
(MaHD int NOT NULL CONSTRAINT PK_ORDERS
   PRIMARY KEY, SoHD int NULL CONSTRAINT
   UQ_ORDER_NUMBER UNIQUE)
```


Bảo toàn thực thể

Ràng buộc Unique

Ví dụ 2: Định nghĩa mức cột

```
CREATE TABLE Events (
EventID int NOT NULL UNIQUE,
EventTitle nvarchar (100) NULL ,
EventDescription ntext NULL)
```

Ví dụ 3: Định nghĩa mức bảng

```
CREATE TABLE Orders (
OrderID int IDENTITY (1, 1) NOT NULL,
CustomerID nchar (5), UNIQUE NONCLUSTERED (OrderID) WITH
   FILLFACTOR=90 )
```


Bảo toàn miền giá trị

- **Toàn vẹn miền giá trị (Domain Integrity):** kiểm tra dữ liệu nhập vào các cột có thỏa điều kiện ban đầu không
- Kiểm tra toàn vẹn miền giá trị dựa vào:
  - Kiểu dữ liệu.
  - Định dạng : thông qua CHECK constraints và rules
  - Phạm vi giá trị thông qua FOREIGN KEY constraints, CHECK constraints, DEFAULT, NOT NULL, rules.

Bảo toàn miền giá trị

- **Check Constraints:** giúp thực thi toàn vẹn miền bằng cách xác nhận hoặc kiểm tra các dữ liệu được chèn vào một cột trước khi chấp nhận giá trị. Có thể có nhiều kiểm tra ràng buộc trong một cột

```
CREATE TABLE table_name
    (column_name data_type
    [CONSTRAINT constraint_name]
    CHECK (logical expression)
```


41

Bảo toàn miền giá trị

- **Ví dụ**:

```
CREATE TABLE nhanvien
(
    manv smallint PRIMARY KEY CLUSTERED,
    tennv varchar(50) NOT NULL ,
    tuoimin tinyint NOT NULL CHECK(tuoimin>=18),
    tuoimax tinyint NOT NULL CHECK(tuoimax<=40)
)
```


42

Bảo toàn miền giá trị

```
CREATE TABLE Orders (
OrderID int IDENTITY (1, 1) NOT NULL,
CustomerID nchar (5) CHECK (CustomerID LIKE '[A-Z][A-
  Z][A-Z][A-Z][A-Z]'),
EmployeeID int NULL, OrderDate datetime NULL
CHECK (OrderDate BETWEEN '01/01/70' AND GETDATE()),
RequiredDate datetime NULL, ShipVia int NULL
CHECK (ShipVia IN (1, 2, 3, 4)),
Freight money NULL CHECK (Freight>=0),
ShipCountry nvarchar (15),
CHECK (RequiredDate>OrderDate))
```


43

Bảo toàn miền giá trị

- **Default constraint:** Gán giá trị mặc định cho một cột
- DEFAULT có thể áp dụng cho bất kỳ cột nào trong bảng ngoại trừ cột có kiểu timestamp hay có thuộc tính IDENTITY.
- *constant_expression:* chỉ có giá trị hằng như chuỗi ký tự, hàm hệ thống, hay giá trị NULL

```
CREATE TABLE Table_name
    (Column_name Datatype [NULL| NOT NULL]
     [CONSTRAINT Constraint_name] DEFAULT
        expression[,...])
```


44

Bảo toàn miền giá trị

Ví dụ 1

```
CREATE TABLE HoaDon
(MaHD int, LoaiHD Char(1) DEFAULT 'X', NgayLap DateTime NOT NULL)
```


Bảo toàn tham chiếu

- **Referential Integrity**: Tính toàn vẹn tham chiếu, duy trì các mối quan hệ được xác định giữa các bảng khi một record được nhập vào hoặc xóa.
- Toàn vẹn tham chiếu được dựa trên mối quan hệ giữa **khóa ngoại** và các khóa chính hoặc giữa khóa ngoại và khóa duy nhất (unique keys).

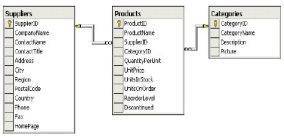
Bảo toàn tham chiếu

- Ràng buộc tham chiếu để tránh cho người dùng:
  - Thêm một record vào một bảng quan hệ nếu không có record liên quan trong bảng chính.
  - Thay đổi các giá trị trong một bảng chính tạo các record mồ côi trong một bảng liên quan.
  - Xóa các bản ghi từ một bảng chính nếu có record trong bảng quan hệ

Bảo toàn tham chiếu

Ràng buộc Foreign key

- Quan hệ chỉ có thể được tạo ra giữa các bảng trong cùng 1 CSDL và trên cùng 1 server.
- Khoá ngoại chỉ có thể tham bảng chính:
  - Là 1 cột hay 1 phần của khoá chính
  - Là cột có ràng buộc unique
  - Là cột có chỉ mục unique
- Một bảng có thể có tối đa 253 khoá ngoại và có thể tham chiếu đến 253 bảng khác nhau.



Bảo toàn tham chiếu

Ràng buộc Foreign key

Định nghĩa FOREIGN KEY CONSTRAIT khi tạo bảng

CREATE TABLE TableName
(columnName datatype [,…],
[CONSTRAINT constraintName]
FOREIGN KEY[(column[,..n])]
REFERENCES *ref_table* [**(** r*ef_column [,..n])])**]
[ON DELETE { CASCADE | NO ACTION }]
[ON UPDATE { CASCADE | NO ACTION }]
[NOT FOR REPLICATION]

Bảo toàn tham chiếu

Ràng buộc Foreign key

ON UPDATE|DELETE {CASCADE | NO ACTION}

- Xác định hành động cần phải thực hiện cho 1 hàng trong bảng đang tạo nếu hàng đó có quan hệ tham chiếu và hàng tham chiếu bị xoá khỏi bảng chính. Mặc định là NO ACTION.
- CASCADE: dùng để xác định là hàng sẽ bị cập nhật/xoá khỏi bảng tham chiếu nếu hàng đó bị cập nhật/xóa khỏi bảng chính
- NO ACTION: SQL Server sẽ đưa ra thông báo lỗi và việc xoá hàng trên bảng chính sẽ bị từ chối.

Bảo toàn tham chiếu

- Ví dụ:

Create table **Phongban**
(Mapb int,
Tenpb varchar(30),
Constraint pb_PK primary key (mapb)
)
Create table **nhanvien**
(manv int, Hoten varchar(40), Mapb int,
Constraint manv_PK primary key (manv),
Contraint mapb_Fk foreign key(mapb) references phongban(mapb)
)

Bảo toàn tham chiếu

Ràng buộc Foreign key

- **Ví dụ 1**

CREATE TABLE VITRI
(MaVt int Primary key, DiaChi varchar(40))

CREATE TABLE PhongBan
(Mapb int primary key,
TenPb varchar(30),
MaVT int REFERENCES VITRI(MaVt)
)

Bảo toàn tham chiếu

Ràng buộc Foreign key

- **Ví dụ 2**

```
CREATE TABLE NHANVIEN (
    manv   CHAR(9) NOT NULL,
    honv VARCHAR(15) NOT NULL,
    tennv VARCHAR(15) NOT NULL,
    ngsinh DATETIME, diachi VARCHAR(30),
    phai CHAR(1), ma_nql CHAR(9),
    phg INT  NOT NULL,
    CONSTRAINT Nv_PK PRIMARY KEY (manv),
    CONSTRAINT Nv_fk FOREIGN KEY (phg)
    REFERENCES PHONGBAN(mapb))
```


Bảo toàn tham chiếu

Ràng buộc Check

- Ví dụ 4:

```
CREATE TABLE PHANCONG(
    ma_nvien CHAR(9) NOT NULL,
    soda INT NOT NULL,
    thoigian DECIMAL(3,1) NOT NULL,
    PRIMARY KEY (ma_nvien, soda),
    FOREIGN KEY (ma_nvien) REFERENCES
NHANVIEN),
    FOREIGN KEY (soda) REFERENCES DEAN(mada),
    CHECK (thoigian ≥ 0))
```


Bảo toàn tham chiếu

Ví dụ:

```
    Create table nhanvien
(
        manv int, Hoten varchar(40),
        Mapb int,
        Constraint manv_PK primary key (manv),
        Contraint mapb_Fk foreign key(mapb)
        references phongban(mapb) on delete set
        default on update cascade
    );
```


Sửa cấu trúc bảng

Cú pháp

```
ALTER TABLE <table_name>
{ALTER COLUMN <column_name> <new_data_type>}
| {ADD [<column_name> <data_type>]}
| {DROP COLUMN <column_name>}
```

Hiệu chỉnh cấu trúc bảng

- **Thêm thuộc tính (thêm cột)**

Cú pháp:

```
ALTER TABLE <Tên bảng>
ADD <tên cột> <kiểu dữ liệu> [NOT NULL]
[CONSTRAINT...]
```

Ví dụ: thêm cột

```
ALTER TABLE SanPham
ADD NgayNhap SmallDateTime
```


Hiệu chỉnh cấu trúc bảng

- **Xóa thuộc tính (xóa cột)**

Cú pháp:

