Chương 3
Ngôn ngữ Thao Tác Dữ Liệu

Ngôn ngữ thao tác dữ liệu

- Truy vấn dữ liệu từ table (SELECT)
- Chèn dữ liệu vào table (INSERT)
- Cập nhật dữ liệu vào table(UPDATE)
- Xóa dữ liệu(DELETE)
- Fulltext Search

Cấu trúc lệnh truy vấn

```
SELECT Select_List
   FROM Table_List
   [WITH (BUFFERING = lExpr)]
   [WHERE Conditions]
   [GROUP BY Column_List]
   [UNION Clause]
   [HAVING Conditions]
   [ORDER BY Column_List]
   [INTO Clause | TO Clause ]
   [Additional_Display_Options]
```

Lệnh select

- **Câu truy vấn cơ bản**
- Ví dụ:

```
SELECT *
FROM Orders

SELECT OrderID, OrderDate, CustomerID
FROM Orders
```

Lệnh select

- Truy vấn loại bỏ các dòng bị trùng:
  - **Cú pháp: Select Distinct**
- Ví dụ:

```
SELECT DISTINCT Order_Date as
"Date of Order"
FROM Orders
```

```
SELECT diadiem
FROM   DIADIEM_PHG
```

diadiem
TP HCM
HA NOI
TP HCM

```
SELECT   DISTINCT diadiem
FROM     DIADIEM_PHG
```

diadiem
TP HCM
HA NOI

Lệnh select

- 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

Lệnh select

- Truy vấn dùng các toán tử chuẩn trong biểu thức điều kiện:
- Ví dụ:

```
SELECT ProductID, UnitPrice
FROM Product_T
WHERE UnitPrice < 275;
```

Lệnh select

- Ví dụ:

```
SELECT ProductID, ProductName,UnitPrice
FROM Products
WHERE ProductName like 'N%'
AND UnitPrice > 300
```

Truy vấn trên nhiều bảng

- Ví dụ:

```
SELECT c.CustomerID, CompanyName, OrderID, OrderDate
FROM Customers C INNER JOIN Orders O
ON C.CustomerID = O.CustomerID
```

Truy vấn seft join

- Truy vấn trên một bảng liên kết với chính nó
- Ví dụ:

```
SELECT e.Firstname+' '+e.Lastname AS 'Employee',
m.Firstname+' '+m.Lastname AS 'Manager'
FROM Employees e JOIN Employees m
ON e.ManagerID = m.EmployeeID
```

Lệnh Union

- **Union** nối kết quả từ nhiều câu lệnh select
- Ví dụ:

```
SELECT Firstname+' '+Lastname AS name, Homephone
FROM Employees
UNION
SELECT Companyname, Phone
FROM Customers
```

12

Các hàm tổng hợp dữ liệu

- **Các hàm tổng hợp - Aggregate Functions**: tổng hợp thông tin từ nhiều bộ thành một bộ.
- Chức năng grouping được sử dụng để tạo nhóm trước khi thực hiện tổng hợp dữ liệu.
- Các hàm tổng hợp: **COUNT, SUM, MAX, MIN, AVG.**

13

Các hàm tổng hợp dữ liệu

- Ví dụ: *đếm số nhóm sách*

```
select count(*) as sonhom
from nhomsach
```

- *Ví dụ: tính tổng số cuốn sách đã bán*

```
select sum(soluong) as tongsoluong
from chitiethoadon
```

Mệnh đề GROUP BY

- **Mệnh đề GROUP BY:** chỉ định các thuộc tính kết nhóm xuất hiện trong mệnh đề **select**, kết quả của hàm thống kê được áp dụng cho các bộ trong cùng một nhóm.
- Ví dụ: *tính tổng số cuốn sách của mỗi nhóm sách*

```
select n.manhom, tennhom, sum(soluong) as tongsoluong
from nhomsach n join danhmucsach d on n.MaNhom=d.MaNhom
                join chitiethoadon c on d.MaSach=c.MaSach
group by n.manhom, tennhom
```

15

Mệnh đề HAVING

- **Mệnh đề Where**: xác định điều kiện lọc trước khi khi nhóm dữ liệu
- Ví dụ: *liệt kê tổng số lượng của các nhóm sách có mã là .*

Select n.manhom, tennhom, tongsl=sum(soluong)

