Ngôn ngữ lập trình Transaction – SQL

Sau khi học xong phần này, sinh viên có thể vận dụng các lệnh trong ngôn ngữ lập trình Transaction – SQL để viết các đoạn mã lệnh trong SQL Server

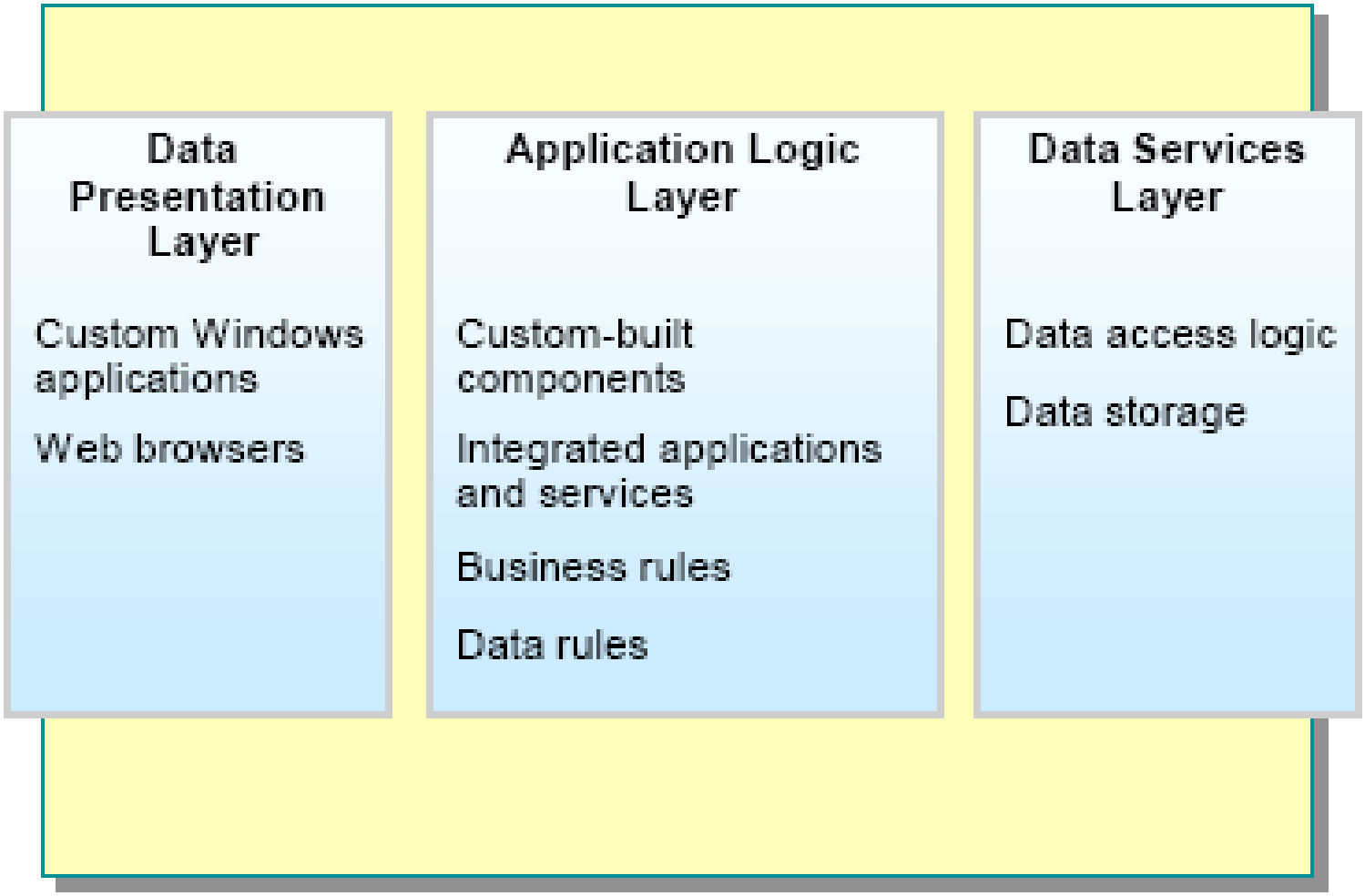
Nội dung bài học

1. Enterprise Application Architechter
2. Biến và các kiểu dữ liệu
3. Toán tử
4. Hàm
5. Các câu lệnh điều khiển
6. Các cách sử dụng các lệnh T-SQL