```
ALTER TABLE <Tên bảng>
DROP COLUMN <tên cột> [CONSTRAINT…]
```

```
ALTER TABLE Sanpham
DROP COLUMN NgayNhap
```


Hiệu chỉnh cấu trúc bảng

- **Thay đổi kiểu dữ liệu của thuộc tính**

Cú pháp:

```
ALTER TABLE <Tên bảng>
ALTER COLUMN <Thuộc tính> <Kiểu dữ liệu>
[CONSTRAINT...]
```

Ví dụ: sửa kiểu dữ liệu cho cột

```
ALTER TABLE SanPham
ALTER COLUMN NgayNhap DateTime NOT NULL
```


Hiệu chỉnh cấu trúc bảng

- **Thêm ràng buộc**

Cú pháp:

```
ALTER TABLE <Tên bảng>
ADD CONSTRAINT…
```

Hiệu chỉnh cấu trúc bảng

- **Thêm ràng buộc**

Ví dụ 1:

```
CREATE TABLE HoaDon
(MaHD int, LoaiHD Char(1) DEFAULT 'X', NgayLap DateTime NOT NULL)
```

```
ALTER TABLE HoaDon
ADD DEFAULT Getdate() FOR NgayLap
```

Hay

```
ALTER TABLE HoaDon
ADD CONSTRAINT Ngay_DF DEFAULT
Getdate() FOR NgayLap
```


Hiệu chỉnh cấu trúc bảng

- **Thêm ràng buộc**

Ví dụ 2:

```
CREATE TABLE NhanVien
(MaNV char(4) CHECK (Manv LIKE '[0-9][0-9][0- 9][0-9]', Hoten Varchar(40), LCB int CHECK (LCB BETWEEN 0 AND 50000, HSPC real, Thanhpho varchar(10) CONSTRAINT chkCity CHECK(Thanhpho IN ('Berkeley', 'Boston', 'Chicago', ' Dallas'))
```

- **Hay**

```
ALTER TABLE Nhanvien
    ADD CONSTRAINT NV_HSPC
        CHECK (HSPC>=0.1 AND HSPC<0.5)
```


Hiệu chỉnh cấu trúc bảng

- **Thêm ràng buộc**

Ví dụ 2:

```
CREATE TABLE NhanVien
(MaNV char(4) CHECK (Manv LIKE '[0-9][0-9][0- 9][0-9]', Hoten Varchar(40), LCB int CHECK (LCB BETWEEN 0 AND 50000, HSPC real, Thanhpho varchar(10) CONSTRAINT chkCity CHECK(Thanhpho IN ('Berkeley', 'Boston', 'Chicago', ' Dallas'))
```

- **Hay**

```
ALTER TABLE Nhanvien
    ADD CONSTRAINT NV_HSPC
        CHECK (HSPC>=0.1 AND HSPC<0.5)
```


Hiệu chỉnh cấu trúc bảng

- **Xóa ràng buộc**

Cú pháp:

```
ALTER TABLE <Tên bảng>
    DROP CONSTRAINT Constraint_name
```

Ví dụ

```
ALTER TABLE Table3
        DROP CONSTRAINT Table3_PK
```

Xóa bảng

- Xóa bảng và toàn bộ dữ liệu trong bảng

Cú pháp:

```
DROP TABLE <Tên bảng>
```

Ví dụ:

DROP TABLE DEPARTMENT

Xóa bảng (Truncate table)

- Lệnh TRUNCATE TABLE xóa tất cả các dòng trong bảng đồng thời giải phóng không gian lưu trữ bảng.

```
TRUNCATE TABLE Tên bảng
```


Ngôn ngữ thao tác dữ liệu

- **Thêm một record mới vào bảng**:

Cú pháp:

```
INSERT INTO tablename
VALUES ('value1', 'value2',. . . ,'valueN')
```

– Ví dụ:

Insert into sinhvien values (’01’,’Le van A','‘CDTH1A')

Ngôn ngữ thao tác dữ liệu

- **Cập nhật dữ liệu trong bảng**:

– Cú pháp:

```
UPDATE table name
SET attribute value=new value
WHERE condition
```

– Ví dụ:

```
update SinhVien
set MaLop=‘CDTH1A'
where MaSV=‘A01'
```

Ngôn ngữ thao tác dữ liệu

- **Xóa các record trong bảng:**
  - Cú pháp:

DELETE FROM table name
WHERE condition

- Ví dụ:
  - Delete from Sinhvien
    Where Malop='CDTH1A'

Xem Tables

Cú pháp: Xem thông tin Table

sp_help <table_name>

Sp_help cthoadon

Cú pháp: Xem dữ liệu Table

SELECT <select_list> FROM <table_name>

Select * from cthoadon

Thủ tục lưu trữ hệ thống

sp_help- System stored procedure

- Để kiểm tra xem bảng đã được tạo hay chưa?

sp_help table_name

- Để kiểm tra xem kiểu dữ liệu của người dùng đã được tạo hay chưa?

sp_help datatype_name

Xem Constraints

- **Viewing Constraints**
  - Sp_helpConstraint Events
- **Verify constraints by inserting data**

	EventID	EventType	EventTitle	EventDescription	EventLanguage
1	1	Party	NULL	NULL	NULL

EventDate	EventEndDate	EventCreator
2004-02-15 01:28:00	2004-02-16 01:28:00	sa

ES

  - SELECT * FROM Events

Bài tập

Example

a) Tạo Table có khóa chính

CREATE TABLE KhachHang

(Makh char(5), Tenkh Varchar(40), DiaChi Varchar(50), DienThoai Nvarchar(10) CONSTRAINT Makh_pk Primary key(Makh))

b) Tạo Table có khóa ngoại

CREATE TABLE HoaDon

(Mahd Char(5), NgayLap Datetime, Makh Char(5) CONSTRAINT Mahd_pk Primary key(Mahd)

CONSTRAINT Makh_fk Foreign key References KhachHang (Makh))

Modifyling Table_Defining Constraints

Example

a) ALTER TABLE Sanpham
ADD CONSTRAINT Masp_pk Primary key(Masp)

b) ALTER TABLE ChiTietHoaDon
ADD CONSTRAINT Masp_Mahd_pk Primary key(Mahd,Masp)

c) ALTER TABLE ChiTietHoaDon
ADD CONSTRAINT Masp_fk Foregin key (Masp) References Sanpham(Masp)

d) ALTER TABLE ChiTietHoaDon
ADD CONSTRAINT Mahd_fk Foregin key(Mahd) References HoaDon(Mahd)

Bài 7

TRUY XUẤT CSDL

Ngôn ngữ thao tác dữ liệu – DML và DCL

Accessing and Modifying Data

1. Truy xuất dữ liệu – Lệnh Select
   - Truy vấn đơn giản
   - Truy vấn từ nhiều bảng
   - Truy vấn con - SubQuery
   - Các mệnh đề EXISTS, DISTINCT, COMPUTE BY
2. Sửa chữa dữ liệu
   - Lệnh INSERT
   - Lệnh UPDATE
   - Lệnh DELETE

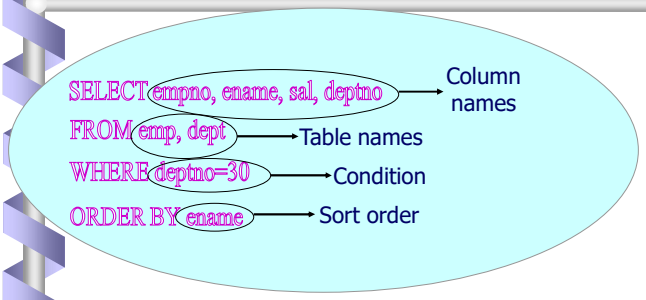
Cú pháp lệnh Select

SELECT [ALL | DISTINCT] [TOP n [WITH TIES]] select_list
[INTO new_table]
FROM table_source
[WHERE search_condition]
[GROUP BY group_by_expression]
[HAVING search_condition]
[ORDER BY order_expression [ASC | DESC]]

- *ORDER BY : Sắp xếp*
- *WHERE: Điều kiện*
- *GROUP BY: Nhóm*
- *HAVING: Điều kiện nhóm*

77

Ví dụ lệnh SELECT



78

Truy vấn đơn giản

- Chọn tất cả các cột trong một bảng

Syntax

```
SELECT * FROM < tablename>
```

Example

```
SELECT * FROM [Khach Hang]
```


79

Truy vấn đơn giản

- Chọn một vài cột trong một bảng

Syntax

```
SELECT <column1>,<column2>
FROM <tablename>
```

Example

```
SELECT Masp, Tensp  FROM [San Pham]
```


80

Truy vấn đơn giản

- Kết nối các cột thành một cột

Syntax

```
SELECT <column1>+<'constant' >
FROM <tablename>
```

Example

```
SELECT HoNV+ ‘ ‘ + TenNv
FROM [Nhan vien]
```


81

Truy vấn đơn giản

- Đặt tên cho cột mới

Syntax

```
SELECT <column1> as <'alias'>
FROM <tablename>
```

Example

```
SELECT Honv +’ ‘ +Tennv AS ‘HOTEN' FROM
[Nhan vien]
```


82

Truy vấn đơn giản

- Tạo cột tính toán

Example

