

# microgen

## *DBClarity Developer™*



## Product Overview and Examples

SQL Procedures

## SQL Procedures

- SQL Procedures
- SQL Procedure Blocks

- Typical stored procedures are complex algorithms
  - Constructs not available in modern programming languages
  - Procedural code can easily become problematic
- Cases where a whole routine is implemented as one complex SQL statement is covered by using SQL Rules, however when two tables need to be updated, using a stored procedure is unavoidable

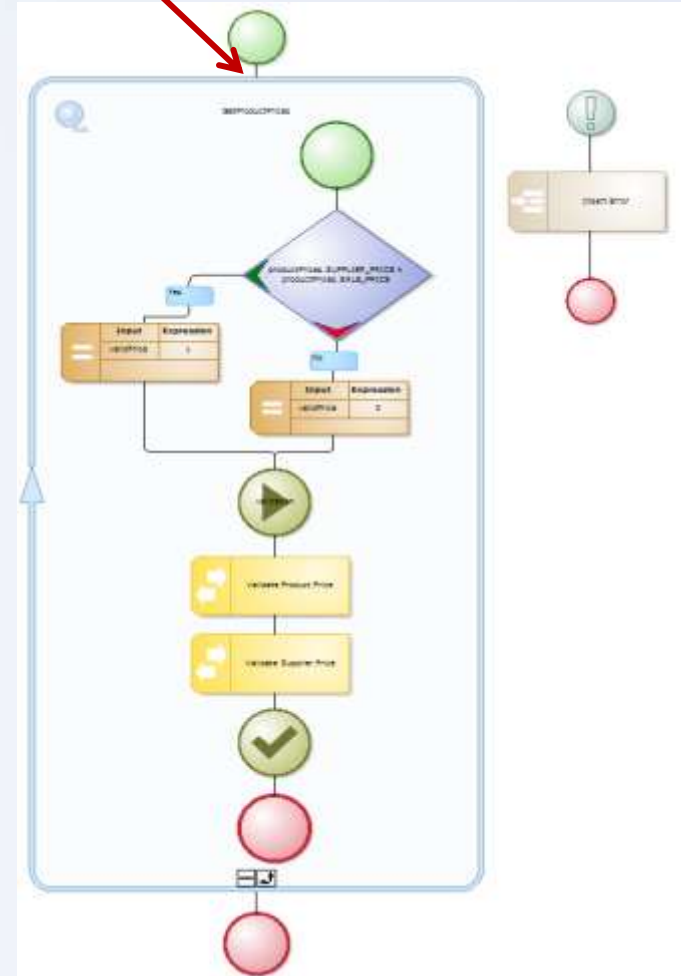
# SQL Procedures

Developers writing the code

Outline the algorithm in the SQL Procedure

```

1 CREATE PROCEDURE PriceValidation
2 BEGIN
3     DECLARE validPrice DECIMAL(38);
4     DECLARE errorCode DECIMAL(38) DEFAULT 1;
5     DECLARE errorMessage VARCHAR(64000) DEFAULT 'Unknown Error';
6     DECLARE EXIT HANDLER FOR SQLEXCEPTION
7     CALL InsertError(errorCode, errorMessage);
8     GetProductPrices:
9     FOR productPrices AS
10    SELECT
11        SUPPLIER_PRODUCT.SUPPLIER_ID AS SUPPLIER_ID,
12        PRODUCT.PRODUCT_ID AS PRODUCT_ID,
13        PRODUCT.UNIT_PRICE AS SALE_PRICE,
14        SUPPLIER_PRODUCT.SUPPLIER_PRICE AS SUPPLIER_PRICE
15    FROM
16        PRODUCT
17        INNER JOIN SUPPLIER_PRODUCT SUPPLIER_PRODUCT ON
18            PRODUCT.PRODUCT_ID = SUPPLIER_PRODUCT.PRODUCT_ID
19    DO
20        IF productPrices.SUPPLIER_PRICE < productPrices.SALE_PRICE THEN
21            SET validPrice = 1;
22        ELSE
23            SET validPrice = 0;
24        END IF;
25        UPDATE PRODUCT PRODUCT
26        SET
27            VALID_PRICE = validPrice
28        WHERE
29            PRODUCT.PRODUCT_ID = productPrices.PRODUCT_ID;
30        UPDATE SUPPLIER_PRODUCT SUPPLIER_PRODUCT
31        SET
32            VALID_PRICE = validPrice
33        WHERE
34            (SUPPLIER_PRODUCT.PRODUCT_ID = productPrices.PRODUCT_ID)
35            AND
36            (SUPPLIER_PRODUCT.SUPPLIER_ID = productPrices.SUPPLIER_ID);
37        COMMIT;
38    END FOR GetProductPrices;
39 END
    
```