Thiết kế Enterprise Application Architecture

- Xác định các lớp logical (Logical Layers)
- Thiết kế các lớp vật lý (Physical Layers)
- Truy xuất dữ liệu

Logical Layers



Data presentation Layer

- Được xem là user service and cho phép user xem và thao tác lên data: web browser and các Microsoft Windows® applications
- Sử dụng các service mà application logic layer cung cấp

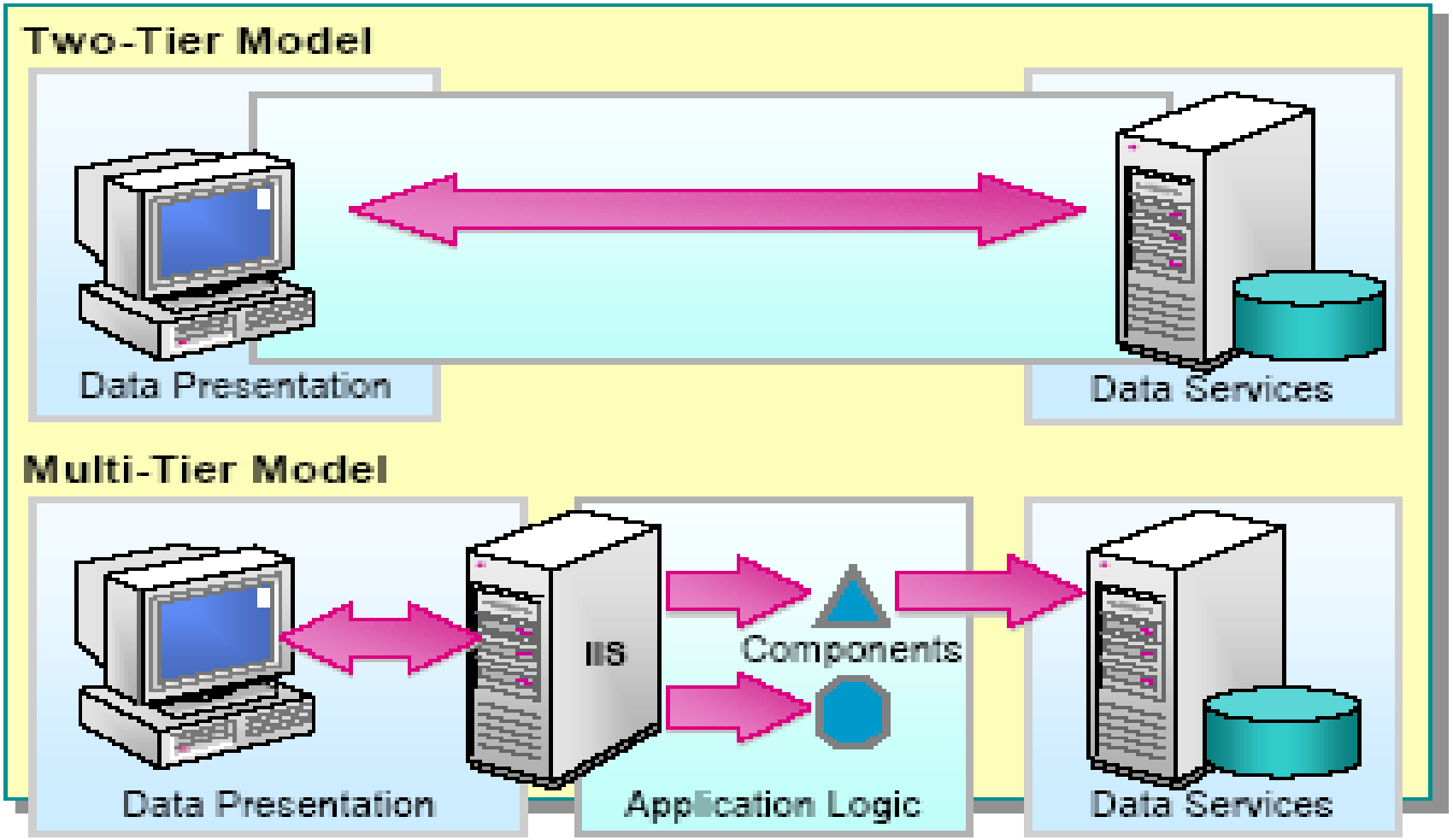
Application Logic Layer

- Chứa application logic, định nghĩa các rules và processes giúp cho user không cần truy xuất trực tiếp vào database