```
SELECT Mahd, Soluong*Dongia AS ‘Tong Tien'
FROM [Chi Tiet Hoa Don]
```


83

Truy vấn đơn giản

- Loại bỏ những dòng trùng nhau

Syntax

```
SELECT DISTINCT <column1>
FROM <tablename>
```

Example

```
SELECT DISTINCT Makh
FROM [hoa don]
```

```
SELECT diadiem
FROM   DIADIEM_PHG
```

diadiem
TP HCM
HA NOI
TP HCM

```
SELECT   DISTINCT diadiem
FROM     DIADIEM_PHG
```

diadiem
TP HCM
HA NOI

84

Truy vấn đơn giản

- Chỉ có n hàng đầu tiên hay n% của các hàng của bảng kết quả được xuất

Syntax

SELECT TOP n [PERCENT] <column name>
FROM <tablename>

Examples

SELECT TOP 3 masp, dongia FROM [Chi tiet hoa don]
SELECT TOP 4 PERCENT masp FROM [san pham]

Truy vấn đơn giản

- Ví dụ: liệt kê 3 hóa đơn có cước phí cao nhất

Select top 3 **with ties** OrderID, Freight
From Orders
Order by Freight DESC

	OrderID	Freight
1	10540	1007.64
2	10372	890.78
3	10267	830.75

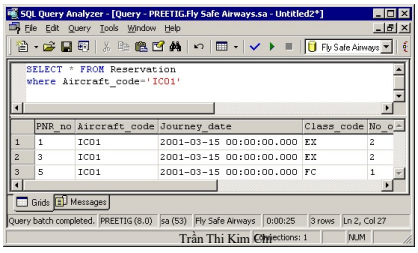
Không WITH TIES

	OrderID	Freight
1	10540	1007.64
2	10372	890.78
3	10267	830.75
4	11030	830.75

Có WITH TIES

Mệnh đề WHERE

- Chứa điều kiện lọc dữ liệu cần trả về
- Cú pháp: WHERE <search_condition>



Phép toán quan hệ - Relational Operators

Operator	Meaning
=	Equal To
>	Greater Than
<	Less than
>=	Greater Than or Equal To
<=	Less Than or Equal To
!	Not

Phép toán Logical

- Phép toán và AND

Example

```
SELECT Mahd, NgayLapHD, Makh
FROM [Hoa don]
WHERE Month(NgayLapHD) = 3 AND
Year(NgayLapHD)=1992
```


89

Phép toán Logical

- OR operator

Example

```
SELECT * FROM [Hoa don]
WHERE Makh = 'FISC' OR Makh = 'HUNSAN'
```


90

Logical Operators (contd.)

- NOT operator

Example

```
SELECT * FROM [Hoa don]
WHERE NOT Manv= 10
```


91

Các toán tử SQL

- LIKE: giống 1 chuỗi
- IS NOT NULL: không phải giá trị rỗng
- BETWEEN…AND: giữa 2 giá trị
- IN: đạt giá trị trong 1 danh sách
- ALL/ ANY (SOME): được dùng trong lệnh truy vấn con và kết quả là nhiều dòng.
  - <ANY: Nhỏ hơn trị cao nhất
  - >ANY: Lớn hơn trị thấp nhất
  - =ANY: Tương đương với IN
  - >ALL: Lớn hơn trị cao nhất
  - <ALL: Nhỏ hơn trị thấp nhất
- EXIST: kiểm tra sự tồn tại của dữ liệu trong nhiều dòng.

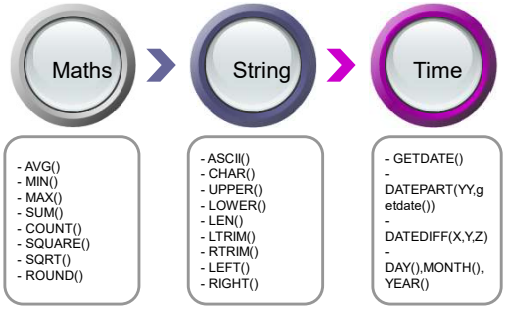
92

Wildcard Characters

Wildcard	Description	Example
_	Represents a single character	SELECT Meal_Code FROM Meal WHERE Meal_Code LIKE 'C_'
%	Represents a string of any length	SELECT Meal_Code FROM Meal WHERE Meal_Code LIKE 'CO_%'
[]	Represents a single character within the range enclosed in the brackets	SELECT * FROM flight WHERE aircraft_code LIKE '9W0[1-2]'
[^]	Represents any single character not within the range enclosed in the brackets	SELECT * FROM flight WHERE aircraft_code LIKE '9W0[^1-2]'

93

Các Hàm - Functions



94

Các Hàm - Functions

	Function	Description
General Functions	ISDATE(exp)	Returns 1 if exp is a valid date
	ISNULL(exp1,exp2)	Returns Null if exp1 is NULL, otherwise exp1 returned
	ISNUMERIC(exp)	Returns 1 if exp is a number type
String Functions	ASCII(char)	Returns the ASCII value of a Character.
	CHAR(int)	Returns the character value for an ASCII integer value.
	CHARINDEX(string1, string2, start)	Returns the starting position for string1 in string2 optionally starting at position start.

95

Các Hàm - Functions

	Function	Description
String Functions	NCHAR(int)	Returns the UNICODE character represented by int.
	LEN(string)	Returns the length of the string.
	LOWER(string)	Returns the string passed in with all characters converted to lowercase.
	UPPER(string)	Returns the string passed in with all characters converted to uppercase.

96

Các Hàm - Functions

	Function	Description
String Functions	REVERSE(string)	Returns the reverse of a character expression.
	RIGHT(string, int)	Returns the int number of characters from the right side of the string.
	LEFT(string, int)	Returns the first int characters from String.

97

Các Hàm - Functions

	Function	Description
String Functions	SUBSTRING(string, start, int)	Returns a portion of the string string starting at position start and continuing for int characters.
	RTRIM(string)	Returns the string with all blank spaces from the end of the string Removed.
	LTRIM(string)	Returns the string with all blank spaces from the left side of the string removed.

98

Các Hàm - Functions

	Function	Description
String Functions	SPACE(int)	Returns int number of spaces.
	STR(float, length, decimal)	Converts a numeric value to a string.
	STUFF(string, start, length, char)	Removes length characters from string starting with character start and replaces them with char.

99

Các Hàm - Functions

	Function	Description
String Functions	UNICODE(Unicode string)	Returns the numeric value of the first character of a UNICODE Expression.

100

Các Hàm - Functions

	Function	Description
Date and Time Functions	DATEPART(day/ month/..,day)	Returns the specific part of the date as an integer.
	DAY(date)	Returns the numeric day of the week for date.

101

Các Hàm - Functions

	Function	Description
	GETDATE()	**GETDATE() Returns the current server date and *time*.**
	MONTH(*date*)	**Returns the numeric month number of *date*.**
	YEAR (*date*)	**Returns the numeric year number of *date*.**

102

So sánh Chuỗi

- So sánh gần đúng sử dụng "like"
  - Hai ký tự thay thế: '_' và '%'
- Tìm tất cả các mã nhân viên có địa chỉ ở quận 1.

```
SELECT *
FROM [Nhan vien]
WHERE diachi LIKE '%Q1'
```

Cho biết tên nhân viên sinh vào những năm 1960

```
SELECT Honv, Tennv, NgaySinh
FROM [Nhan vien]
WHERE  convert(char(8), NgaySinh) like '_____6_'
```


103

Các thuộc tính Trùng tên

tuple variable

Biến bộ

- Cho biết hai nhân viên có cùng thành phố

```
SELECT    NV1.TenNv,
NV2.TenNv,NV1.ThanhPho,NV2.Thanhpho
FROM      [Nhan vien] NV1, [Nhan vien] NV2
WHERE     NV1.Thanhpho=NV2.Thanhpho
```

- Có thể sử dụng biến bộ bất kỳ lúc nào để thuận tiện và dễ đọc!

