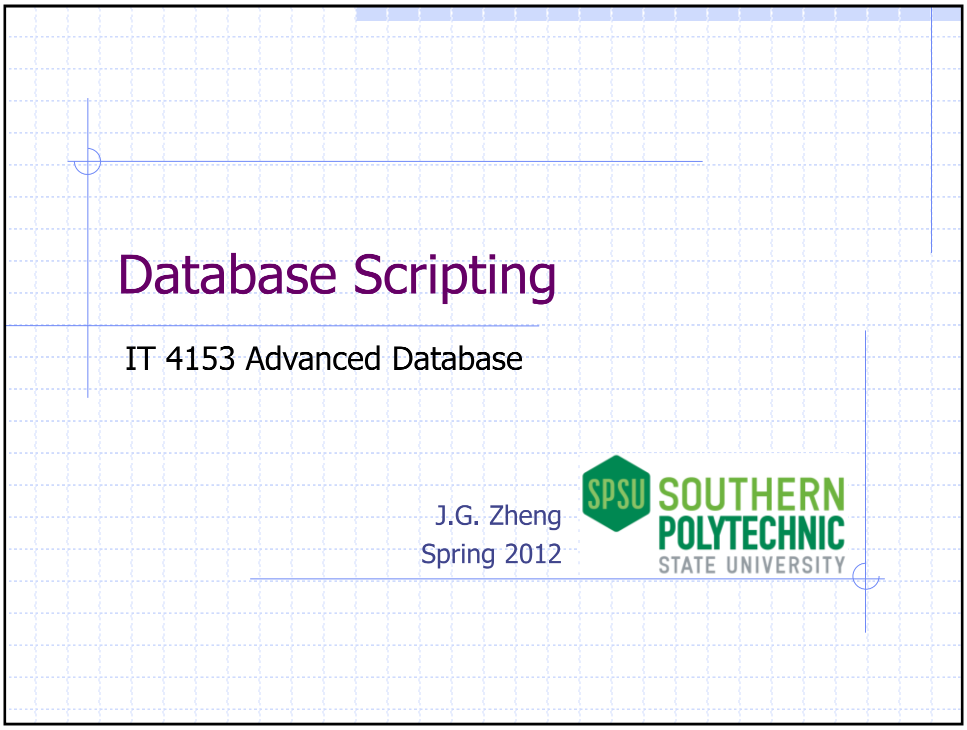
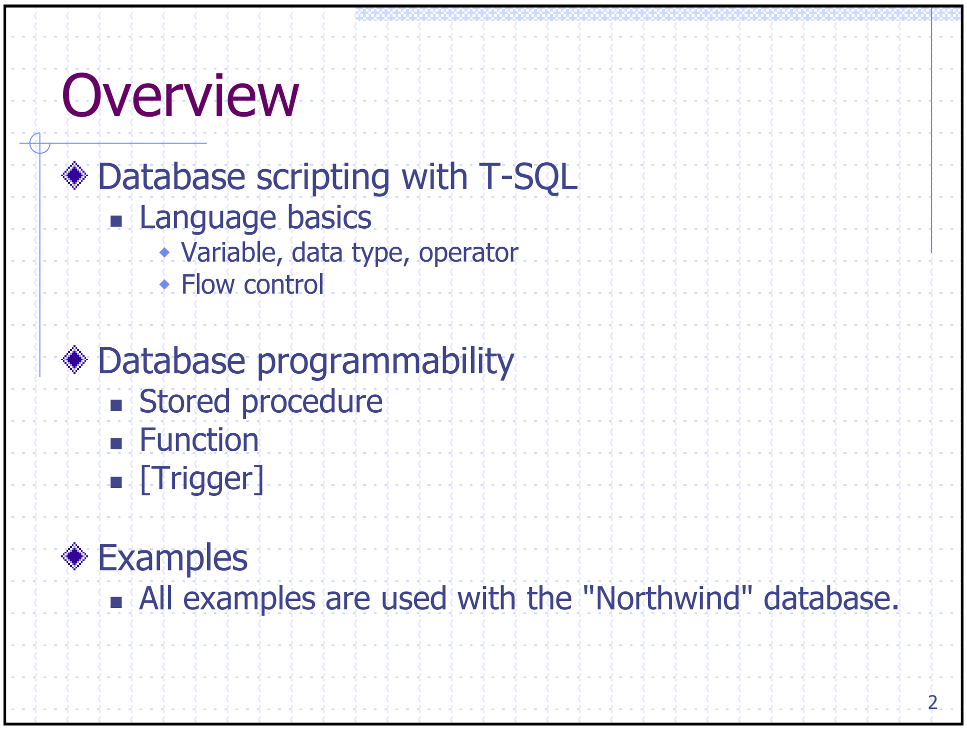


