Functions

Topic 4 Lesson 3
Creating and using user-defined functions

Function: A DB Programming Object

- A function is executed typically in a SQL SELECT statement
- Allows you to create functions specific to the schema
- Only accepts IN arguments
- So NO keywords IN|OUT|INOUT
- Can only return a scalar value

```
CREATE FUNCTION function_name
(
     [parameter_name_1
  data_type]
     [, parameter_name_2
  data_type]...
)
RETURNS data_type
[NOT] DETERMINISTIC
{CONTAINS SQL|NO SQL|READS SQL
  DATA|MODIFIES SQL DATA}
sql_block
```

Function Example

EXAMPLE and SYNTAX

```
CREATE FUNCTION name
    ( argument1 argument1Type
    )
    RETURNS DECIMAL(11,2)
    DETERMINISTIC
BEGIN
-- Function definition
END
```

```
DELIMITER //

CREATE FUNCTION get_vendor_id
(
   vendor_name_param VARCHAR(50)
)
RETURNS INT
DETERMINISTIC READS SQL DATA
BEGIN
  DECLARE vendor_id_var INT;

  SELECT vendor_id
  INTO vendor_id_var
  FROM vendors
  WHERE vendor_name =
  vendor_name_param;

  RETURN(vendor_id_var);
END//
```

Function characteristics

CONTAINS SQL indicates that the routine does contain statements that read or write data. This is the default

NO SQL indicates that the routine contains no SQL statements.

READS SQL DATA indicates that the routine contains statements that read data, for example SELECT, but no statements that write data.

MODIFIES SQL DATA indicates that the routine contains statements that may change the database

DETERMINISTIC indicates it always produces the same result for the same input

NOT DETERMINISTIC indicates it may produce different result for the same input

Function to retrieve balance due

```
DELIMITER //

CREATE FUNCTION get_balance_due
(
   invoice_id_param INT
)
RETURNS DECIMAL(9,2)
DETERMINISTIC READS SQL DATA
BEGIN
  DECLARE balance_due_var DECIMAL(9,2);

  SELECT invoice_total - payment_total - credit_total
  INTO balance_due_var
  FROM invoices
  WHERE invoice_id = invoice_id_param;

  RETURN balance_due_var;
END//
```

```
SELECT vendor_id, invoice_number,
       get_balance_due(invoice_id) AS balance_due
FROM invoices WHERE vendor_id = 37
```

Benefits from a DB Function

- Define schema specific operations on the data
- Easier to maintain code, since the code is stored once in the database as opposed to duplicated in different applications
- Save coding time since the function is written once and can be used by all developers

Calling a user-defined function

```
SELECT
  invoice_number,
  invoice_total
FROM invoices
WHERE vendor_id =
  get_vendor_id('IBM')
  ;
```

Characteristics

```
DETERMINISTIC
NOT DETERMINISTIC
READS SQL DATA
MODIFIES SQL DATA
CONTAINS SQL
NO SQL
```

Example of NON-DETERMINISTIC Function

```
DELIMITER //

CREATE FUNCTION
  rand_int()
RETURNS INT
NOT DETERMINISTIC
NO SQL
BEGIN
   RETURN ROUND(RAND()
  * 1000);
END//
```

Default is all functions are NOT DETERMINISTIC

Make sure to use the keyword DETERMINISTIC, if indeed your function returns the same value for the same input

Differences between a function, procedure

- A procedure does not return a value. Instead, it is invoked with a CALL statement to perform an operation such as modifying a table or processing retrieved records.
- A function is invoked within an expression and returns a single value directly to the caller to be used in the expression.
- You cannot invoke a function with a CALL statement, nor can you invoke a procedure in an expression.

Summary

Functions allow you to return a single value from your database.

It is instantiated from a SQL SELECT command