```
SELECT    NV1.TenNv,
NV2.TenNv,NV1.ThanhPho,NV2.Thanhpho
FROM      [Nhan vien] NV1, [Nhan vien] NV2
WHERE     NV1.Thanhpho=NV2.Thanhpho
```


104

Truy vấn từ nhiều bảng & Where (điều kiện kết nối)

- Danh sách các hoá đơn gồm Mahd, tenkh

```
SELECT   Mahd, [Hoa Don].Makh, Tenkh
FROM     [Hoa don], [Khach hang]
WHERE    [Hoa don].makh= [khach hang].Makh
```

- Danh sách các hoá đơn do nhan viên có tên bắt đầu là D lập

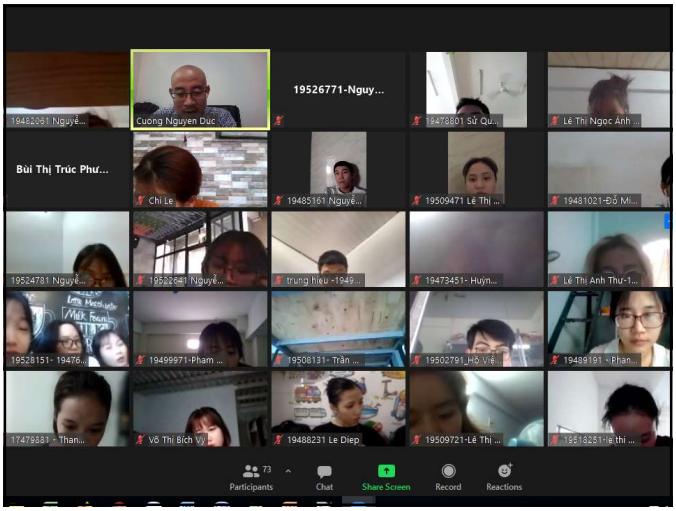
```
SELECT   [Hoa don].Mahd, Honv +' '+Tennv as Hoten
FROM     [Nhan vien], [Hoa don]
WHERE    [Nhan vien].manv = [Hoa don].Manv And TenNV like
         'D%'
```

- Danh sách các hoá đơn do nhan viên có Thành phố là HCM lập

```
SELECT   O.Mahd, Honv +' '+Tennv as HoTen
FROM     [Nhan vien]  E, [Hoa don]  O
WHERE    E.Manv =O.Manv And Thanhpho ='HCM'
```

105

105



106

Sắp xếp - ORDER BY Clause

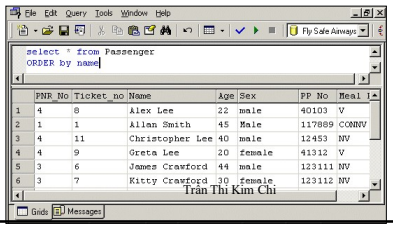
Xác định thứ tự của bộ kết quả

Cú pháp

[ORDER BY { *order_by_expression* [ASC | DESC] } [,...*n*]]

ASC (ascending) : xếp theo thứ tự tăng

DESC (descending): xếp theo thứ tự giảm



107

107

Nhóm dữ liệu trong bảng kết quả

- Những mệnh đề dùng để nhóm dữ trong bảng kết quả:
  - **GROUP BY**: tổng hợp bảng kết quả theo nhóm bằng cách dùng các hàm gộp
  - **COMPUTE** và **COMPUTE BY**: mệnh đề COMPUTE trong lệnh SELECT được dùng để phát ra các hàng tổng hợp bằng cách dùng hàm gộp. Mệnh đề COMPUTE BY được dùng để tổng hợp thêm các hàng kết quả theo cột

108

108

Mệnh đề GROUP BY

- Cú pháp:

  [GROUP BY [ALL]*group_by_expression* [,...*n*] ALL

  bảng kết quả sẽ chứa tất cả các nhóm kể cả những nhóm không thỏa mãn điều kiện lọc trong trong mệnh đề WHERE, những nhóm không thoả điều kiện sẽ có giá trị null.
- *group_by_expression:* biểu thức dùng để xác định cột được nhóm

109

Các Hàm tập hợp

Tên Hàm	Ý nghĩa
SUM	Tính tổng các số
COUNT	Đếm số phần tử
AVG	Tính giá trị trung bình
MIN	Trả về giá trị nhỏ nhất
MAX	Trả về giá trị lớn nhất

110

Mệnh đề GROUP BY

- Ví dụ:

```
SELECT Mahd, SUM(Soluong* Dongia)
AS 'Thanh tien' FROM [Chi tiet hoa don]
GROUP BY mahd
```

```
SELECT Mahd, AVG(Soluong * DonGia)
AS 'Trung Binh' FROM [Chi Tiet hoa don]
GROUP BY Mahd
```


111

Mệnh đề GROUP BY

- Ví dụ:

```
SELECT Mahd, MIN(Soluong * Dongia)
AS 'Thanh tien nho nhat' FROM [Chi Tiet hoa don]
GROUP BY Mahd
```

```
SELECT Mahd, MAX(Soluong * Dongia)
AS 'Thanh Tien Lon Nhat' FROM [Chi Tiet Hoa Don]
GROUP BY Mahd
```


112

Mệnh đề GROUP BY

- Ví dụ:

```
SELECT Count(Mahd)
AS 'So Hoa Don' FROM [Hoa don]
```

```
SELECT Count(*)
AS 'So Hoa Don' FROM [Hoa Don]
```

```
SELECT Makh, Count(Makh) –count(mahd)
AS 'So HD cua tung khach hang' FROM [Hoa don]
GROUP BY Makh
```


Mệnh đề GROUP BY

- Ví dụ:

```
SELECT Masp, Sum(Soluong) As Total
FROM [Chi Tiet Hoa Don]
WHERE Masp=2
GROUP BY Masp
```

```
SELECT Makh, Count(Mahd)
AS 'So HD cua khach hang' FROM [Hoa don]
WHERE Makh like '%o'
GROUP BY Makh
```


GROUP BY và HAVING

- Có thể hạn chế các nhóm trong bảng kết quả bằng mệnh đề HAVING.
- Chỉ sau khi dữ liệu đã được nhóm và tổng hợp, điều kiện trong mệnh đề HAVING mới được áp dụng.
- Không thể dùng 1 cột mà nó không tham gia vào hàm gộp của mệnh đề SELECT hay của mệnh đề GROUP BY.
- **SELECT Masp, AVG(Dongia) FROM [San pham] GROUP BY Masp HAVING (AVG(Dongia) > 10)**

Sử dụng WHERE và HAVING

- Mệnh đề HAVING giống như mệnh đề WHERE nhưng chỉ áp dụng cho cả nhóm trong khi mệnh đề WHERE áp dụng cho từng hàng.
- Một truy vấn có thể chứa cả mệnh đề WHERE và mệnh đề HAVING.
  - Mệnh đề WHERE được áp dụng trước cho các hàng trong bảng được truy vấn. Chỉ những hàng nào thoả mãn điều kiện của mệnh đề WHERE mới được nhóm dữ liệu.
  - Sau đó mệnh đề HAVING sẽ được áp dụng cho các nhóm. Chỉ những nhóm thoả mãn điều kiện HAVING mới được xuất ra bảng kết quả.

Sử dụng WHERE và HAVING

Ví dụ 1

```
SELECT Masp, Sum(Soluong) As Total
FROM [Chi tiet hoa don]
GROUP BY masp
HAVING Sum(soluong)>=30
```

```
SELECT Makh, Count(Mahd)
AS ‘So hoa don cua kH‘ FROM [Hoa don]
GROUP BY Makh
HAVING Count(Mahd)<=5
```


Mệnh đề COMPUTE

Thường dùng để kiểm tra số liệu, dùng kèm với các hàm thống kê SUM, AVG, MAX, MIN,…

Ví dụ

```
SELECT Masp, Mahd, Soluong
FROM [Chi Tiet Hoa Don]
ORDER BY Masp, Mahd
COMPUTE Sum(Soluong)
```

```
SELECT Masp, Mahd, Soluong As Total
FROM [Chi Tiet Hoa Don]
ORDER BY Masp, Mahd
COMPUTE SUM(Soluong) By Masp
COMPUTE SUM(Soluong)
```


Mệnh đề COMPUTE

- **COMPUTE…BY…: Có kết nhóm**

1) SELECT c.Makh, o.Mahd, (od.Soluong * od.Dongia) as 'total' FROM [Hoa don] o, [Chi Tiet hoa don] od,[Khach hang] c
   WHERE c.Makh = o.Makh AND o.Mahd = od.Mahd AND c.Makh LIKE ‘T%‘ ORDER BY c.Makh COMPUTE SUM(od.Soluong * od.Dongia)
2) SELECT c.Makh, o.Mahd, (od.Soluong * od.Dongia) as ‘Tong‘ FROM [Hoa don] o, [Chi Tiet Hoa Don] od, [Khach hang] c
   WHERE c.Makh = o.Makh AND o.Mahd = od.Mahd AND c.Makh LIKE ‘T%‘ ORDER BY c.Makh COMPUTE SUM(od.soluong * od.dongia) BY c.Makh

Mệnh đề COMPUTE

- Khi sử dụng mệnh đề COMPUTE ... BY cần tuân theo các qui tắc dưới đây:
  - Từ khóa DISTINCT không cho phép sử dụng với các hàm gộp dòng
  - Hàm COUNT(*) không được sử dụng trong COMPUTE.
  - Sau COMPUTE có thể sử dụng nhiều hàm gộp, khi đó các hàm phải phân cách nhau bởi dấu phẩy.
  - Các cột sử dụng trong các hàm gộp xuất hiện trong mệnh đề COMPUTE phải có mặt trong danh sách chọn.
  - Không sử dụng SELECT INTO trong một câu lệnh SELECT có sử dụng COMPUTE.
  - Nếu sử dụng mệnh đề COMPUTE ... BY thì cũng phải sử dụng mệnh đề ORDER BY. Các cột liệt kê trong COMPUTE … BY phải giống hệt hay là một tập con của những gì được liệt kê sau ORDER BY. Chúng phải có cùng thứ tự từ trái qua phải, bắt đầu với cùng một biểu thức và không bỏ qua bất kỳ một biểu thức nào.

Mệnh đề JOIN-Kết nối nhiều bảng

- Mệnh đề join dùng để kết nối dữ liệu từ nhiều hơn 1 bảng
- Cú pháp

SELECT column_name [,n..]
FROM table_name table_alias
[CROSS|INNER|[LEFT | RIGHT]OUTER] JOIN table_name table_alias
[ON table_name.ref_column_name join_operator table_name.ref_column_name]
[WHERE search_condition]

121

Kết nối các bảng

- Kết nối chỉ tồn tại trong thời gian truy vấn.
- Kết nối không thay đổi dữ liệu trong các bảng của cơ sở dữ liệu.
- Nên tạo bí danh (alias) cho tên bảng để tránh gõ tên dài và làm truy vấn dễ đọc hơn
- Ví dụ