Database Scripting

IT 4153 Advanced Database

J.G. Zheng
Spring 2012
SOUTHERN POLYTECHNIC STATE UNIVERSITY



Overview

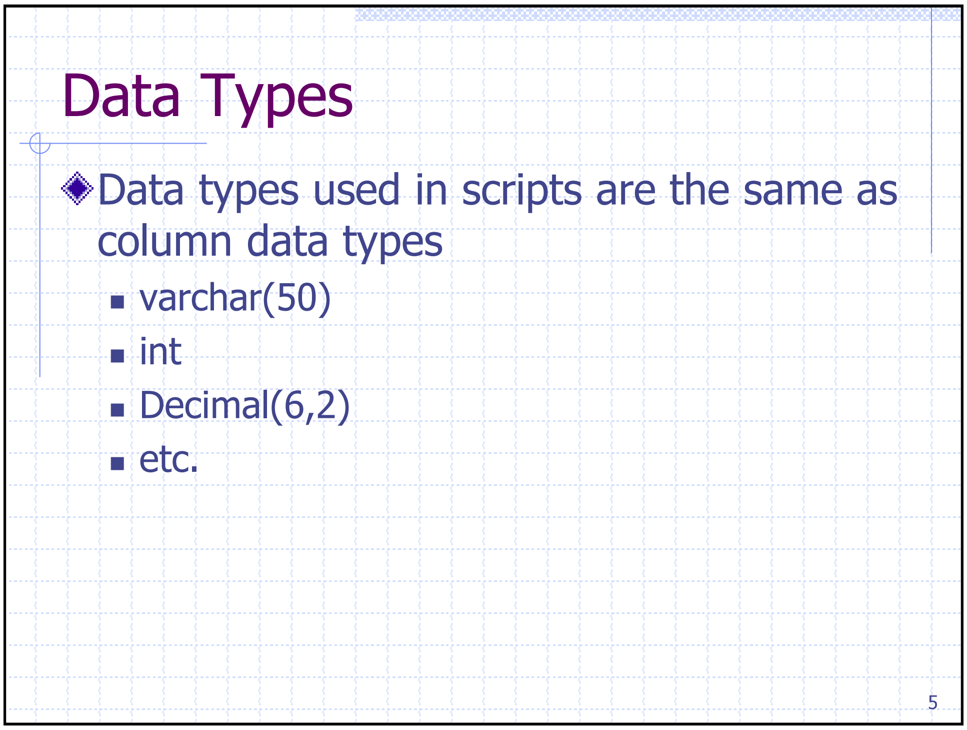
- Database scripting with T-SQL
  - Language basics
    - Variable, data type, operator
    - Flow control
- Database programmability
  - Stored procedure
  - Function
  - [Trigger]
- Examples
  - All examples are used with the "Northwind" database.

DBMS Scripting

- DBMS uses its own programs (scripts) for many types of operations
  - Script files
  - Stored procedure
  - Function
  - Trigger
- These programs use scripting languages that are usually an extension of the SQL standard
  - Transact-SQL (T-SQL)
    - SQL Server
  - PL-SQL
    - Oracle