- Clients kết nối vào business service để kết nối vào data server. Business service là các custom-built components hoặc integrated applications và services, ví dụ như Web services.
- Application logic layer chứa các components để tạo thành transaction services, messaging services, hoặc object và connection management services.

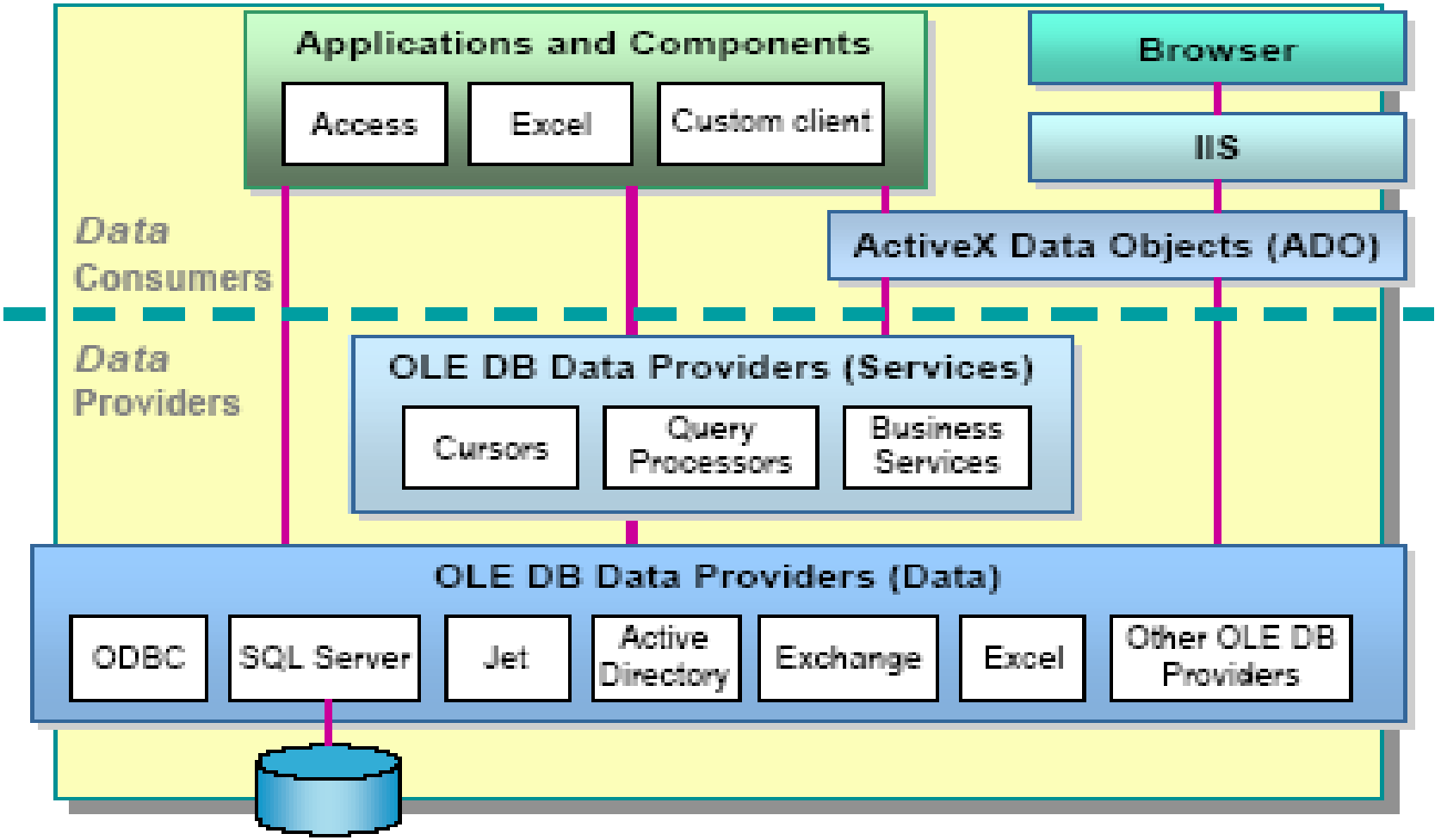
Data Services Layer

- Data services bao gồm data access logic và data storage.
- Bao gồm các SQL Server stored procedures để quản lý data traffic và integrity trên the database server.

