

SQL User Defined Code

Kathleen Durant
CS 3200

User Session Objects

- Literals
 - Text single quoted strings
 - Numbers
- Database objects: databases, tables, fields, procedures and functions
 - Can set a default database/schema
 - Can also provide full context: *database.table.field*
 - System defined functions
 - User defined functions and procedures
- User session Variables
 - Allows you to store the results of a query
 - Can be passed to a function or another query

User Session Variables

- Variable are accessible by using the @ operation
 - @variablename
- Use set to assign a value to a variable
 - Set @var = 1; or @var := 1;
 - Variables not assigned a value has a default value of NULL
 - Can use other methods to assign a variable value
- Data type for a variable is determined by the last assigned value
- User variables can be assigned a value from a limited set of data types: integer, decimal, floating-point, binary or non-binary string, or NULL value
 - Assignment of other types are converted to one of the above listed types

Variable limitations

- User variable defined by one client cannot be seen or used by other clients
- All variables for a given client session are automatically freed when that client exits
- A select expression is evaluated when it is sent to the client
 - **Do not expect a variable to be evaluated in a subquery**
 - All levels of the subquery will have the same value for the variable
- User variables are intended to provide data values to a SQL statement
 - They cannot provide table name , field name, or command literal to a query
 - EXCEPTION: Prepared statements

Variable limitations

- User variables may be used in most contexts where expressions are permitted
 - Exception: limit
- Do not assign a value to and read the value of the same variable within a single statement
 - The value is the initial value of the variable
 - Exception the SET command

Control flow statements

- IF expression THEN statement
ELSEIF statement2
ELSE staatement3;
- CASE WHEN EXPRESSION THEN ...
WHEN EXPRESSION2 THEN ...
ELSE END;
- WHILE expression DO
END WHILE;
- REPEAT statements
UNTIL expression
- END REPEAT;

Four types of stored programs

- Stored procedures
 - An executable database object that contains a block of procedural SQL code
 - Use parameters to pass values to or from the procedure to the call program
 - Use the call program to execute a procedure
 - Can make changes to the database
 - Can return a result set
- Stored functions
 - A user-defined function is an executable database object that contains a block of procedural SQL code
 - Functions can only return a scalar or single value
 - Function can accept only IN parameters
 - Cannot make changes database
- Trigger
- Event

Stored procedure

- Is executed using the CALL statement
- Can return a result set
- Can pass value via the Arguments
 - IN – argument used as input variable (default argument type)
 - OUT – argument used as output variable
 - INOUT – argument users as input and output variable
- Example:

```
CREATE PROCEDURE (IN val1, OUT val2, INOUT val3)
BEGIN
DECLARE local_var var_type;
END
```


Stored procedure benefits

- Operations are performed uniformly
- Easier to maintain since the code is stored once in the database as opposed to duplicated in different applications
- Traffic is reduced between the client and the server since the stored procedure is executed on the server
- Security is enhanced – since clients can be granted fewer data base objects permission and still retrieve the data it needs

Procedure Example

```
DELIMITER $$  
CREATE PROCEDURE counter()  
BEGIN  
  DECLARE x INT; -- example of DECLARE  
  SET x = 1; -- EXAMPLE OF SET  
  WHILE x <= 5 DO -- WHILE LOOP  
    SET x = x + 1;  
  END WHILE; -- CLOSE OF WHILE LOOP  
  SELECT x; -- 6 -- THIS WILL PRINT THE VALUE OF THE VARIABLE  
END  
$$ DELIMITER ;
```

Cursors

- Procedures can call the SELECT statement
- SELECT statement can return a variable sized result set
 - Multiple rows
- Procedure or application code uses a cursor to walk through each of the returned records one at a time
- DECLARE cursorname CURSOR FOR
- DECLARE CONTINUE HANDLER FOR ERROR error_handler;
 - Error is thrown when you try to read beyond the last record associated with a cursor
 - Define a handler for this error to continue procedure control flow after reading all records

FUNCTION

- Is executed typically in a SQL 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
- Example
- `CREATE FUNCTION name
 (argument1 argument1Type
)`

`RETURNS DECIMAL(11,2)`

`BEGIN`

`END`

Function benefits

- Define schema specific operations on the data
- Easier to maintain 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

Prepared statements

- Can create a SQL statement where the checked values vary
- Can protect the database against SQL injection if the arguments are defined as values to check instead of as free form SQL code
- Less overhead for parsing the statement each time it is executed. Statement is set up just change the input values
- Use PREPARE to prepare the SQL statement
 - Inserts the arguments
 - `PREPARE preparestatement FROM statementstring;`
- Use EXECUTE to execute the command
 - `EXECUTE preparestatement;`
- Use DEALLOCATE to free resources associated with the statement