Basic Symbols

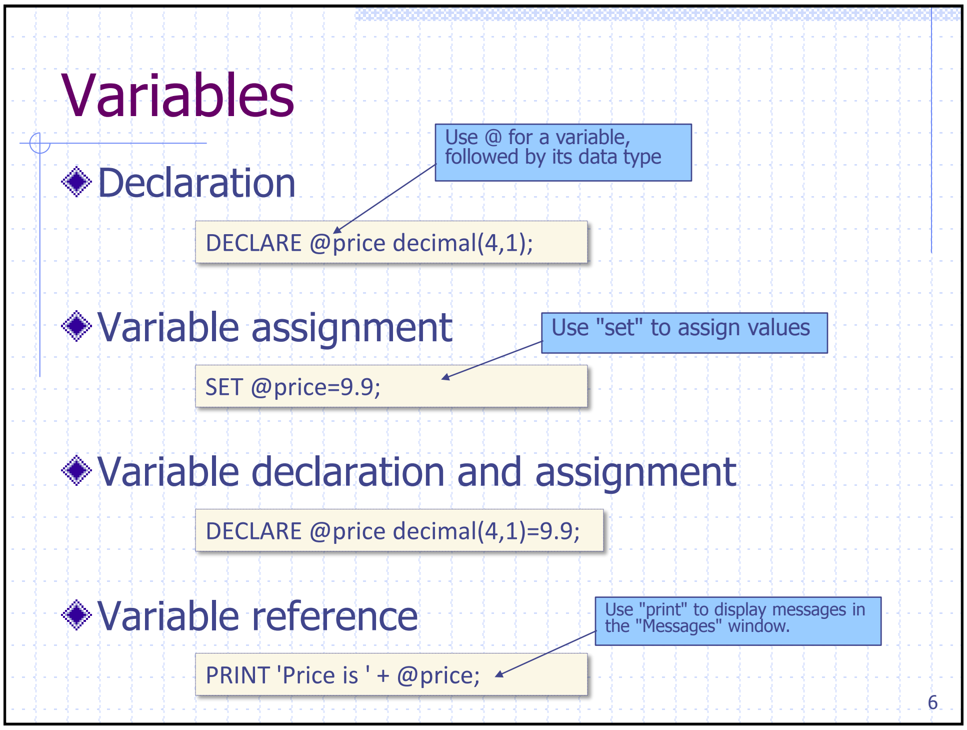
- End of statement
  - ; (optional in the current release)
- Comments
  - Single line
    - --
  - Multi line
    - /*…*/
- Literal/strings
  - '…'



Data Types

- Data types used in scripts are the same as column data types
  - varchar(50)
  - int
  - Decimal(6,2)
  - etc.

5



Variables

- Declaration

Use @ for a variable, followed by its data type

```
DECLARE @price decimal(4,1);
```

- Variable assignment

Use "set" to assign values

```
SET @price=9.9;
```

- Variable declaration and assignment

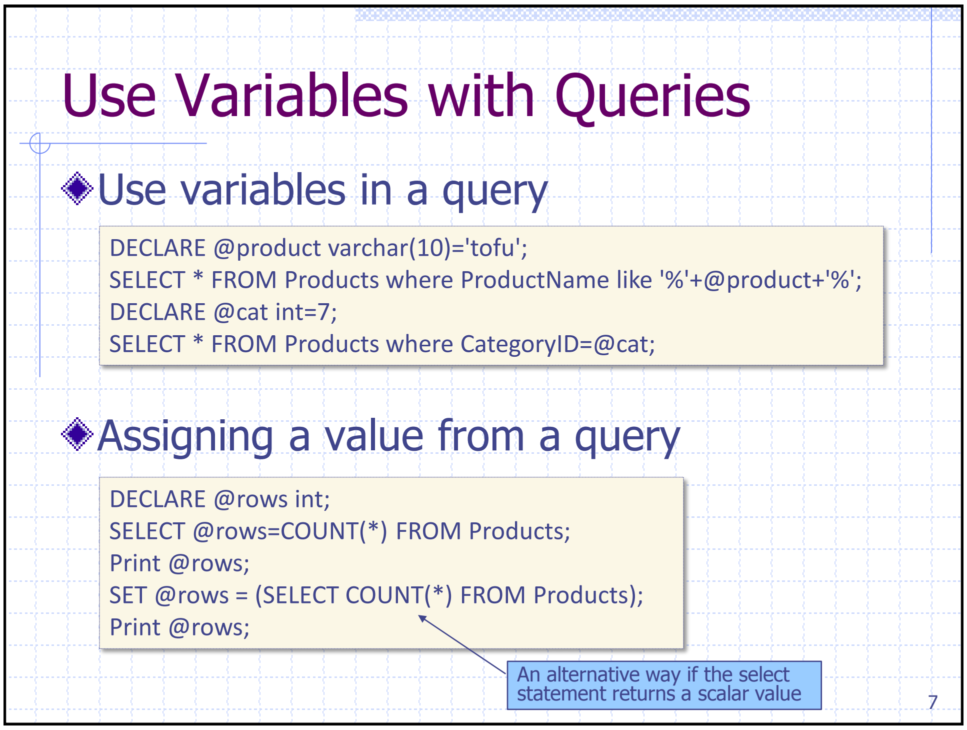
```
DECLARE @price decimal(4,1)=9.9;
```

- Variable reference

Use "print" to display messages in the "Messages" window.

```
PRINT 'Price is ' + @price;
```