```
SELECT t.Mahd, NgaylapHD, Masp, Soluong, dongia,
   Thanhtien as Soluong*Dongia
from [Chi tiet hoa don] t
JOIN [Hoa don] h on t.mahd=h.mahd
WHERE Makh='SJC'
```


122

Các cột tham gia kết nối

- Nếu kết nối nhiều hơn 2 bảng thì kết nối 2 bảng trước, sau đó kết nối nhóm này với bảng thứ ba.
- Ví dụ

```
SELECT o.Mahd,c.makh, p.Masp, Tensp,
Soluong, o.dongia, Thanhtien =soluong*o.dongia
FROM [Chi tiet Hoa don] o JOIN SanPham p
ON o.Masp = p.Masp
JOIN [Hoa don] c
ON o.Mahd = c.Mahd
```


123

Các loại kết nối

- Inner Join
- Outer Join
- Cross Join
- Equi Join
- Natural Join
- Self Join

124

Kết nối nội - Inner joins

- Trong kết nối nội, dữ liệu từ nhiều bảng được hiển thị sau khi so sánh giá trị trong 1 cột chung. Chỉ những hàng mà có giá trị thoả mãn điều kiện kết nối trong cột chung đó mới được hiển thị.
- Tích Cartesian: việc kết nối nhiều bảng mà không có điều kiện kết nối trong mệnh đề ON sẽ tạo ra tích cartesian giữa 2 bảng
- Ví dụ:

```
SELECT t.Mahd, h.NgaylapHD, masp, Soluong, dongia,
   Thanhtien as Soluong*Dongia from [Chi tiet hoa don] t
JOIN [Hoa don] h on t.mahd=h.mahd
```


125

Kết nối nội - Inner joins

```
SELECT K.Makh, TenKH,
       Mahd, NgaylapHD
       FROM [Khach hang] K INNER JOIN [Hoa don] h
       ON K.Makh = H.Makh

SELECT Tensp, c.Mahd, Soluong, dongia,
       Soluong * DonGia AS [Thanhtien]
       FROM [San pham] AS s INNER JOIN [Chi tiet hoa
       don] AS c ON s.Masp = c.Masp
       INNER JOIN [Hoa don] As h
       ON h.mahd =c.mahd
       WHERE Month(NgayLapHD) = 3
```


126

Kết nối nội với toán tử lớn hơn

- Có thể thực hiện kết nối 2 bảng với điều kiện kết nối dùng toán tử không bằng nhau.
- Ví dụ:
- SELECT Tensp, c.Mahd, Soluong, c.dongia,
- Soluong * c.DonGia AS [Thanhtien]
- FROM [San pham] AS s INNER JOIN [Chi tiet hoa don] AS c ON s.Masp > c.Masp

127

Kết nối nội – Dùng where

Bảng DONVI

MADV	TENDV
1	Doi ngoai
2	Hanh chinh
3	Ke toan
4	Kinh doanh

Bảng NHANVIEN

HOTEN	MADV
Thanh	1
Hoa	2
Nam	2
Vinh	1
Hung	5
Phuong	NULL

Câu lệnh:

```
SELECT *
FROM nhanvien,donvi
WHERE nhanvien.madv=donvi.madv
```

có kết quả là:

HOTEN	MADV	MADV	TENDV
Thanh	1	1	Doi ngoai
Hoa	2	2	Hanh chinh
Nam	2	2	Hanh chinh
Vinh	1	1	Doi ngoai

128

Kết nối ngoại - Outer joins

- Kết nối ngoại được dùng để cho ra kết quả chứa tất cả các hàng của 1 bảng và các hàng trùng nhau của bảng còn lại. Những cột mà không có giá trị phù hợp sẽ được hiển thị giá trị NULL.
- ***Cú pháp***

SELECT column_name, column_name [,column_name]
FROM table_name [LEFT | RIGHT] OUTER JOIN table_name
ON table_name.ref_column_name
join_operator table_name.ref_column_name

Kết nối ngoại - Outer joins

SQL cung cấp các loại phép nối ngoài sau đây:

- **Phép nối ngoài trái** (ký hiệu: *=): Phép nối này hiển thị trong kết quả truy vấn tất cả các dòng dữ liệu của bảng nằm bên trái trong điều kiện nối cho dù những dòng này không thoả mãn điều kiện của phép nối
- **Phép nối ngoài phải** (ký hiệu: =*): Phép nối này hiển thị trong kết quả truy vấn tất cả các dòng dữ liệu của bảng nằm bên phải trong điều kiện nối cho dù những dòng này không thoả điều kiện của phép nối.

Kết nối ngoại – Left Outer joins

Bảng DONVI

MADV	TENDV
1	Doi ngoai
2	Hanh chinh
3	Ke toan
4	Kinh doanh

Bảng NHANVIEN

HOTEN	MADV
Thanh	1
Hoa	2
Nam	2
Vinh	1
Hung	5
Phuong	NULL

Nếu thực hiện phép nối ngoài trái giữa bảng NHANVIEN và bảng DONVI:

```
SELECT *
FROM nhanvien,donvi
WHERE nhanvien.madv*=donvi.madv
```

kết quả của câu lệnh sẽ là:

HOTEN	MADV	MADV	TENDV
Thanh	1	1	Doi ngoai
Hoa	2	2	Hanh chinh
Nam	2	2	Hanh chinh
Vinh	1	1	Doi ngoai
Hung	5	NULL	NULL
Phuong	NULL	NULL	NULL

Kết nối trái - LEFT OUTER JOIN

- Tất cả các hàng từ bảng bên trái trong mối kết nối giữa 2 bảng sẽ được hiển thị trong bảng kết quả.
- **Ví dụ:**

SELECT Manv, Hoten, cv, d.macv,tench
FROM Nhanvien n INNER JOIN ChucVu d ON n.cv=d.macv

SELECT Manv, Hoten, cv, d.macv,tench
FROM Nhanvien n LEFT OUTER JOIN ChucVu d ON n.cv=d.macv

Kết nối ngoại – Right Outer joins

Bảng DONVI

MADV	TENDV
1	Doi ngoai
2	Hanh chinh
3	Ke toan
4	Kinh doanh

Bảng NHANVIEN

HOTEN	MADV
Thanh	1
Hoa	2
Nam	2
Vinh	1
Hung	5
Phuong	NULL

Và kết quả của phép nối ngoài phải:

```
select *
from nhanvien,donvi
where nhanvien.madv=*donvi.madv
```

như sau:

HOTEN	MADV	MADV	TENDV
Thanh	1	1	Doi ngoai
Vinh	1	1	Doi ngoai
Hoa	2	2	Hanh chinh
Nam	2	2	Hanh chinh
NULL	NULL	3	Ke toan
NULL	NULL	4	Kinh doanh

133

Kết nối trái - LEFT OUTER JOIN

- Tất cả các hàng từ bảng bên trái trong mối kết nối giữa 2 bảng sẽ được hiển thị trong bảng kết quả.
- **Ví dụ:**

SELECT Manv, Hoten, cv, d.macv,tench
FROM Nhanvien n RIGHT OUTER JOIN ChucVu d ON n.cv=d.macv

SELECT Manv, Hoten, cv, d.macv,tench
FROM Nhanvien n FULL JOIN ChucVu d ON n.cv=d.macv

134

Kết nối trái - FULL OUTER JOIN

Bảng DONVI

MADV	TENDV
1	Doi ngoai
2	Hanh chinh
3	Ke toan
4	Kinh doanh

Bảng NHANVIEN

HOTEN	MADV
Thanh	1
Hoa	2
Nam	2
Vinh	1
Hung	5
Phuong	NULL

Ví dụ 2.33: Với hai bảng NHANVIEN và DONVI như ở trên, câu lệnh