From nhomsach n join danhmucsach d on n.manhom=d.manhom join chitiethoadon c on d.masach=c.masach

Where n.manhom =1

Gropu by n.manhom, tennhom

16

Mệnh đề HAVING

- **Mệnh đề Having**: xác định điều kiện lọc sau khi nhóm dữ liệu
- Ví dụ*: liệt kê các nhóm sách có tổng số sách >=30.*

```
select n.manhom, tennhom, sum(soluong) as tongsoluong
from nhomsach n join danhmucsach d on n.MaNhom=d.MaNhom
                join chitiethoadon c on d.MaSach=c.MaSach
group by n.manhom, tennhom
having sum(soluong)>=30
```

Select - Compute By

- Trả về **tổng giá trị** của nhiều dòng tương ứng với field.
- Ví dụ:

```
SELECT SalesOrderID, UnitPrice, UnitPriceDiscount
FROM Sales.SalesOrderDetail
ORDER BY SalesOrderID
COMPUTE SUM(UnitPrice), SUM(UnitPriceDiscount);
```

```
SELECT SalesOrderID, UnitPrice, UnitPriceDiscount
FROM Sales.SalesOrderDetail
ORDER BY SalesOrderID
COMPUTE SUM(UnitPrice), SUM(UnitPriceDiscount) BY SalesOrderID;
```

18

Truy vấn con - Nested Queries

- **Nested query** là một query chứa một query khác, query được chứa bên trong gọi là *subquery* .
  - Subquery thường xuất hiện trong mệnh đề WHERE của query.
  - Ngoài ra Subquery cũng có thể xuất hiện trong mệnh đề FROM hoặc HAVING.

19

Các phép toán dùng trong nested query

- **IN**: so sánh một giá trị **v** với một tập giá trị **V,** kết quả là TRUE nếu **v** tồn tại trong **V.**
- Ví dụ: *liệt kê các sách thuộc nhóm sách 'Tin học'*

```
select masach, tensach
from danhmucsach
where MaNhom in (select MaNhom from nhomsach
                 where TenNhom =N'Tin học')
```


20

Các phép toán dùng trong nested query

- **NOT IN:** so sánh một giá trị **v** với một tập giá trị **V,** kết quả là TRUE nếu **v** không tồn tại trong **V**
- Ví dụ: *Tìm những quyển sách chưa bán*

```
select masach, tensach
from danhmucsach
where MaSach not in (select masach from chitiethoadon)
```


21

Các phép toán dùng trong nested query

- **ANY:** kết hợp với các phép toán **op** (>, >=, <, <=, and <>), kết quả là TRUE nếu và chỉ nếu các giá trị trong tập **v** thỏa mãn phép toán **op** với ít nhất là một giá trị trong **V**.
  - >ANY có nghĩa lớn hơn ít nhất 1 giá trị
  - Vd: >ANY (1, 2, 3) lớn hơn 1
- **Ví dụ:** *liệt kê các sách có đơn giá lớn hơn đơn giá của ít nhất một sách trong nhóm sách 'N002'*

```
select masach, tensach, DonGia
from danhmucsach
where DonGia >any (select DonGia from danhmucsach
                   where MaNhom='N002')
```

22

Các phép toán dùng trong nested query

- **ALL**: kết hợp với các phép toán **op** (>, >=, <, <=, and <>), kết quả là TRUE nếu và chỉ nếu các giá trị trong tập **v** thỏa mãn phép toán **op** với tất cả giá trị trong **V**
  - >ALL có nghĩa lớn hơn mọi giá trị.
  - Vd: >ALL (1, 2, 3) lớn hơn 3
- **Ví dụ:** *liệt kê các sách có đơn giá lớn hơn đơn giá của tất cả sách trong nhóm sách 'N002'*

```
select masach, tensach, DonGia
from danhmucsach
where DonGia >all (select DonGia from danhmucsach
                   where MaNhom='N002')
```

23

Các phép toán dùng trong nested query

- **EXISTS**: kiểm tra kết quả của subquery có rỗng hay không, exists trả về giá trị là TRUE nếu kết quả của subquery chứa ít nhất là một bộ giá trị.
- Ví dụ: *liệt kê các nhân viên lập hóa đơn*

```
select manv, TenNV
from nhanvien n
where exists (select MaNV from hoadon
              where MaNV= n.MaNV)
```

24

Các phép toán dùng trong nested query

- **NOT EXISTS**: trả về giá trị là TRUE nếu kết quả của subquery không chứa bộ giá trị nào.