6



Use Variables with Queries

- Use variables in a query

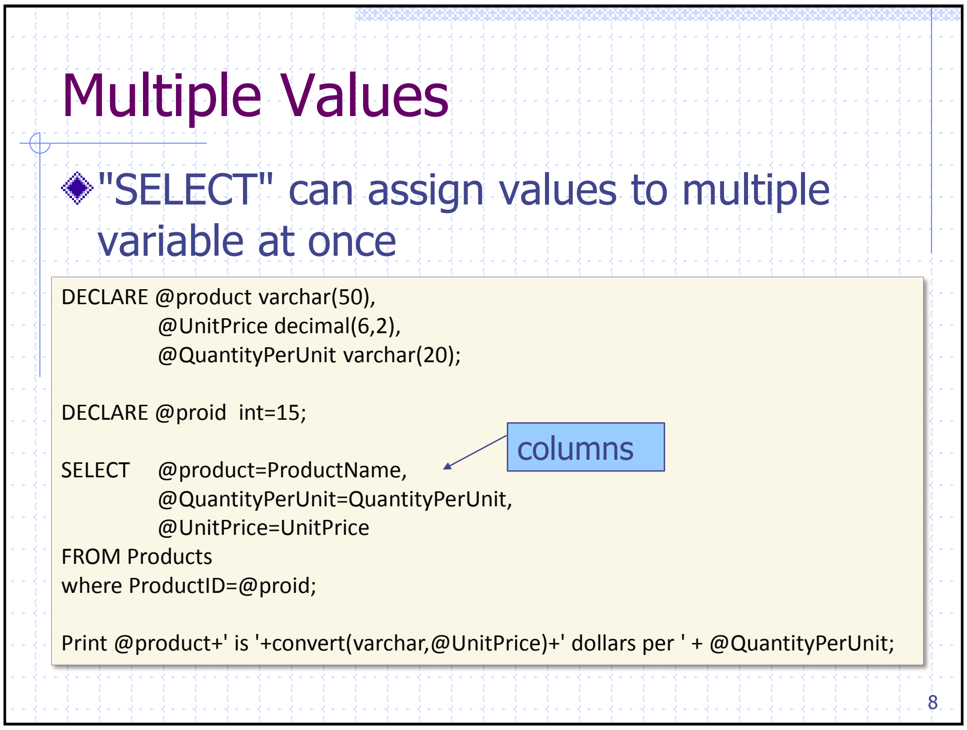
```
DECLARE @product varchar(10)='tofu';
SELECT * FROM Products where ProductName like '%'+@product+'%';
DECLARE @cat int=7;
SELECT * FROM Products where CategoryID=@cat;
```

- Assigning a value from a query

```
DECLARE @rows int;
SELECT @rows=COUNT(*) FROM Products;
Print @rows;
SET @rows = (SELECT COUNT(*) FROM Products);
Print @rows;
```

An alternative way if the select statement returns a scalar value

7



Multiple Values

- "SELECT" can assign values to multiple variable at once

```
DECLARE @product varchar(50),
            @UnitPrice decimal(6,2),
            @QuantityPerUnit varchar(20);

DECLARE @proid  int=15;

SELECT    @product=ProductName,
            @QuantityPerUnit=QuantityPerUnit,
            @UnitPrice=UnitPrice
FROM Products
where ProductID=@proid;

Print @product+' is '+convert(varchar,@UnitPrice)+' dollars per ' + @QuantityPerUnit;
```

columns

8

Operators

- Arithmetic
  - +, -, *, /, %
- Assignment
  - =
- Comparison
  - >, <, >=, <=, =, <> (or !=, which is standard)
  - between ... and ..., IN (...)
  - LIKE
  - IS NULL
- Logical
  - AND, OR, NOT, EXISTS
- Concatenation
  - +