# SQL Procedures

- The SQL Procedure diagram represents database stored procedure code and is used to generate a database create procedure statement
- The blocks represent particular code constructs
- Vertically oriented and represents the control flow –
  - Blocks representing simple atomic operations are executed one by one in the order defined by the links
  - There is no data on the links
- The diagram starts with the green circle (Begin Block) and ends with the red one (End Block)

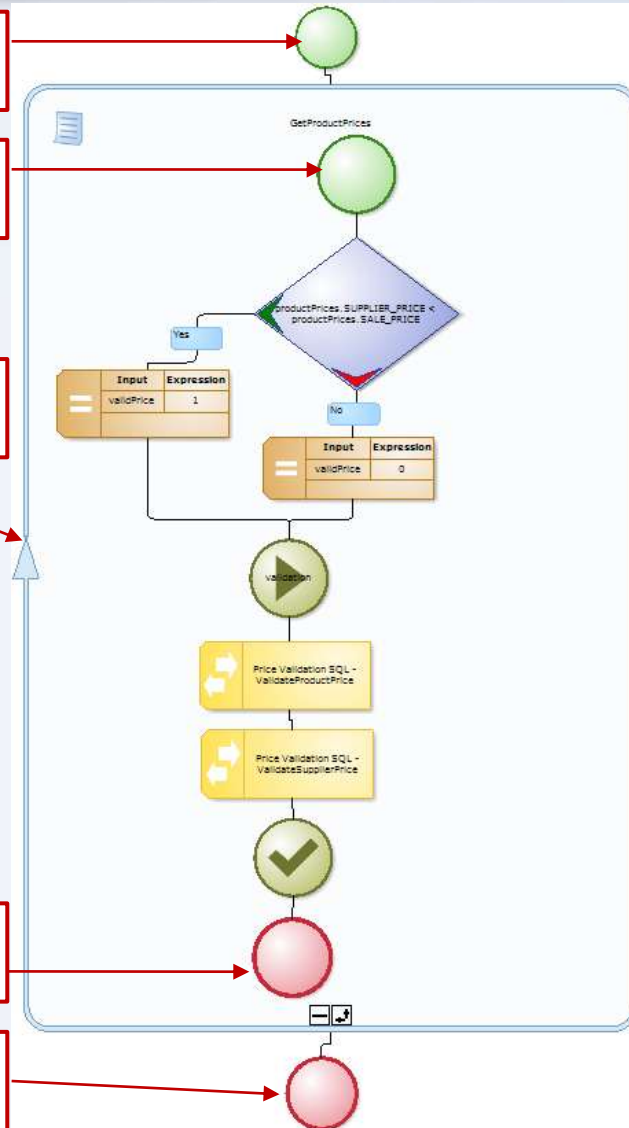
The Procedure begins here

The Loop begins here

The Loop sub-region

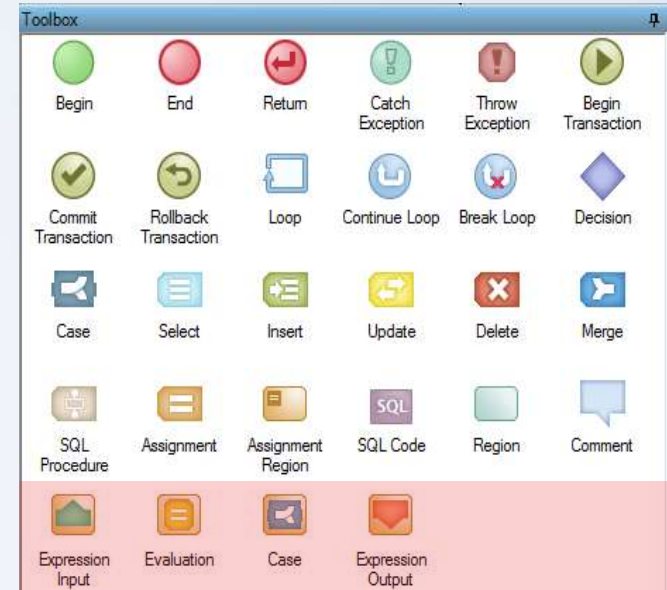
The Loop ends here

The Procedure ends here



# SQL Procedures

- The data is stored in the variables declared in the first Begin Block
- The input/output parameters are defined in the first Begin Block
- The diagram may have sub-regions (Loop or Region Block)
- The flow within the sub-regions must also start with the Begin Block and be finished with the End Block
- SQL Procedures are deployed as a stored procedure, function or trigger to any supported database
- SQL Procedures Toolbox
  - The blocks define the order of execution of the code.