Ví dụ: *liệt kê các nhân viên không lập hóa đơn nào*

```
select manv, TenNV
from nhanvien n
where not exists (select MaNV from hoadon
                  where MaNV= n.MaNV)
```

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

- Có thể tạo một bảng mới dựa vào kết quả của câu lệnh **select.**
- **Ví dụ:**

```
SELECT C.CustomerID AS NameId, OrderID, OrderDate
INTO Customer_Order
FROM Customers C INNER JOIN Orders O
ON C.CustomerID = O.CustomerID
WHERE month(OrderDate) = 7
```

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
```

Select With

- Trả ra một tập kết quả lưu trữ tạm thời trong common table expression (CTE).
- Tạo CTE

```
WITH name_CTE(fields_list)
AS
    (
        SELECT stmt
    )
```

```
USE AdventureWorks2008;
GO
--1
WITH SumSale AS
(SELECT SUM(TotalDue) AS SumTotalDue,
CustomerID
FROM Sales.SalesOrderHeader
GROUP BY CustomerID)
```

Select With

- Sử dụng CTE

```
SELECT fields_list
FROM name_CTE
[WHERE]
[GROUP BY]
[ORDER BY]
[HAVING]
```

```
USE AdventureWorks2008;
GO
--1
WITH SumSale AS
(SELECT SUM(TotalDue) AS SumTotalDue,
CustomerID
FROM Sales.SalesOrderHeader
GROUP BY CustomerID)

SELECT o.CustomerID, TotalDue,
TotalDue / SumTotalDue * 100 AS
PercentOfSales
FROM SumSale INNER JOIN
Sales.SalesOrderHeader AS o
ON SumSale.CustomerID = o.CustomerID
ORDER BY CustomerID;
```

Select với cấu trúc Case..When

– Searched CASE function

```
CASE
      WHEN Boolean_expression THEN
            result_expression [ ...n ]
      [ ELSE else_result_expression ]
END
```

Select với cấu trúc Case..When

- Example 2 :

```
Select ProductName, Unitprice,
'Classification'=CASE
                when Unitprice<10 then 'Low price'
                When Unitprice Between 10 and 20 then
                                'Moderately Price'
                when Unitprice>20 then 'Expensive'
                else 'Unknown'
  end
From Products
```

Select với cấu trúc Case..When

- Example 2 :

```
Select productid, Quantity, UnitPrice, [discount%]=
CASE
    When Quantity <=5 then 0.05
    When Quantity between 6 and 10 then 0.07
    When Quantity between 11 and 20 then 0.09
Else 0.1
end
From [Order Details]
Order by Quantity, Productid
```

Select Merge

- Điểm mới trong SQL Server 2008 cho phép merge dữ liệu từ 2 table trở lên.
- Ví dụ:

```
CREATE TABLE T1(col1 Int Primary Key);
CREATE TABLE T2(col1 Numeric(12, 2) Primary Key);

SELECT T1.col1, T2.col1
FROM T1 Inner Merge Join T2
ON dbo.T1.col1 = dbo.T2.col1;
```

Merge query

- MERGE query là một loại query có thể thực hiện các hoạt động Insert, Update, Delete trên một bảng đích dựa trên các kết quả của một liên kết với một bảng nguồn
- Lợi thế quan trọng của câu lệnh MERGE là tất cả các dữ liệu được đọc và xử lý một lần.

Merge query

- **Cú pháp đơn giản**

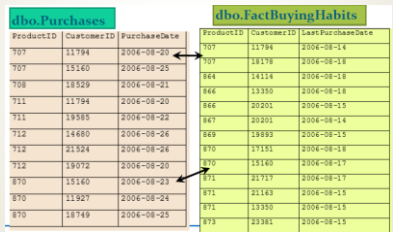
```
Merge TargetTable
using SourceTable
on Joinconditions
[WHEN MATCHED THEN DML]
[WHEN NOT MATCHED BY TARGET THEN DML]
[WHEN NOT MATCHED BY SOURCE THEN DML]
```

Merge query

TABLE 42.1 Join Methods Used for **WHEN** Clauses

Specified WHEN Clauses	Join Method
WHEN MATCHED clause only	INNER JOIN
WHEN NOT MATCHED BY TARGET clause, but *not* the WHEN NOT MATCHED BY SOURCE clause	LEFT OUTER JOIN from source to target
WHEN MATCHED clause and the WHEN NOT MATCHED BY SOURCE clause, but *not* the WHEN NOT MATCHED BY TARGET clause	RIGHT OUTER JOIN from source to target
WHEN NOT MATCHED BY TARGET clause and the WHEN NOT MATCHED BY SOURCE clause	FULL OUTER JOIN
WHEN NOT MATCHED BY SOURCE clause only	ANTI SEMI JOIN

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	2006-08-24
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

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)

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)
```