Thiết kế các lớp vật lý



Truy xuất dữ liệu



Khai báo biến

Dùng từ khoá declare để khai báo biến

DECLARE {@local_variable data_type} [,...*n*]

Gán giá trị cho biến

SET *@local_variable_name = expression*

Ví dụ

```
DECLARE @vLastName char(20),
@vFirstName varchar(11)

SET @vLastName = 'Dodsworth‘

SELECT @vFirstName = FirstName
FROM Northwind..Employees
WHERE LastName = @vLastName

PRINT @vFirstName + ' ' + @vLastName
```

Gán giá trị cho biến bằng từ khoá set

hoặc bằng câu lệnh select

Data Type (1)

- **Integers**
  - Bigint: 8 bytes
  - Int: 4 bytes
  - Smallint: 2 bytes
  - Tinyint: 1 byte, từ 0 -> 255.
- **bit**
  - Bit: 1 hoặc 0 value.
- **decimal and numeric**
  - Decimal từ -10^38+1->10^38 –1.
  - Numeric: giống **decimal.**
- **money and smallmoney**
  - Money: 8 bytes
  - Smallmoney: 4 bytes
- **Approximate Numerics**
  - Float: từ -1.79E + 308 -> 1.79E + 308.
  - Real: từ -3.40E + 38 -> 3.40E + 38.

Data Type (2)

- **datetime and smalldatetime**
  - Datetime: từ 1/1/1753-> 31/12/9999.
  - Smalldatetime từ 1/1/1900, -> 6/6/2079.
- **Character Strings**
  - Char: Fixed-length non-Unicode character, <= 8,000 ký tự
  - Varchar: Variable-length non-Unicode , <= 8,000 ký tự
  - Text: Variable-length non-Unicode <=2^31 - 1 (2,147,483,647) ký tự
- **Unicode Character Strings**
  - ncharFixed-length Unicode , <=4,000 characters.
  - nvarcharVariable-length Unicode, <=4,000 characters
  - Ntext Variable-length Unicode <= 2^30 - 1 (1,073,741,823) characters.
- **Other Data Type**
  - Cursor: là một tham chiếu đến một cursor.
- Một biến không thể có kiểu là **text**, **ntext**, hoặc **image**

Toán tử (operators)

- Các loại toán tử
  - Số học: *, /, %, - , +
  - So sánh: =, <>, >, >=, <, <=
  - Nối chuỗi: +
  - Luận lý: AND, OR, NOT

Thứ tự ưu tiên các toán tử

Type	Operator	Symbol
Grouping	Primary grouping	( )
Arithmetic	Multiplicative	* / %
Arithmetic	Additive	- +
Other	String concatenation	+
Logical	NOT	NOT
Logical	AND	AND
Logical	OR	OR

Functions (1)

- Aggregate functions: tính toán trên một nhóm và trả về một giá trị. Ví dụ:

SELECT AVG(UnitPrice) FROM Products

Products

28.8663

(1 row(s) affected)

Functions (2)

- Scalar functions: Tác động lên một giá trị và trả về một giá trị. Có thể sử dụng hàm trong các biểu thức.
- Chúng ta có thể nhóm các scalar function theo nhóm sau:

Configuration	Trả về các thông tin về configuration
Cursor	Trả về các thông tin về Cursor
DateTime	Hàm tác động lên giá trị dateTime nhập vào và trả về một giá trị là string, numeric, hoặc datetime
Mathematical	Hàm số học
Metadata	Thông tin về database
String	Các hàm chuỗi