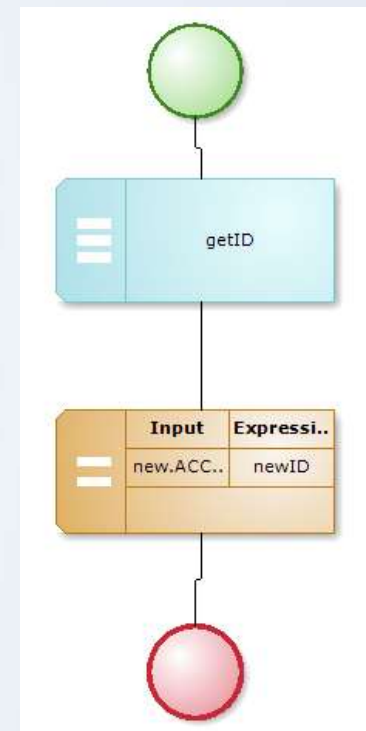
- **Begin Block**

- Controls the start of the flow
- Functions as a Procedure or Database Trigger
- The first executed block is the one after the Begin Block
- Defines the input and output parameters and accessible variables
- SQL Procedure can have one Begin Block

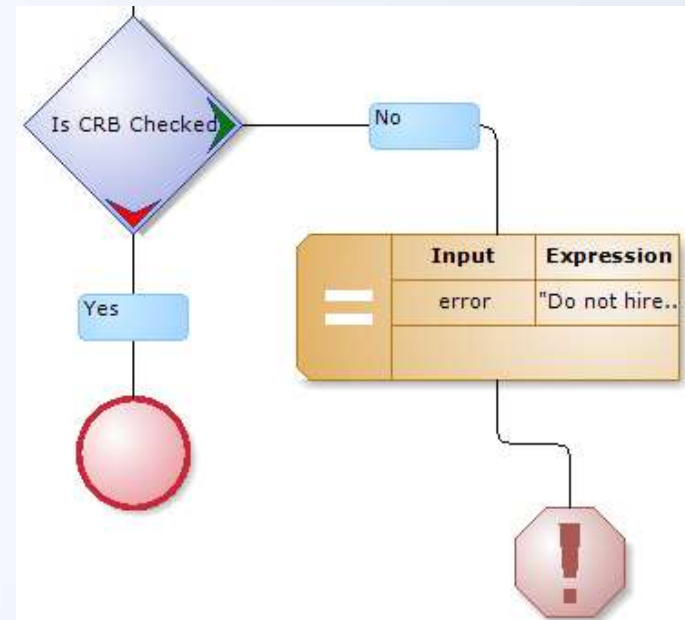
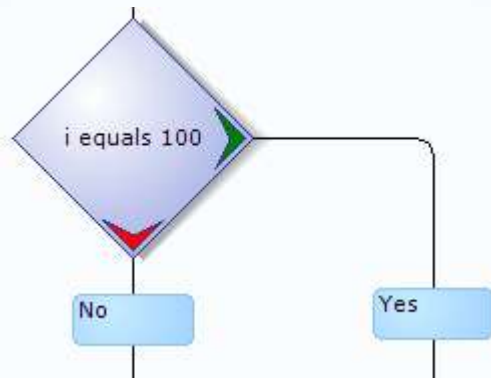
- **End Block**

- Controls the end of the flow
- Execution of the procedure ends at this block
- Multiple End Blocks are allowed

```
□ BEGIN
  SELECT
    seq.NEXTVAL
  INTO
    newID
  FROM DUAL
  ;
  new.ACCOUNT_ID := newID;
□ END;
```



- Decision Block
- Conditional Block
- Evaluates conditions and directs the flow according to the result which is True or False
- Can function as a Decision or an Exception



## Exception Handling Blocks

- The exception handling routine succeeded by handlers based on the normal control flow between a series of Decision Blocks
- Exception handling blocks in a region block are local and throw/catch exceptions with that region only



### Catch Exception Block

- Beginning of the exception handling – intercepts the control flow when an exception is thrown
- Only one Catch Exception Block can be in a region and one in the main SQL Procedure diagram



### Throw Exception Block

- The control goes to the associated Catch Exception Block or the procedure ends if none is defined

## Case Block (for SQL Procedures and SQL Expressions)



- Conditional block that functions in two modes:

### Simple Case Mode:

Compares one expression to another using a WHEN Clause.

Where result is True, the expression is used in a THEN Clause.

If False, then ELSE.

### Searched Case Mode:

Evaluates expression for each WHEN Clause.