```
SELECT *
FROM nhanvien FULL OUTER JOIN donvi
    ON nhanvien.madv=donvi.madv
```

cho kết quả là:

HOTEN	MADV	MADV	TENDV
Thanh	1	1	Doi ngoai
Hoa	2	2	Hanh chinh
Nam	2	2	Hanh chinh
Vinh	1	1	Doi ngoai
Hung	5	NULL	NULL
Phuong	NULL	NULL	NULL
NULL	NULL	4	Kinh doanh
NULL	NULL	3	Ke toan

135

Cross join

- Cross join trả về mọi tổ hợp có thể có của tất cả các hàng trong các bảng kết nối.
- Cross join không có mệnh đề ON
  - Nếu không mệnh đề WHERE, cross join sẽ tạo ra tích Cartesian
  - Nếu có mệnh đề WHERE, cross join sẽ thực hiện như 1 kết nối nội

136

Kết nối chéo - Cross join

- Ví dụ 1:

SELECT [San pham].Masp, tensp, soluong, c.dongia FROM [San pham] CROSS JOIN [Chi tiet hoa don]

- Ví dụ 2:

SELECT [San pham].Masp, tensp, soluong, dongia FROM [San pham] CROSS JOIN [Chi tiet hoa don] as c WHERE [San pham].masp = c.Masp

Truy vấn tự kết nối (self-join)

- Ví dụ: tìm tất cả các khách hàng mua ít nhất 2 đơn hàng

SELECT t1.Makh, t1.MAHD,t2.MAHD
FROM [Hoa don] t1 JOIN [Hoa don] t2
ON t1.Mahd<> t2.Mahd AND
t1.Makh = t2.Makh

$$\pi_{StudID}(\sigma_{CrsCode \neq CrsCode2}(TRANSCRIPT \triangleright\triangleleft TRANSCRIPT[StudID, CrsCode2, Semester2, Grade2]))$$

Truy vấn select - merge

A new feature in SQL2008 is merge statement. You can merge 2 or more tables

```
SELECT ltr.FName, lt.LectureDay, lt.LectureTime
FROM dbo.LECTURE lt
inner merge JOIN dbo.LECTURER ltr
on lt.StaffNO=ltr.StaffNO
ORDER BY ltr.FName
```

	FName	LectureDay	LectureTime
1	Francisco	Wednesday	2008-04-30 00:00:00.000
2	Patricio	Monday	2008-05-13 00:00:00.000
3	Patricio	Monday	2008-04-15 00:00:00.000
4	Pedro	Tuesday	2008-04-20 00:00:00.000
5	Roland	Friday	2008-09-14 00:00:00.000
6	Sven	Monday	2008-05-23 00:00:00.000
7	Sven	Wednesday	2008-07-13 00:00:00.000
8	Victoria	tuesday	2008-03-25 00:00:00.000
9	Yang	Friday	2008-02-15 00:00:00.000

Truy vấn với giá trị Null

Phép nối và các giá trị NULL
Nếu trong các cột của các bảng tham gia vào điều kiện của phép nối có các giá trị NULL thì các giá trị NULL được xem như là không bằng nhau.
Ví dụ: Giả sử ta có hai bảng TABLE1 và TABLE2 như sau:

TABLE1

A	B
1	b1
NULL	b2
4	b3

TABLE2

C	D
NULL	d1
4	d2

Câu lệnh:

```
SELECT *
FROM table1, table2
WHERE A *= C
```

Có kết quả là:

A	B	C	D
1	b1	NULL	NULL
NULL	b2	NULL	NULL
4	b3	4	d2

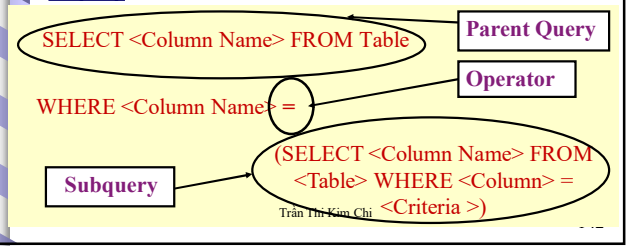
Truy vấn con - Subqueries

- Subquery là lệnh SELECT mà kết quả trả về là 1 giá trị đơn (single value) và được đặt lồng vào bên trong các lệnh SELECT, INSERT, UPDATE, hay DELETE, hay bên trong truy vấn con khác.
- Subquery có thể được dùng bất kỳ nơi nào mà biểu thức được phép dùng

Truy vấn con - Subqueries

A subquery is a SELECT statement inside another SELECT statement.

Cú pháp



Truy vấn con - Subqueries

Khi sử dụng truy vấn con cần lưu ý một số quy tắc sau:

- Một truy vấn con phải được viết trong cặp dấu ngoặc. Trong hầu hết các trường hợp, một truy vấn con thường phải có kết quả là một cột (tức là chỉ có duy nhất một cột trong danh sách chọn).
- Mệnh đề COMPUTE không được phép sử dụng trong truy vấn con.
- Các tên cột xuất hiện trong truy vấn con có thể là các cột của các bảng trong truy vấn ngoài.
- Một truy vấn con thường được sử dụng làm điều kiện trong mệnh đề WHERE hoặc HAVING của một truy vấn khác.
- Nếu truy vấn con trả về đúng một giá trị, nó có thể sử dụng như là một thành phần bên trong một biểu thức (chẳng hạn xuất hiện trong một phép so sánh bằng)

Truy vấn con - Subqueries

- **TH1: Dùng để lấy các field**

```
SELECT h.Mahd, h.NgayLapHD, (SELECT
    MAX(C.Dongia)
    FROM [Chi tiet hoa don] AS c
        WHERE h.Mahd =c.Mahd) AS MaxUnitPrice
    FROM [Hoa don] AS h
```

- Subquery có thể được dùng để khôi phục dữ liệu từ nhiều bảng và có thể được dùng như 1 cách khác của join (alternative to a join)

Truy vấn con - Subqueries

- **TH2: Phép so sánh đối với kết quả truy vấn con**

C1: Cho biết những sản phẩm nào có dongia bằng đơn giá của sản phẩm có tên bắt đầu bằng chữ N

```
SELECT Tensp FROM [San pham]
WHERE Dongia > (SELECT Dongia FROM [San pham] WHERE TenSp like 'N%')
```

- Cách 2: dùng join

```
SELECT P.Tensp FROM [San pham] AS p JOIN [San pham] AS P2
ON (P.Dongia >P2.Dongia)
WHERE P2.TenSp = 'Nem'
```


145

Truy vấn con - Subqueries

- **TH2: Phép so sánh đối với kết quả truy vấn con**

C2: **Cho biết những sản phẩm có thên bắt đầu bằng chữ N và dongia bằng đơn giá của sản phẩm khác**

```
SELECT Tensp FROM [San pham]
WHERE TenSp like 'N%' And Dongia > (SELECT Dongia FROM [San pham])
```

- Cách 2: dùng join

```
SELECT P.Tensp FROM [San pham] AS p JOIN [San pham] AS P2
ON (P.Dongia >P2.Dongia)
WHERE P2.TenSp = 'Nem'
```


146

Truy vấn con - Subqueries

- Subquery có thể được dùng theo 1 trong các dạng sau:
  - WHERE *expression* [NOT] IN (*subquery*)
  - WHERE *expression comparison_operator* [ANY | ALL] (*subquery*)
  - WHERE [NOT] EXISTS (*subquery*)

147

Subqueries với toán tử IN

TH3: Kết quả của subquery được dùng với IN (hay với NOT IN) thường là 1 danh sách (list) chứa từ 0 đến nhiều giá trị.

•Ví dụ:

```
SELECT Mahd, NgayLapHD FROM [Hoa don]
WHERE Makh IN (SELECT Makh FROM [Khach hang]
WHERE Diachi like '%Q1')
```

•Dùng cách 2 với mệnh đề join???

148

Subquery với các toán tử so sánh

- Các toán tử so sánh đơn giản (unmodified comparison operator)

  =, < >, >, > =, <, ! >, ! <, or < =

- TH4: Subquery bắt đầu với toán tử so sánh đơn giản và trả về 1 **giá trị đơn (single value)**

  SELECT Masp FROM [San pham] WHERE dongia > (SELECT MIN(Dongia) FROM [Chi tiet hoa don])

149

Subquery trả về một giá trị

- Use Aggregrate Functions in Subquery as Max(), Min(), Avg(),...
- Khi một subquery được sử dụng trong mệnh đề SELECT, nó trả về dữ liệu thay cho một cột.
- Loại subquery này được gọi là scalar query bởi vì nó chỉ trả về một giá trị.

Ví dụ

```
SELECT Tensp, Dongia FROM [San pham]
        WHERE DVT ='kg' and Dongia <
        (SELECT AVG(Dongia) FROM [San pham])
```


150

Subquery trả về nhiều giá trị

- Sử dụng toán tử In, Not in, Any, all…

Ví dụ

1) SELECT Mahd FROM [Chi tiet hoa don]
WHERE Masp
In (SELECT Masp FROM [San Pham]
WHERE DVT ='Kg')

2) SELECT Mahd, Masp, Soluong FROM [Chi tiet hoa don] WHERE Soluong >=All (SELECT Soluong FROM [Chi tiet hoa don])

151

Subquery trả về nhiều giá trị

- Cho biết các sản phẩm có đơn giá cao nhất

```
SELECT      *
FROM        [San Pham]
WHERE       Dongia >= ALL
        (SELECT Dongia FROM [San pham])
```

- Cho biết các sản phẩm có đơn vị tính có chữ a và có đơn giá cao nhất