Functions (3)_ Ví dụ

```
SELECT DB_NAME() AS 'database'
Database
Northwind
(1 row(s) affected)
```

```
SET DATEFORMAT dmy
GO
DECLARE @vdate datetime
SET @vdate = '29/11/00'
SELECT @vdate
2000-11-29 00:00:00.000
```

Mathematical Functions

ABS	DEGREES	RAND
ACOS	EXP	ROUND
ASIN	FLOOR	SIGN
ATAN	LOG	SIN
ATN2	LOG10	SQUARE
CEILING	PI	SQRT
COS	POWER	TAN
COT	RADIANS	

Aggregate Functions

AVG	MAX
BINARY_CHECKSUM	MIN
CHECKSUM	SUM
CHECKSUM_AGG	STDEV
COUNT	STDEVP
COUNT_BIG	VAR
GROUPING	VARP

DateTime functions

Function	Determinism
DATEADD	Deterministic
DATEDIFF	Deterministic
DATENAME	Nondeterministic
DATEPART	Deterministic except when used as DATEPART (dw, date). dw, the weekday datepart, depends on the value set by SET DATEFIRST, which sets the first day of the week.
DAY	Deterministic
GETDATE	Nondeterministic
GETUTCDATE	Nondeterministic
MONTH	Deterministic
YEAR	Deterministic

String functions

ASCII	NCHAR	SOUNDEX
CHAR	PATINDEX	SPACE
CHARINDEX	REPLACE	STR
DIFFERENCE	QUOTENAME	STUFF
LEFT	REPLICATE	SUBSTRING
LEN	REVERSE	UNICODE
LOWER	RIGHT	UPPER
LTRIM	RTRIM	

Cast và Convert

- CAST **(** *expression* AS *data_type* **)**

- CONVERT **(** *data_type* [**(** *length* **)**] **,** *expression* [**,** *style*] **)**

Control-of-Flow Language

1. **BEGIN...END : định nghĩa một khối lệnh**

```
BEGIN
   sql_statement | statement_block
END
```

2. If .. else

IF *Boolean_expression*
{ *sql_statement* | *statement_block* }
[ELSE
{ *sql_statement* | *statement_block* }]

```
if (select count(*) from customers where country='canada') >
    0
begin
   print ‘There are many Canada customers'
end
else
    print ‘Welcome'
```

Ví dụ lệnh If…. Else

```
USE Northwind
IF EXISTS (SELECT OrderID FROM Orders
WHERE CustomerID = 'Frank')
PRINT '*** Customer cannot be deleted ***'
ELSE
BEGIN
DELETE Customers WHERE CustomerID = 'Frank'
PRINT '*** Customer deleted ***'
END
```

3. While

WHILE *Boolean_expression*
{ *sql_statement* | *statement_block* }
[BREAK]
{ *sql_statement* | *statement_block* }
[CONTINUE]

- BREAK: thoát ra khỏi vòng while
- CONTINUE: restart lại vòng lập, bỏ qua các lệnh sau CONTINUE.

GOTO

GOTO LabelName

```
IF (SELECT SYSTEM_USER()) = 'payroll'
   GOTO calculate_salary
    -- Other program code would appear here.
    -- When the IF statement evaluates to TRUE,
    -- the statements between the GOTO and
    -- the calculate_salary label are ignored.
    -- When the IF statement evaluates to FALSE the
    -- statements following the GOTO are executed.
calculate_salary:
  -- Statements to calculate a salary would appear after the label.
```

Row ….Level

- ***Simple CASE function:***

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

- ***Searched CASE function:***

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

Ví dụ

```
SELECT ProductID, 'Product Inventory Status' =
CASE
    WHEN (UnitsInStock < UnitsOnOrder AND Discontinued = 0)
       THEN 'Negative Inventory - Order Now!'
    WHEN ((UnitsInStock-UnitsOnOrder) < ReorderLevel AND
                                          Discontinued = 0)
       THEN 'Reorder level reached- Place Order'
    WHEN (Discontinued = 1) THEN '***Discontinued***'
    ELSE 'In Stock'
END
FROM Products
```