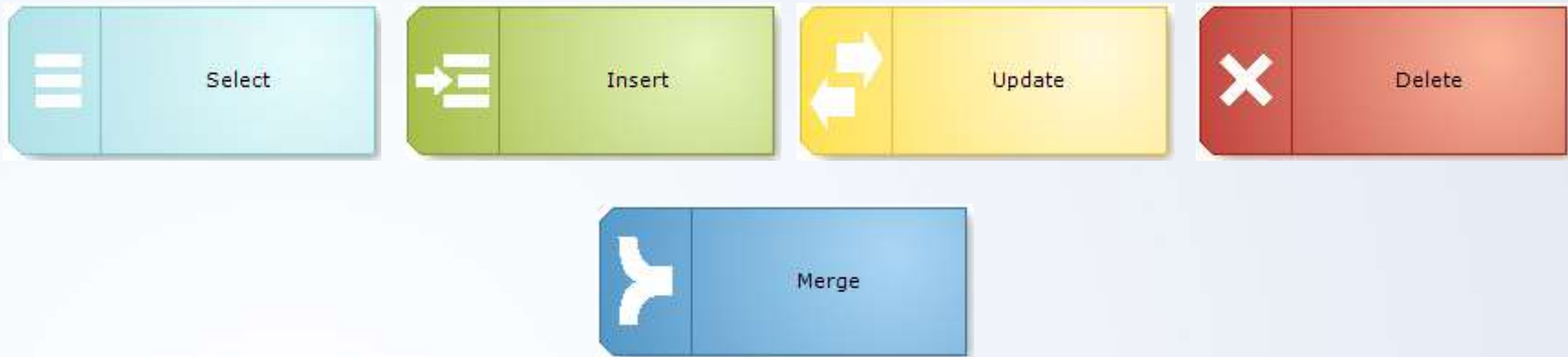
Returns the result expression of the first expression that is True.

If no expressions are True, then ELSE.

```
CASE A
  WHEN 1 THEN A
  WHEN 2 THEN B
  ELSE C
END
```

```
CASE
  WHEN A = B + C THEN A
  WHEN A > 0 THEN B
  ELSE C
END
```

## Command Blocks



- The Select Block uses the SELECT statement to set values of the variables or to lock the rows in a table
- Insert, Update, Delete and Merge Blocks modifies data in the tables
- All blocks have incoming and outgoing links with the control flowing through the block
- Each block contains a Query Container to embed a SQL Rule

## Transaction Blocks



### Begin Transaction Block

- Starting point of a local transaction
- Defines the “save point” to which the rollback can be performed



### Commit Transaction Block

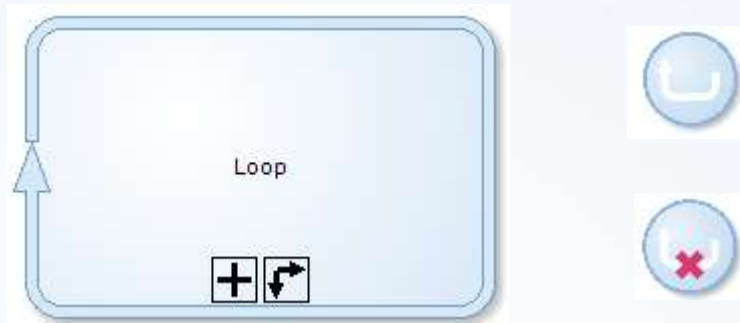
- Represents the end of a successful implicit or explicit transaction



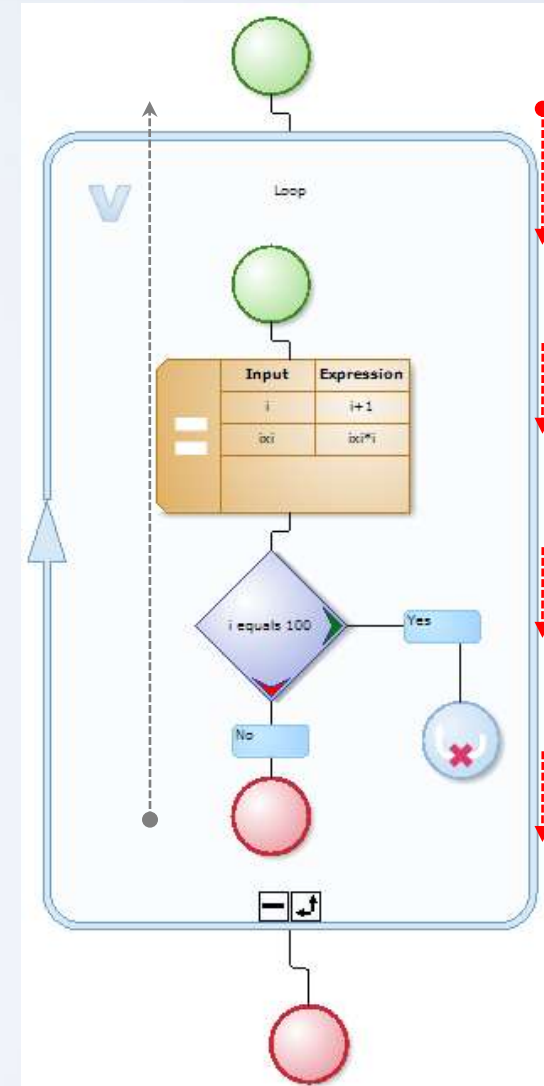
### Rollback Transaction Block

- Rolls back the transaction to the “save point” inside the transaction

## Loop Blocks



- Embedded code called repeatedly
- Iterates over the data-set returned by the SELECT statement (loops through multiple rows in a table) until a certain condition is met