Ví dụ - Merge query

```
CREATE TABLE dbo.BookInventory
(
 TitleID INT NOT NULL PRIMARY KEY,
 Title NVARCHAR(100) NOT NULL,
 Quantity INT NOT NULL
  CONSTRAINT Qutitydeft1 DEFAULT 0
);
```

Bảng Đích

```
CREATE TABLE dbo.BookOrder
(
 TitleID INT NOT NULL PRIMARY KEY,
 Title NVARCHAR(100) NOT NULL,
 Quantity INT NOT NULL
  CONSTRAINT Qutitydefit DEFAULT 0
);
```

Bảng Nguồn

Ví dụ Merge - WHEN MATCHED

```
MERGE BookInventory bi
USING BookOrder bo
ON bi.TitleID = bo.TitleID
WHEN MATCHED THEN
  UPDATE
  SET bi.Quantity = bi.Quantity+bo.Quantity;
```

Ví dụ Merge - WHEN MATCHED

```
MERGE BookInventory bi
USING BookOrder bo
ON bi.TitleID = bo.TitleID
WHEN MATCHED AND
  bi.Quantity + bo.Quantity = 0 THEN
  DELETE
WHEN MATCHED THEN
  UPDATE
  SET bi.Quantity = bi.Quantity + bo.Quantity;
```

Ví dụ Merge - WHEN MATCHED

```
MERGE BookInventory bi
USING BookOrder bo
ON bi.TitleID = bo.TitleID
WHEN MATCHED AND
  bi.Quantity + bo.Quantity = 0 THEN
  DELETE
WHEN MATCHED THEN
  UPDATE
  SET bi.Quantity = bi.Quantity +
bo.Quantity;
```

Ví dụ Merge -WHEN NOT MATCHED BY TARGET

```
MERGE BookInventory bi
USING BookOrder bo
ON bi.TitleID = bo.TitleID
WHEN MATCHED AND
  bi.Quantity + bo.Quantity = 0 THEN DELETE
WHEN MATCHED THEN UPDATE
  SET bi.Quantity = bi.Quantity + bo.Quantity
WHEN NOT MATCHED BY TARGET THEN
  INSERT (TitleID, Title, Quantity)
  VALUES (bo.TitleID, bo.Title,bo.Quantity);
```

Ví dụ Merge -WHEN NOT MATCHED BY SOURCE

```
MERGE BookInventory bi
USING BookOrder bo
ON bi.TitleID = bo.TitleID
WHEN MATCHED AND
  bi.Quantity + bo.Quantity = 0 THEN DELETE
WHEN MATCHED THEN UPDATE
  SET bi.Quantity = bi.Quantity + bo.Quantity
WHEN NOT MATCHED BY TARGET THEN
  INSERT (TitleID, Title, Quantity)
  VALUES (bo.TitleID, bo.Title,bo.Quantity)
WHEN NOT MATCHED BY SOURCE
  AND bi.Quantity = 0 THEN DELETE;
```

Implementing the WHEN NOT MATCHED BY SOURCE

```
MERGE BookInventory bi
USING BookOrder bo
ON bi.TitleID = bo.TitleID
WHEN MATCHED AND
  bi.Quantity + bo.Quantity = 0 THEN DELETE
WHEN MATCHED THEN UPDATE
  SET bi.Quantity = bi.Quantity +
bo.Quantity
WHEN NOT MATCHED BY TARGET THEN
  INSERT (TitleID, Title, Quantity)
  VALUES (bo.TitleID, bo.Title,bo.Quantity)
WHEN NOT MATCHED BY SOURCE
  AND bi.Quantity = 0 THEN DELETE;
```