Kết quả câu lệnh

ProductID	Product Inventory Status
1	In Stock
2	Negative Inventory - Order Now!
3	Negative Inventory - Order Now!
4	In Stock
5	***Discontinued***
6	In Stock
7	In Stock
8	In Stock
9	***Discontinued***
10	In Stock
11	Negative Inventory - Order Now!
12	In Stock
13	Reorder level reached - Place Order

.
.
.
(77 row(s) affected)

Áp dụng trong database QLVT

- Liệt kê các chi tiết hoá đơn của hoá đơn HD01 gồm các thông tin: Mã vật tư, SL, Giá bán, KM với

KM = 0 nếu SL <10

KM = (SL*giaban) * 0.1 nếu SL >=10

KM = (SL*giaban) * 0.2 nếu SL >=100

Comments

- Inline comments

```
SELECT ProductName,
(UnitsInStock + UnitsOnOrder) AS Max, -- Calculates inventory
 SupplierID
FROM Products
SELECT
```

- Block comments

```
/*
** This code retrieves all rows of the products table
** and displays the unit price, the unit price increased
** by 10 percent, and the name of the product.
*/
SELECT UnitPrice, (UnitPrice * 1.1), ProductName FROM Products
```

Lệnh RAISERROR

- Trả về một thông báo lỗi do user định nghĩa
- Cú pháp:

RAISERROR **(** { *msg_id* | *msg_str* }

{ **,** *severity* **,** *state* } **)**

[WITH *option* [**,**...*n*]]

Tham số

- *msg_id* là mã lỗi được lưu trong **sysmessages** table.
- *msg_str*: là chuỗi thông báo lỗi được định dạng giống như lệnh printf trong lập trình C
- Severity: mức độ nghiêm trọng của lỗi. Có giá trị từ 0->18 được dùng bởi user, từ 19 -> 25 được dùng bởi **sysadmin (dùng với** WITH LOG).
- State: số nguyên từ 1 ->127 mô tả mức độ cần thiết của lỗi.

Các cách thực hiện các lệnh T-SQL

- **Các lệnh có cấu trúc động**
- **Dùng Batch**
- **Dùng Scripts**
- **Dùng Transactions**
- **Dùng XML**

Dùng cấu trúc lệnh động

- Dùng lệnh **EXECUTE** với các hằng chuỗi và biến
- Sử dụng khi ta phải gán giá trị cho biến tại thời điểm execute
- Các biến và table tạm chỉ tồn tại trong thời gian thực thi lệnh.

```
DECLARE @dbname varchar(30), @tblname varchar(30)
SET @dbname = 'Northwind'
SET @tblname = 'Products'
EXECUTE
('USE ' + @dbname + ' SELECT * FROM '+ @tblname)
```

Sử dụng khối (batch)

- Một hoặc nhiều lệnh T- SQL được submit cùng lúc với nhau.
- Sử dụng lệnh GO để kết thúc một khối
- Các biến không thể tham chiếu sau lệnh GO
- Không thể dùng các lệnh sau đây trong batch:
  - CREATE PROCEDURE
  - CREATE VIEW
  - CREATE TRIGGER
  - CREATE RULE
  - CREATE DEFAULT

Ví dụ lệnh khối lệnh batch hợp lệ

```
CREATE DATABASE ...
CREATE TABLE ...
GO
CREATE VIEW1 ...
GO
CREATE VIEW2 ...
GO
```

CREATE DATABASE ...

CREATE TABLE ...

CREATE TRIGGER ...

CREATE TRIGGER ...

GO

Ví dụ về khối lệnh batch không hợp lệ

Sử dụng script

- Script là một tập tin có phần mở rộng là .sql, có nội dung là các lệnh T-SQL, được tạo bởi bất kỳ một Text editor nào.
- Được thực hiện trong công cụ SQL Query Analyzer hoặc osql Utility
- Được dùng lại khi cần.