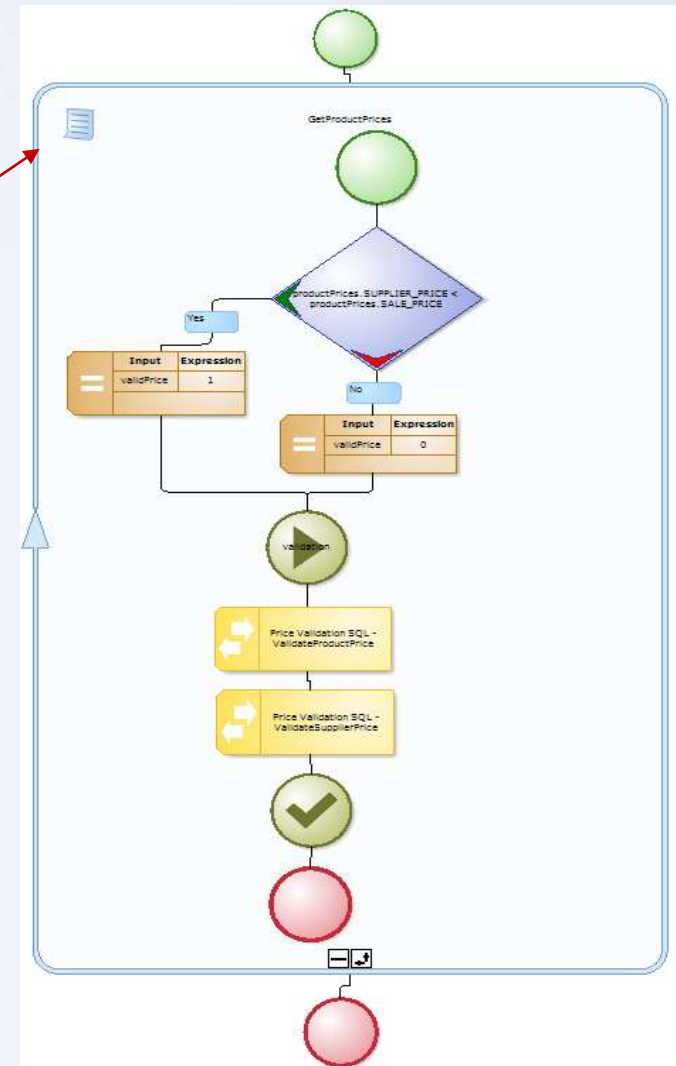
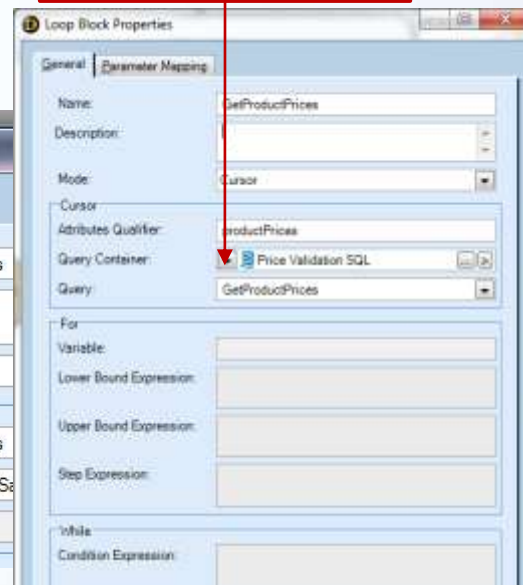
# Loop Over SQL SELECT Result-Set

**microgen**  
DBClarity Developer™

- The Loop sub-region executes the embedded sub-diagram while reading the data from a database -using a SELECT statement.
- The sub-diagram is executed once for each row that is read.
- The SELECT statement comes from:
  - EDF Query
  - SQL Rule