9

Selection

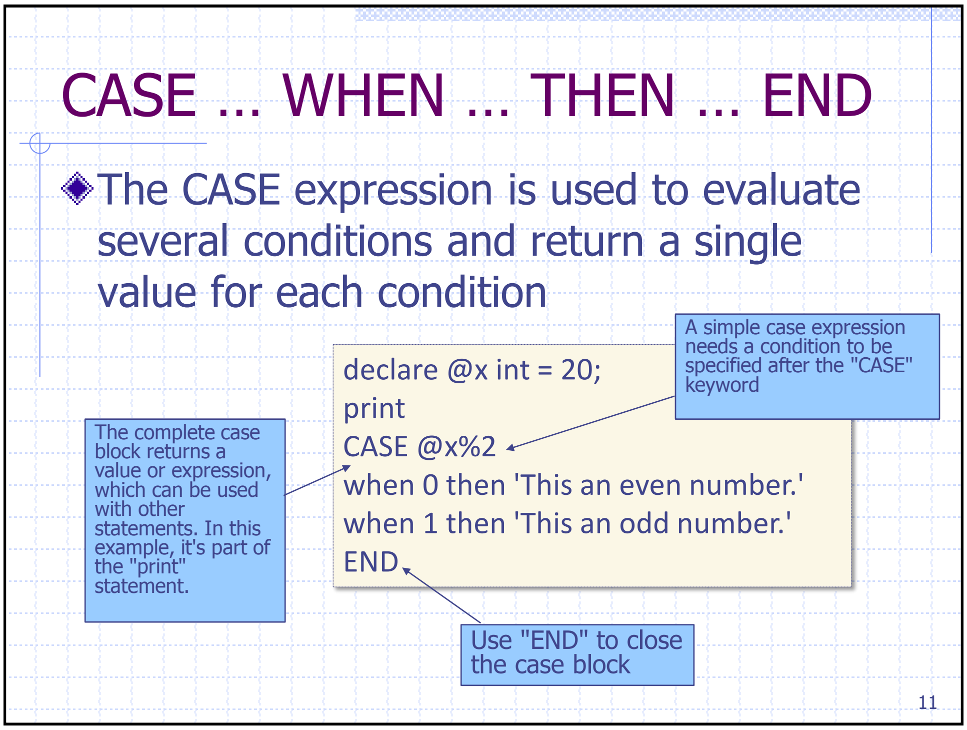
- Basic structures
  - if ...
  - if ... else
  - if ... else if ... else

```
declare @x int = 20;
IF @x%2 = 0
begin
        print 'This an even number.';
end
ELSE
begin
        print 'This an odd number.';
end
```

"begin" and "end" define the script block, just like { } in Java or C#.

- The test condition can be any expression used in the WHERE clause that return a boolean value
  - using comparison operators and logical operators

10



CASE … WHEN … THEN … END

- The CASE expression is used to evaluate several conditions and return a single value for each condition

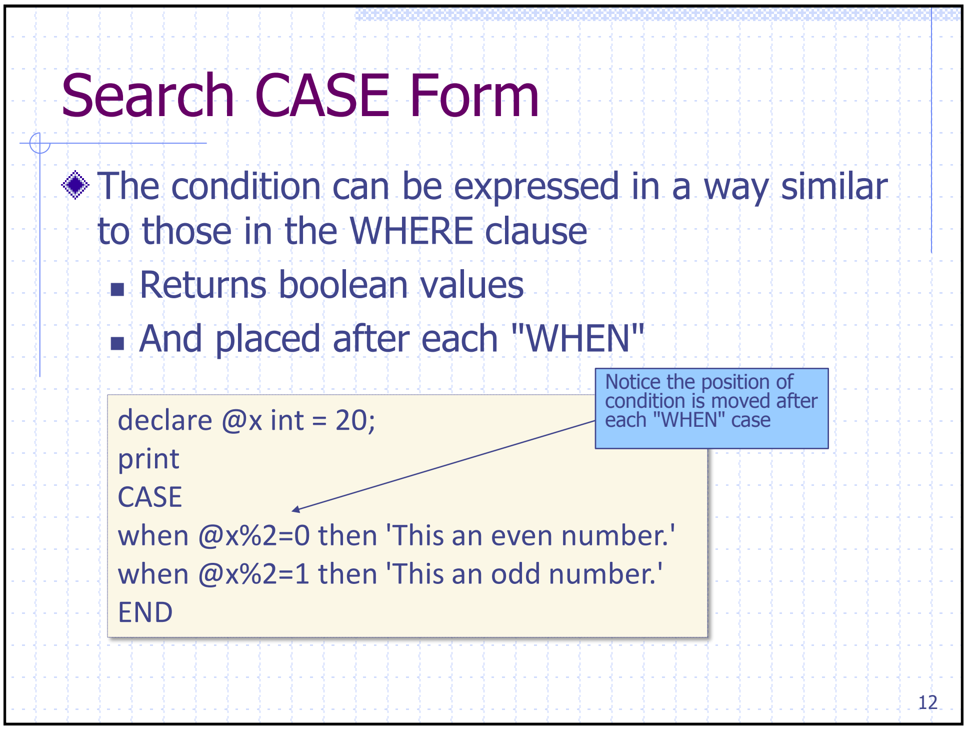
```
declare @x int = 20;
print
CASE @x%2
when 0 then 'This an even number.'
when 1 then 'This an odd number.'
END
```

A simple case expression needs a condition to be specified after the "CASE" keyword

The complete case block returns a value or expression, which can be used with other statements. In this example, it's part of the "print" statement.