```
SELECT      *
FROM        [San pham]
WHERE       DVT like '%a%' and
            Dongia >= ALL
        (SELECT Dongia FROM [san pham])
```


152

Subquery với các toán tử so sánh

- Subquery trả về một giá trị

SELECT Tensp, Dongia FROM [San pham]
WHERE Dongia< (SELECT AVG(Dongia) FROM [San pham] WHERE Masp<5)

- Subquery trả về một dãy giá trị

Select Mahd, s.MASP,tensp from [SAN PHAM]s join [Chi tiet hoa don] c on s.MASP=c.MASP
Where s.Masp IN (select Masp
From [Chi Tiet Hoa don]
Where Masp<5)

Subquery với các toán tử so sánh

- Toán tử so sánh phức (Modified Comparison operators) là toán tử so sánh đơn giản có kèm theo các từ All, any hay some

(=toán tử s/sánh+ [All,Any,some])

- Subquery bắt đầu với toán tử so sánh phức sẽ trả về 1 danh sách (list) của 0 hay nhiều gia trị và có thể bao gồm cả mệnh đề GROUP BY hay HAVING.

Subquery với các toán tử so sánh

- >ALL có nghĩa lớn hơn mọi giá trị.
- >ANY có nghĩa lớn hơn ít nhất 1 giá trị

Vd: >ALL (1, 2, 3) lớn hơn 3
>ANY (1, 2, 3) lớn hơn 1

- Ví dụ:

SELECT Manv, Honv,Tennv
FROM [Nhan vien] WHERE Thanhpho<> ALL (SELECT thanhpho FROM [Khach hang])

Subquery với các toán tử so sánh

Select Masp, tensp, dongia from [San pham]
Where dongia>ALL (Select dongia from [San pham]
where tensp like 'B%')

Select Masp, tensp, dongia from [San pham]
Where dongia>ANY (Select dongia from [San pham]
where tensp like 'B%')

Select Masp, tensp, dongia from [San pham]
Where dongia=ANY (Select dongia from [San pham]
where tensp like 'B%'))

Using EXISTS and NOT EXISTS

- Để kiểm tra xem một truy vấn con có trả về dòng kết quả nào hay không.
- Lượng từ EXISTS (tương ứng NOT EXISTS) trả về giá trị True (tương ứng False) nếu kết quả của truy vấn con có ít nhất một dòng (tương ứng không có dòng nào).
- Điều khác biệt của việc sử dụng EXISTS với hai cách đã nêu ở trên là trong danh sách chọn của truy vấn con có thể có nhiều hơn hai cột.

Example

```
SELECT Mahd, Makh FROM [Hoa don]
        WHERE EXISTS
        (SELECT Makh FROM [Khach hang])
```


157

Using EXISTS and NOT EXISTS

- Subquery dùng với toán tử EXISTS

```
Select * from [Khach Hang] as k
      where NOT EXISTS ( SELECT * from [Hoa don] as h
       where k.Makh= h.makh )
Select * from [Khach Hang] as k
      where EXISTS ( SELECT * from [Hoa don] as h
            where k.Makh= h.makh )
where k.Makh= h.makh )
```


158

Using EXISTS and NOT EXISTS

- Cho biết những sản phẩm nào chưa bán được dùng 3 cách left outer join, not in và not exists

1)Select * from [San Pham] s
where NOT EXISTS (SELECT * from [Chi Tiet Hoa don] c
where s.masp=c.masp)

2)Select * from [San Pham]
where Masp NOT IN (SELECT masp from [Chi Tiet Hoa don])

3) Select * from [San Pham] s Left outer join [Chi Tiet Hoa don] c ON s.Masp=c.Masp WHERE c.Masp is null

159

Using EXISTS and NOT EXISTS

- Cho biết những sản phẩm nào có tổng số lượng bán được lớn hơn số lượng trung bình bán ra

1) Select masp, tensp, TongSL=sum(Soluong)
from [San Pham] s, [Chi Tiet Hoa Don] c
where s.masp=c.masp
Group by c.masp, tensp
having sum(soluong)>(Select AVG(soluong) form [Chi tiet hoa don])

160

Nested Subqueries

Nested subqueries are subqueries within other subqueries

Example

```
SELECT Makh, Tenkh FROM [Khach hang]
WHERE Makh in
(SELECT Makh from [Hoa don]
WHERE Mahd in
      (SELECT Mahd FROM [Chi tiet hoa don] where masp>3))
```


161

Correlated Subqueries

- A subquery refers to the parent query
- A subquery is re-evaluated for every iteration of the parent query

Example

```
SELECT Makh,Tenkh, Thanhpho FROM [Khach hang]
WHERE Makh IN
(SELECT Makh FROM [Nhan vien], [hoa don]
WHERE [Hoa don].Manv = [Nhan vien].Manv and [Nhan vien].Thanhpho =[Khach hang].Thanhpho)
```


162

Unions – Phép hợp

Toán tử Union để kết nốt 2 câu lệnh Select

Cú pháp

```
SELECT statement
      UNION [ALL]
SELECT statement
```

Ví dụ

```
SELECT c.Thanhpho FROM [Khach hang] c
UNION
SELECT e.Thanhpho FROM [Nhan vien] e
```


163

Unions – Phép hợp

Khi sử dụng toán tử UNION để thực hiện phép hợp, ta cần chú ý các nguyên tắc sau:

- Danh sách cột trong các truy vấn thành phần phải có cùng số lượng.
- Các cột tương ứng trong tất cả các bảng, hoặc tập con bất kỳ các cột được sử dụng trong bản thân mỗi truy vấn thành phần phải cùng kiểu dữ liệu.
- Các cột tương ứng trong bản thân từng truy vấn thành phần của một câu lệnh UNION phải xuất hiện theo thứ tự như nhau. Nguyên nhân là do phép hợp so sánh các cột từng cột một theo thứ tự được cho trong mỗi truy vấn.
- Khi các kiểu dữ liệu khác nhau được kết hợp với nhau trong câu lệnh UNION, chúng sẽ được chuyển sang kiểu dữ liệu cao hơn (nếu có thể được).
- Tiêu đề cột trong kết quả của phép hợp sẽ là tiêu đề cột được chỉ định trong truy vấn đầu tiên.

164

Unions – Phép hợp

Khi sử dụng toán tử UNION để thực hiện phép hợp, ta cần chú ý các nguyên tắc sau:

- Truy vấn thành phần đầu tiên có thể có INTO để tạo mới một bảng từ kết quả của chính phép hợp.
- Mệnh đề ORDER BY và COMPUTE dùng để sắp xếp kết quả truy vấn hoặc tính toán các giá trị thống kê chỉ được sử dụng ở cuối câu lệnh UNION. Chúng không được sử dụng ở trong bất kỳ truy vấn thành phần nào.
- Mệnh đề GROUP BY và HAVING chỉ có thể được sử dụng trong bản thân từng truy vấn thành phần. Chúng không được phép sử dụng để tác động lên kết quả chung của phép hợp.
- Phép toán UNION có thể được sử dụng bên trong câu lệnh INSERT.
- Phép toán UNION không được sử dụng trong câu lệnh CREATE VIEW.

Lệnh SELECT INTO – Tạo bảng

- Ta có thể tạo table mới dựa vào tập kết quả của câu lệnh Select. Table mới có thể là table tạm hay là một table thực sự trong DB.
- Cú pháp:

```
SELECT *|ColumnNam1, ColumnName2,...
    INTO TableName
    FROM Tables
    WHERE Condition
    ORDER By SortFieldName
    GROUP BY FieldGroupName
```


Lệnh SELECT INTO – Tạo bảng

1) Tạo Table Tạm

```
SELECT Tenkh as Ten, ThanhPho
INTO #Temp_Customer
FROM [Khach hang]
```

Xem kết quả

```
Select * From  #Temp_Customer
```


Lệnh SELECT INTO – Tạo bảng

Ví dụ 2

```
SELECT c.Makh As Name, Mahd, NgayLapHD
INTO Customer_Order
FROM [Khach hang] as c INNER Join [Hoa don] As o
ON c.Makh=o.Makh
WHERE Month(NgayLapHD) =7
```

Lệnh SELECT INTO – Tạo bảng

Ví dụ 3 : Tạo bảng dữ liệu từ DataBase khác

```
USE SalesDB
SELECT CompanyName As Name, Phone
INTO KhachHang
FROM NorthWind.dbo.Customers
```


169

Sửa chữa dữ liệu

- The INSERT Statement
- The UPDATE Statement
- The DELETE Statement
- Modifying Data Using XML

170

Lệnh Insert

- Thêm một hay nhiều dòng dữ liệu vào bảng hay một view.
- Lưu ý:
  - Dữ liệu chèn phải đúng kiểu dữ liệu của cột.
  - Không nhập giá trị Null vào cột Not Null.
  - Tuân thủ đúng các toàn vẹn dữ liệu, bẫy lỗi.
  - Chỉ định danh sách các giá trị ứng với các cột.
  - Cột có thuộc tính Identity thì không cần chỉ định giá trị.
  - Tại một thời điểm chỉ có thể chèn vào một bảng duy nhất.

171

Lệnh INSERT

- Cú pháp 1

  INSERT <TableName> [(Col1, Col2, …)]
  VALUES (Value1, Value2,…)
- Ví dụ 1

  INSERT [San pham](Masp, Tensp)
  VALUES (123, 'Ice Tea')

172

Lệnh Insert