The EDF Query is used for this Loop

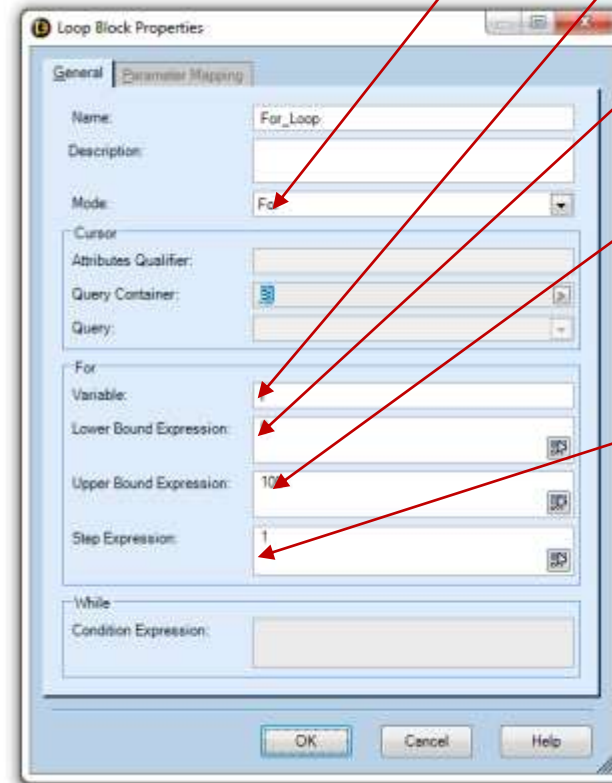
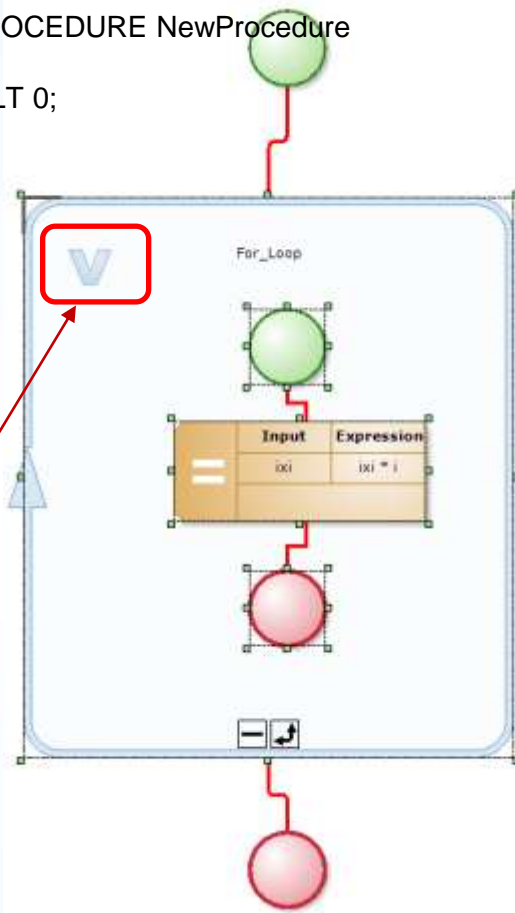
The SQL Rule is used for this Loop



# Loop using a Variable and a Step

A loop with an automatically incremented variable

```
CREATE OR REPLACE PROCEDURE NewProcedure
AS
  ixi NUMBER(38) DEFAULT 0;
BEGIN
  DECLARE
    i NUMBER(38);
  BEGIN
    i := 0;
    <<Loop>>
    WHILE i <= 100 LOOP
      ixi := ixi * i;
      i := i + 1;
    END LOOP Loop;
  END;
END;
```