Use "END" to close the case block

11



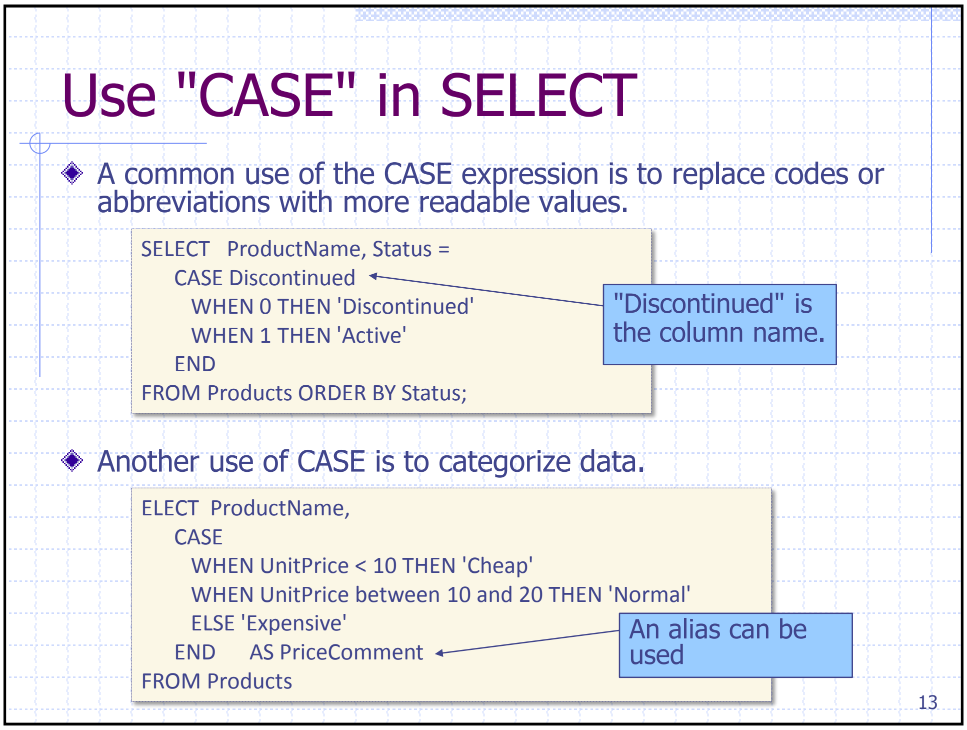
Search CASE Form

- The condition can be expressed in a way similar to those in the WHERE clause
  - Returns boolean values
  - And placed after each "WHEN"

```
declare @x int = 20;
print
CASE
when @x%2=0 then 'This an even number.'
when @x%2=1 then 'This an odd number.'
END
```

Notice the position of condition is moved after each "WHEN" case

12



Use "CASE" in SELECT

- A common use of the CASE expression is to replace codes or abbreviations with more readable values.

```
SELECT   ProductName, Status =
     CASE Discontinued
        WHEN 0 THEN 'Discontinued'
        WHEN 1 THEN 'Active'
     END
FROM Products ORDER BY Status;
```

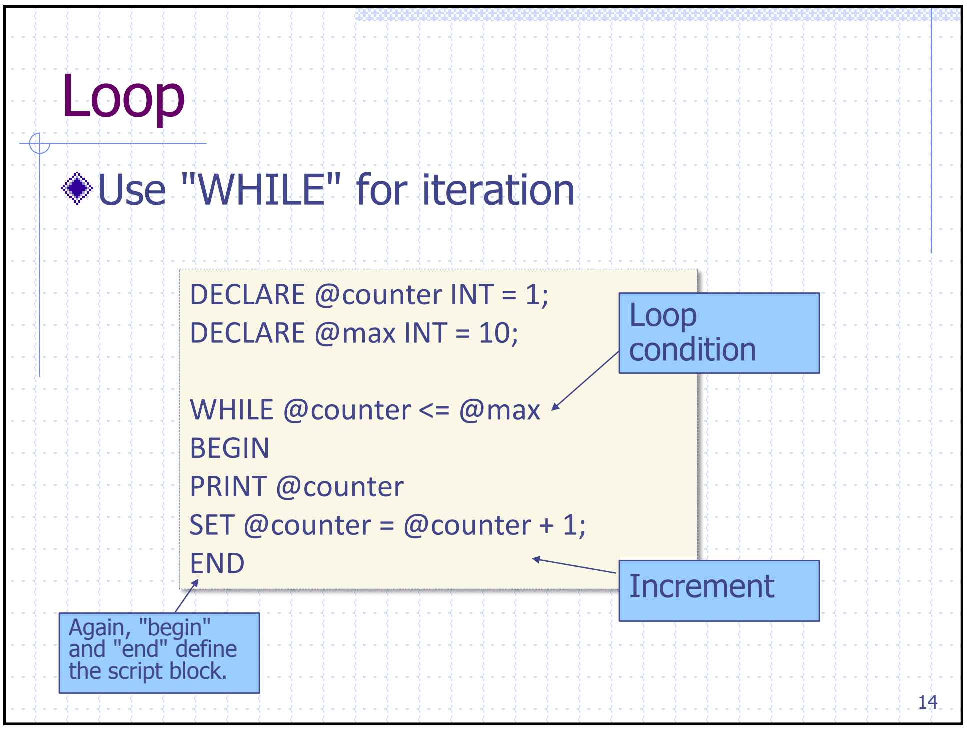
"Discontinued" is the column name.