```
Create table t1 (col1 int identity, col2 varchar(30) default('my
column default'), col3 timestamp, col4 varchar(40) null);

insert into t1 (col4) values('Explicit value')
Select * from t1

insert into t1 (col2,col4) values('Explicit value','Explicit value' )
Select * from t1

insert into t1 (col2) values('Explicit value')
Select * from t1

Insert into t1 default values; Select * from t1
```


Lệnh Insert

```
Create table t2(col1 int identity, col2 varchar(30))
Go
insert t2 values('Row #1')
Go
Select * from t2
Go
set identity_insert t2 ON
Go
insert into t2 (col1,col2) values (-99,'Explicit')
Go
select * from t2
```


Lệnh Insert

- Cú pháp 2: tạo 1 bộ kết quả từ một hay nhiều bảng chèn vào 1 bảng khác

  INSERT <TableName> [(Col1, Col2, …)]

  SELECT (Value1, Value2,…)

- Ví dụ 2

```
create table SPMOI (Masp int, tensp nvarchar(20))
INSERT SPMOI(Od.Masp, P.Tensp)
        SELECT OD.Masp, Tensp
        FROM [San pham] as P INNER JOIN [Chi tiet hoa
        don] AS Od ON P. Masp = Od. Masp
Select * from SPMOI
```


Lệnh **UPDATE**

- Thay đổi/cập nhật dữ liệu trong một bảng

  UPDATE ***<table_name>***
  SET ***<column_name>*** **=** ***<expression>***
  [FROM ***<table_list>*****]**
  [WHERE ***<search_condition>*****]**

- Ví dụ

```
UPDATE [Chi tiet hoa don] SET
Thue=Dongia+0.1*Dongia
WHERE Masp<5

UPDATE [Chi Tiet Hoa don]
SET Dongia=
    ( SELECT Dongia+ Dongia*0.2  FROM [San pham]) where
      Masp=2
```

Lệnh DELETE

- Xóa một hay nhiều dòng trong một bảng hay một truy vấn

  DELETE ***<table_name>***
  [FROM ***<table_list>*****]**
  [WHERE ***<search_condition>*****]**

- Ví dụ

  DELETE FROM [Chi Tiet Hoa Don]
  WHERE Mahd =10148

177

Lệnh DELETE

- Xóa dòng trùng nhau (Duplicate row
- Ba cách:
  - Dùng Windowing
  - Dùng khóa thay thế (surrogate key)
  - Dùng lệnh select distant into

178

Lệnh DELETE

Dùng Windowing

- Window là 1 tập các hàng do người dùng xác định.
- Mệnh đề OVER xác định các partition và xếp thứ tự cho các window trước khi áp dụng các chức năng window.

```
Ranking Window Functions
< OVER_CLAUSE > :: =
    OVER ( [ PARTITION BY value_expression , ... [ n ] ]
           <ORDER BY Clause> )

Aggregate Window Functions
< OVER_CLAUSE > :: =
    OVER ( [ PARTITION BY value_expression , ... [ n ] ] )
```


179

Lệnh DELETE

Ví dụ

```
create table T3(col1 int null, col2
char(5) null)
insert T3(col1,col2)
values
(1,'abc'),(2,'abc'),(2,'abc'),(2,'abc'),
(7,'abc'),(7,'abc')
select * from T3
```


180

Lệnh DELETE

Ví dụ

Select col1,col2, ROW_NUMBER() Over(partition by col1,col2 order by col1) as RN from T3

Col1	Col2	rn
1	abc	1
2	abc	1
2	abc	2
2	abc	3
7	xyz	1
7	xyz	2

181

Lệnh DELETE

Ví dụ: xóa hàng trùng nhau bằng Surrogate Key

```
alter table t3
add pk1 int identity not null
        constraint pk_Khoa Primary Key
Select * from T3
```

Result:

Col1	Col2	PK
1	abc	74
2	abc	75
2	abc	76
2	abc	77
7	xyz	78
7	xyz	79

182

Lệnh DELETE

Ví dụ: xóa hàng trùng nhau bằng Select Distant Into

Select distinct col1,col2 into NoDups from T3

183

Lệnh TRUNCATE TABLE

- Lệnh TRUNCATE TABLE dùng để
  - Xóa toàn bộ dữ liệu trong một bảng mà không ghi nhật ký lại
  - Về chức năng hoàn toàn giống như câu lệnh Delete
  - Nhanh hơn câu lệnh delete
  - Không bật các bẫy lỗi trigger

  TRUNCATE TABLE *table_name*
- Ví dụ

  TRUNCATE TABLE NewProducts

184

Merging Data

- Trong SQL Server 2008, có thể thực hiện các thao tác insert, update, hay delete chỉ trong 1 lệnh.
- Lệnh MERGE cho phép kết nối nguồn dữ liệu với bảng/view đích và sau đó thực hiện nhiều action trên bảng đích.

```
MERGE TargetTable
  USING SourceTable
    ON join conditions
  [WHEN Matched
    THEN DML]
  [WHEN NOT MATCHED BY TARGET
    THEN DML]
  [WHEN NOT MATCHED BY SOURCE
    THEN DML]
```


185

Ví dụ Merging Data để thực thi Insert và Update

dbo.Purchases

ProductID	CustomerID	PurchaseDate
707	11794	2006-08-20
707	15160	2006-08-25
708	18529	2006-08-21
711	11794	2006-08-20
711	19585	2006-08-22
712	14680	2006-08-26
712	21524	2006-08-26
712	19072	2006-08-20
870	15160	2006-08-23
870	11927	2006-08-24
870	18749	2006-08-25

dbo.FactBuyingHabits

ProductID	CustomerID	LastPurchaseDate
707	11794	2006-08-14
707	18178	2006-08-18
864	14114	2006-08-18
866	13350	2006-08-18
866	20201	2006-08-15
867	20201	2006-08-14
869	19893	2006-08-15
870	17151	2006-08-18
870	15160	2006-08-17
871	21717	2006-08-17
871	21163	2006-08-15
871	13350	2006-08-15
873	23381	2006-08-15

186

Ví dụ Merging Data để thực thi Insert và Update

- Hai bảng trên có 2 hàng giống nhau:
  - Customer 11794 và Product 707
  - Customer 15160 và Product 870
- Dùng lệnh MERGE để cập nhật dữ liệu cho 2 hàng này trong bảng **FactBuyingHabits** với số lượng đặt mua trong bảng **Purchases** (bằng mệnh đề WHEN MATCHED THEN), và chèn tất cả hàng còn lại của **Purchases** vào bảng **FactBuyingHabits** (bằng mệnh đề WHEN NOT MATCHED THEN)

187

Ví dụ Merging Data để thực thi Insert và Update

```
MERGE dbo.FactBuyingHabits AS Target
USING (SELECT CustomerID, ProductID, PurchaseDate FROM
  dbo.Purchases) AS Source
ON (Target.ProductID = Source.ProductID AND
  Target.CustomerID = Source.CustomerID)
WHEN MATCHED THEN
  UPDATE SET Target.LastPurchaseDate = Source.PurchaseDate
WHEN NOT MATCHED BY TARGET THEN
  INSERT (CustomerID, ProductID, LastPurchaseDate) VALUES
  (Source.CustomerID, Source.ProductID, Source.PurchaseDate)
```


188

Viewing Tables

Syntax: Viewing table information

```
sp_help <table_name>
sp_help  constraint <table_name>
```

Example: Viewing table information

```
Sp_help [Khach Hang]

Sp_help constraint [Khach hang]
```


189

Removing tables

Syntax

```
DROP TABLE <Table_Name>
```

Example

```
DROP TABLE Sanpham
```


190

191

1. Liệt kê các các phòng mà khách hàng thuê chưa trả. Thông tin bao gồm: Mã phòng, mô tả, loại phòng được sắp xếp theo phòng, nếu cùng phòng thì sắp xếp theo Số ngày ở giảm dần
2. Liệt kê các phòng có đơn giá phòng rẻ nhất. Thông tin bao gồm: Mã phòng, mô tả và đơn giá phòng.
3. Liệt kê danh sách các khách hàng có ký tự đầu của mã khách hàng là A đến C.
4. Liệt kê các phòng chưa có khách hàng thuê. Yêu cầu viết bằng 3 cách (1điểm).
5. Thống kê tổng số phòng theo của các loại phòng có mã là 1 và 3. Gồm loại phòng, số lượng phòng.
6. Cho biết khách hàng nào có tổng tiền phải trả trên 2000000.
7. Liệt kê danh sách các khách hàng đã thuê tất cả các phòng ở tầng 1 dãy nhà A.
8. Dùng lệnh Select into tạo bảng HoaDonTinhTien cho những khách hàng đã trả phòng từ tháng 6 đến tháng 12 của năm 2007 gồm MAKH, TENKH, QUOCTICH, DIENTHOAI, TONGTIEN được sắp xếp theo TenKH. Trong đó TONGTIEN được tính bằng (Số ngày ở * Đơn giá) – Giảm giá
9. Xóa các khách hàng đã trả phòng. (Ngày trả nhỏ hơn ngày hiện hành)

192