Type of Loop

The Loop variable

The variable initial value

The variable last value

The variable increment step

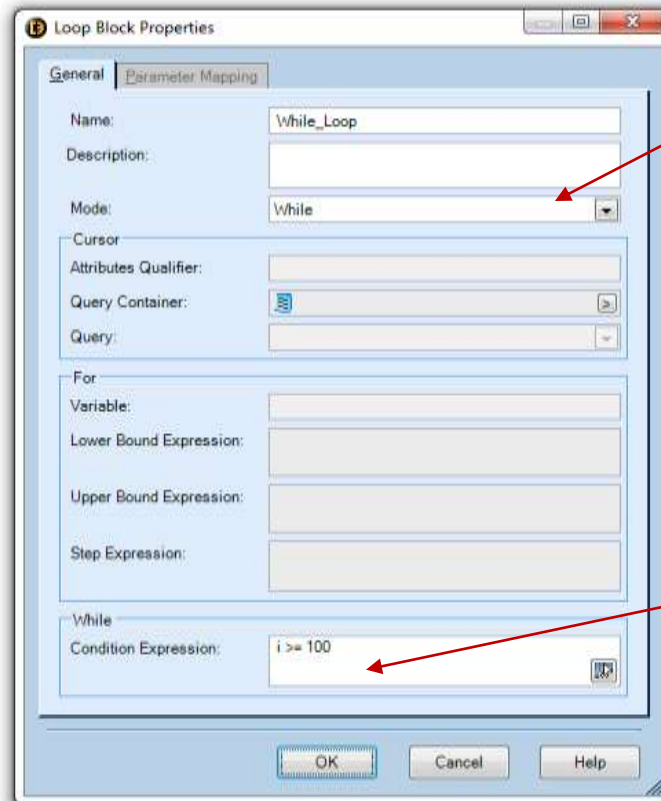
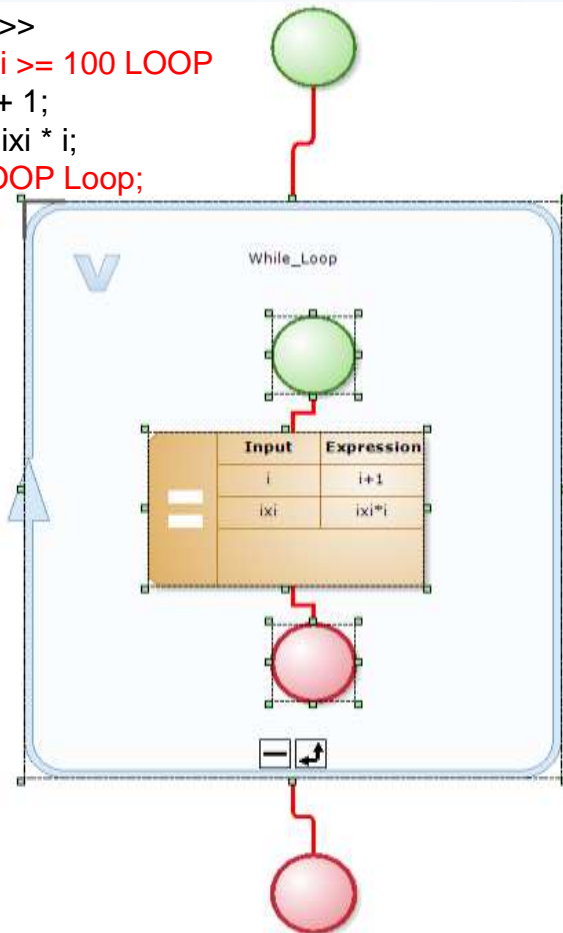
Variable Loop

# Loop using a Variable and Exit Condition microgen DBClarity Developer™

```

CREATE OR REPLACE PROCEDURE NewProcedure
AS
  ixi NUMBER(38) DEFAULT 0;
  i NUMBER(38) DEFAULT 0;
BEGIN
  <<Loop>>
  WHILE i >= 100 LOOP
    i := i + 1;
    ixi := ixi * i;
  END LOOP Loop;
END;