- Another use of CASE is to categorize data.

```
ELECT  ProductName,
     CASE
        WHEN UnitPrice < 10 THEN 'Cheap'
        WHEN UnitPrice between 10 and 20 THEN 'Normal'
        ELSE 'Expensive'
     END     AS PriceComment
FROM Products
```

An alias can be used

13



Loop

- Use "WHILE" for iteration

```
DECLARE @counter INT = 1;
DECLARE @max INT = 10;

WHILE @counter <= @max
BEGIN
PRINT @counter
SET @counter = @counter + 1;
END
```

Loop condition

Increment

Again, "begin" and "end" define the script block.

14

Stored Procedure

- Stored procedures are small programs in the relational database management systems
  - Can be called (executed) in other applications
- Stored procedures group a number of statements together for reuse
  - Like a "void" method in Java or C#

Create a Stored Procedure

```
CREATE PROCEDURE GetProductsByName
        -- Add the parameters for the stored procedure
        @keyword nvarchar(50)
AS
BEGIN
        SELECT * FROM Products
        WHERE ProductName like '%'+@keyword+'%'
END
GO
```

Parameters should be separated by ,

Call/Execute a SP

- Use "EXECUTE" OR "EXEC" to call SPs
- Parameters can be supplied
  - either by using value directly

```
EXECUTE GetProductsByName 'Tofu'
```

  - or by using @parameter_name = value

```
EXEC GetProductsByName @keyword='Tofu'
```

17

System Procedures

- System procedures are provided by SQL Server
  - Starting with sp_ (system procedure)
- Examples
  - Catalog SP
    - sp_databases
    - sp_tables
    - sp_statistics
  - Database Engine SP
    - sp_help
    - sp_spaceused
    - sp_attach_db
    - sp_who
- System procedure reference
  - http://msdn.microsoft.com/en-us/library/ms187961.aspx

18

Function

- Stored procedures do not return values directly; functions do.
  - Like non-void methods in Java or C#
- Functions can be used in
  - Stored procedures
  - Standard SQL queries: DDL, DML, DCL