Pivot query

- Pivot query được dùng để tạo bảng thống kê dạng 2 chiều.
- Cú pháp:

pivoted_table ::=
table_source PIVOT (*aggregate_function* (
value_column)
PIVOT and UNPIVOT FOR *pivot_column*
IN (*column_list*) *table_alias*

Pivot

To use the PIVOT feature, you first decide which column contains the important values for the query.

```
SELECT empid, A, B, C, D
FROM (SELECT empid, custid, qty
FROM dbo.Orders) AS D
PIVOT(SUM(qty) FOR custid IN(A, B, C, D)) AS P;
```

Pivot

Cách 1: Sử dụng Case..When

Example:

```
SELECT empid,
SUM(CASE WHEN custid = 'A' THEN qty END) AS A,
SUM(CASE WHEN custid = 'B' THEN qty END) AS B,
SUM(CASE WHEN custid = 'C' THEN qty END) AS C,
SUM(CASE WHEN custid = 'D' THEN qty END) AS D
FROM dbo.Orders
GROUP BY empid;
```

Pivot

Cách 2: Pivoting with the Native T-SQL *PIVOT* Operator

Cú pháp:

```
SELECT ...
FROM <source_table_or_table_expression>
PIVOT(<agg_func>(<aggregation_element>)
FOR <spreading_element>
IN (<list_of_target_columns>)) AS <result_table_alias>
...;
```

Ex:

```
SELECT empid, A, B, C, D
FROM (SELECT empid, custid, qty
FROM dbo.Orders) AS D
PIVOT(SUM(qty) FOR custid IN(A, B, C, D)) AS P;
```

UnPivot

Unpivoting with the Native T-SQL *UNPIVOT* Operator

Cú pháp:

```
SELECT ...
FROM <source_table_or_table_expression>
UNPIVOT(<target_col_to_hold_source_col_values>
FOR <target_col_to_hold_source_col_names>
IN(<list_of_source_columns>)) AS
<result_table_alias>
```

UnPivot

Sử dụng CASE..WHEN

Ex:

```
SELECT empid, custid,
CASE custid
WHEN 'A' THEN A
WHEN 'B' THEN B
WHEN 'C' THEN C
WHEN 'D' THEN D
END AS qty
FROM dbo.EmpCustOrders
CROSS JOIN (VALUES('A'),('B'),('C'),('D')) AS Custs(custid);
```

UnPivot

Sử dụng CASE..WHEN

Ex:

```
SELECT *
FROM (SELECT empid, custid,
CASE custid
WHEN 'A' THEN A
WHEN 'B' THEN B
WHEN 'C' THEN C
WHEN 'D' THEN D
END AS qty
FROM dbo.EmpCustOrders
CROSS JOIN (VALUES('A'),('B'),('C'),('D')) AS
Custs(custid)) AS D WHERE qty IS NOT NULL;
```

UnPivot

Unpivoting with the Native T-SQL *UNPIVOT* Operator

Ex:

```
SELECT empid, custid, qty
FROM dbo.EmpCustOrders
UNPIVOT(qty FOR custid IN(A, B, C, D)) AS U;
```

Lệnh Insert

- **Cú pháp:**

```
Insert <Table Name>[field_List]
Values (value_1, value_2, …, value_n)
Values (…)…
```

Hoặc

```
Insert <Table Name>[field_List]
(Select statement)
```

Lệnh Insert

- **Ví dụ:**

```
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 Delete

- **Cú pháp**:

```
Delete from <Table name>
[Where <condition>]
```

Hoặc

```
Delete Top [n] percent
From <Table name>
[Where <condition>]
```

Hoặc

```
Output DeletedDelete From
<Table name>
```

Ví dụ

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

Lệnh Update

- **Cú pháp:**

```
Update <Table Name>
Set Field_Name = New_value
[Where <Condition>]
```

Lệnh Update

- 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
```

Câu lệnh TRUNCATE TABLE

- Dùng để xóa các dòng của table
- Nhanh hơn lệnh DELETE
- Không dùng với Trigger

Cú pháp: **TRUNCATE TABLE** ***table_name***

VD

TRUNCATE TABLE NewProducts