Dùng Transactions

- Được xử lý giống một Batch
- Bảo đảm tính toàn vẹn dữ liệu
- Toàn bộ các lệnh trong transaction sẽ thành công hoặc không thành công
- Một transaction có thể có nhiều batch
- Transaction được bắt đầu bằng lệnh

BEGIN TRANSACTION

Và kết thúc bằng lệnh

COMMIT TRANSACTION

Hoặc

ROLLBACK TRANSACTION

Ví dụ transaction trong database QLVT

```
BEGIN TRANSACTION
   insert into chitiethoadon values ('hd001','vt03',50,null,30000)
   IF @@ERROR <> 0
   BEGIN
     RAISERROR ('Transaction not completed.', 16, -1)
     ROLLBACK TRANSACTION
   END
   update vattu set slton = slton-50 where mavt='vt03'
   IF @@ERROR <> 0
   BEGIN
     RAISERROR ('Transaction not completed.', 16, -1)
     ROLLBACK TRANSACTION
   END
COMMIT TRANSACTION
```

Dùng XML

- **Cho phép Client Browser định dạng dữ liệu:** Dùng mệnh để FOR XML trong lệnh SELECT để trả kết quả dạng XML
- Dùng **FOR XML AUTO**
- Hoặc **FOR XML RAW**
- Khi dùng mệnh đề FOR XML trong lệnh SELECT, ta không thể dùng:
  - Lệnh SELECT lồng nhau
  - Mệnh đề INTO .
  - Mệnh đề COMPUTE BY.
  - Dùng Stored procedures mà được gọi trong lệnh INSERT
  - Định nghĩa view hoặc user-defined function để trả về một rowset.

Ví dụ dùng XML (1)

Use QLVT

SELECT makh, tenkh FROM khachhang

FOR XML AUTO

```
<khachhang makh="KH01" tenkh="NGUYEN THI BE"/>
<khachhang makh="KH02" tenkh="LE HOANG NAM"/>
<khachhang makh="KH03" tenkh="TRAN THI CHIEU"/>
<khachhang makh="KH04" tenkh="MAI THI QUE ANH"/>
<khachhang makh="KH05" tenkh="LE VAN SANG"/>
```

Dùng XML (2)

Use QLVT

SELECT makh, tenkh FROM khachhang

FOR XML RAW

```
<row makh="KH01" tenkh="NGUYEN THI BE"/>
<row makh="KH02" tenkh="LE HOANG NAM"/>
<row makh="KH03" tenkh="TRAN THI CHIEU"/>
<row makh="KH04" tenkh="MAI THI QUE ANH"/>
<row makh="KH05" tenkh="LE VAN SANG"/>
<row makh="KH06" tenkh="TRAN HOANG KHAI"/>
```

Dùng database QLVT

Viết một đoạn mã lệnh để cho biết tổng trị giá của tất cả các hoá đơn của khách hàng có mã là KH01 trong năm 2000 với kết quả trả về

Khách hàng KH01 có tổng trị giá các hoá đơn là ……

Nếu Khách hàng đó không có hoá đơn nào thì in ra chuỗi:

Khách hàng này không mua hàng trong năm 2000

```
declare @tg int
select @tg=sum(sl*giaban)
from chitiethoadon as cthd, hoadon as hd
where cthd.mahd = hd.mahd and hd.makh='kh01‘ and year(ngay)=2000
if @tg>0
  print 'Khach hang kh01 co tong tri gia ' + str(@tg,10)
else
  print 'Khach hang chua mua hang'
```

Dùng database QLDIEM

- Liệt kê danh sách các sinh viên gồm các thông tin: MASV, HOTEN, NGAYSINH, PHAI (NU hoặc NAM) của những sinh viên có tuổi lớn hơn hoặc bằng 20.

- Khai báo một biến n để chứa số lượng các chi tiet hoá đơn có trong table cthd của hoá đơn hd001. Nếu n=0 thì xoá hoá đơn có mã là hd001 trong table hoá đơn, ngược lại thì xuất ra thông báo lỗi “Hoa don nay khong xoa duoc”

- Hãy thực hiện các lệnh để xoá một hoá đơn có mã hd là hd001. nếu hoá đơn hd001 có chi tiết hoá đơn thì phải xoá các chi tiết hoá đơn của hd này trước. Các lệnh này phải hoàn thành hoàn toàn.

- Lấy ra danh sách các hoá đơn có tổng trị giá lớn nhất
- Lấy ra các khách hàng đã mua những mặt hàng mà khách hàng kh01 đã mua.

- Tính số lượng hoá đơn của 2 khách hàng KH01 và KH02 và in ra kết quả so sánh 2 số lượng hoá đơn này