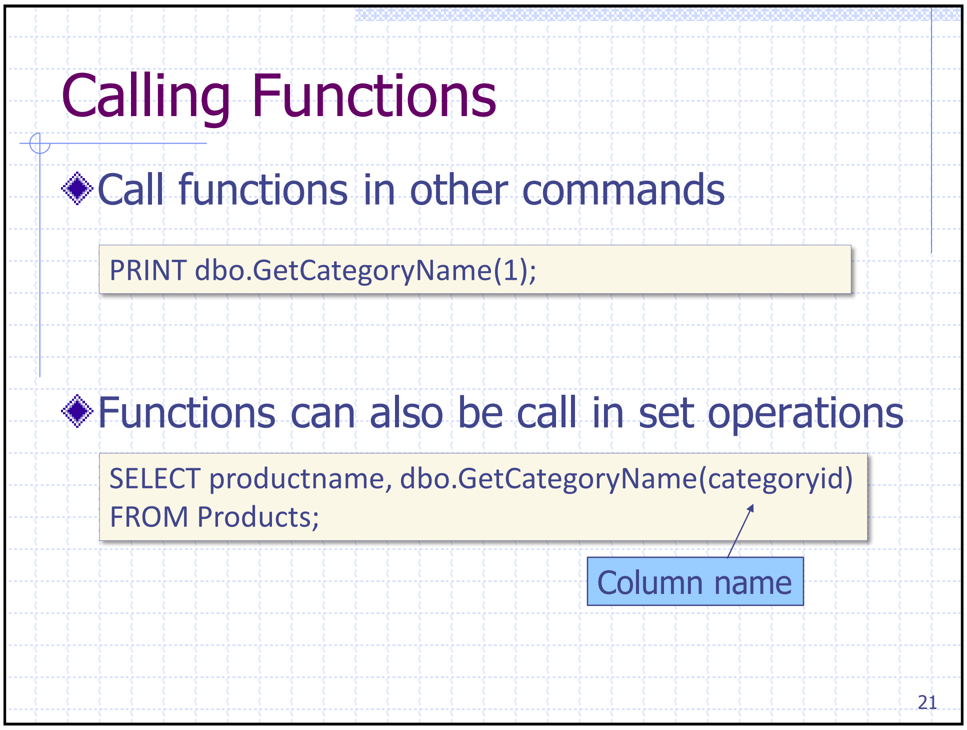
Create a Function

Parameters should be separated by ,

```
CREATE FUNCTION GetCategoryName(@cid int)
RETURNS varchar(50)
AS
BEGIN
        DECLARE @cname varchar(50);
        SELECT @cname=categoryname
                FROM Categories where CategoryID=@cid;
        RETURN @cname
END
```

Return data type

20



Calling Functions

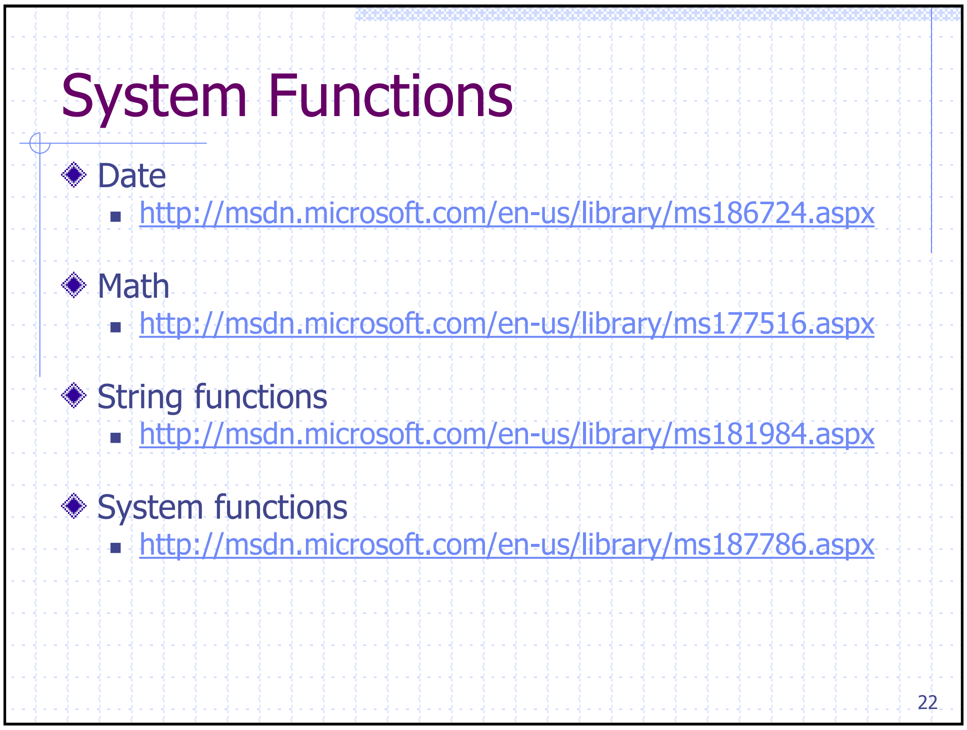
- Call functions in other commands

```
PRINT dbo.GetCategoryName(1);
```

- Functions can also be call in set operations

```
SELECT productname, dbo.GetCategoryName(categoryid)
FROM Products;
```

Column name



System Functions

- Date
  - http://msdn.microsoft.com/en-us/library/ms186724.aspx
- Math
  - http://msdn.microsoft.com/en-us/library/ms177516.aspx
- String functions
  - http://msdn.microsoft.com/en-us/library/ms181984.aspx
- System functions
  - http://msdn.microsoft.com/en-us/library/ms187786.aspx

Summary

- Key concepts
  - Script
  - T-SQL
  - Stored procedure, function
- Key skills
  - Write stored procedures and functions using T-SQL

More SQL Resources

- T-SQL reference
  - http://msdn.microsoft.com/en-us/library/bb510741(v=sql.105).aspx
- T-SQL functions
  - http://msdn.microsoft.com/en-us/library/ms174318.aspx