```

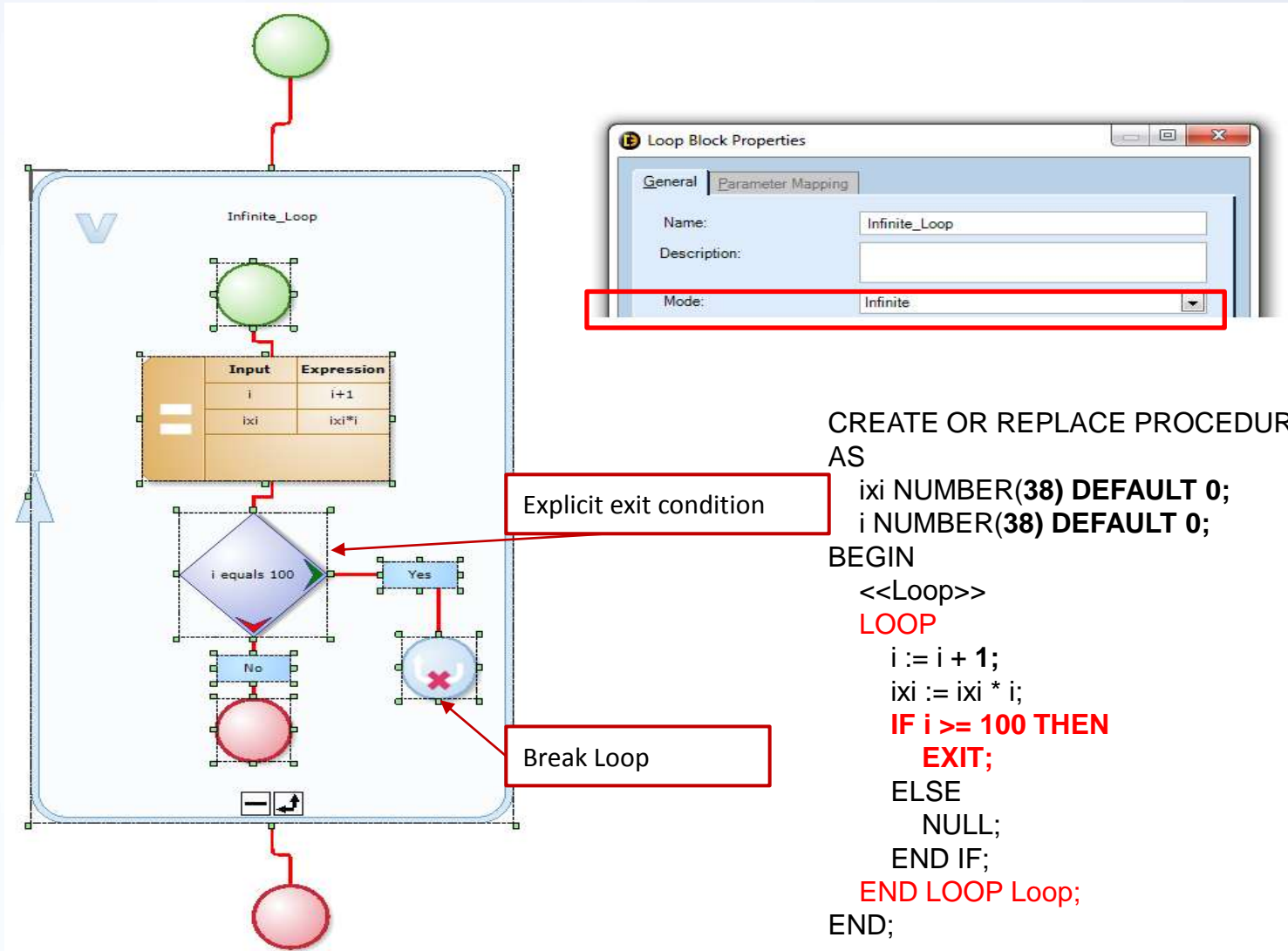


Type of Loop

The Loop exit condition does not depend on any particular variable

# Infinite Loop

**microgen**  
DBClarity Developer™



## Assignment Block

- Assigns values to user-defined variables

The screenshot shows the 'Assignment Block Properties' dialog box. The 'Name' field is set to 'Assignment'. The 'Description' field is empty. Below the 'Assignments' section, there are four icons: a plus sign, a minus sign, a yellow arrow, and a grid icon. The 'Assignments' table is as follows:

Name	Expression
firstPass	0
customerId	SaleOrders.CUSTOMER_ID
saleDate	SaleOrders.ORDER_DATE
customerDailyTotal	customerDailyTotal + SaleOrders.TOTAL

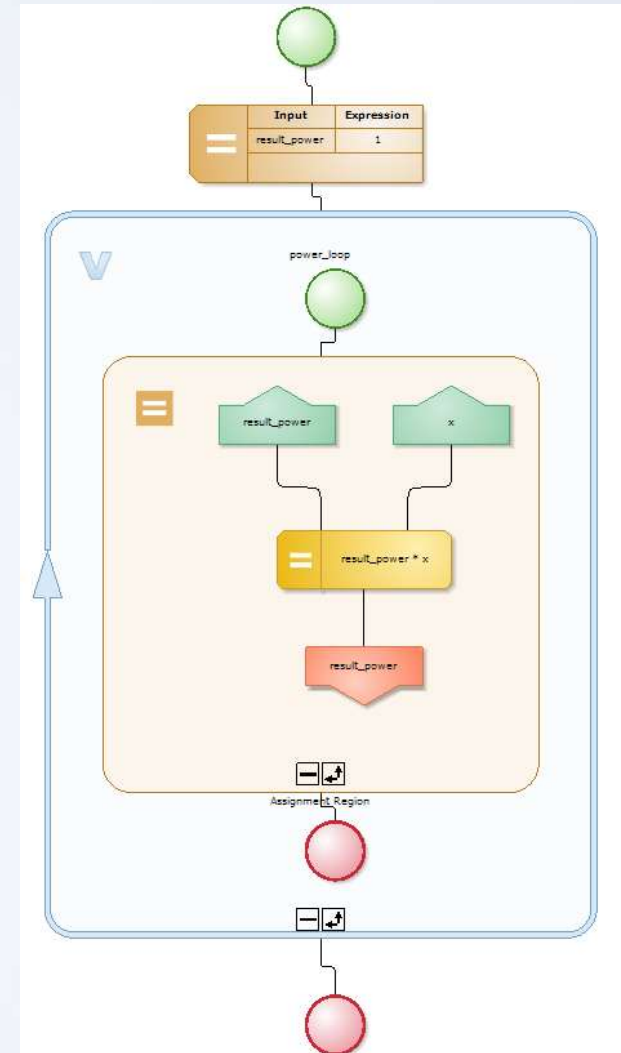
To the right of the dialog box, a larger table displays the configuration for the assignment block:

Input	Expression
firstPass	0
customerId	SaleOrders.CUSTOMER_ID
saleDate	SaleOrders.ORDER_DATE
customerDailyTotal	customerDailyTotal + SaleOrders.TOTAL

# SQL Procedures

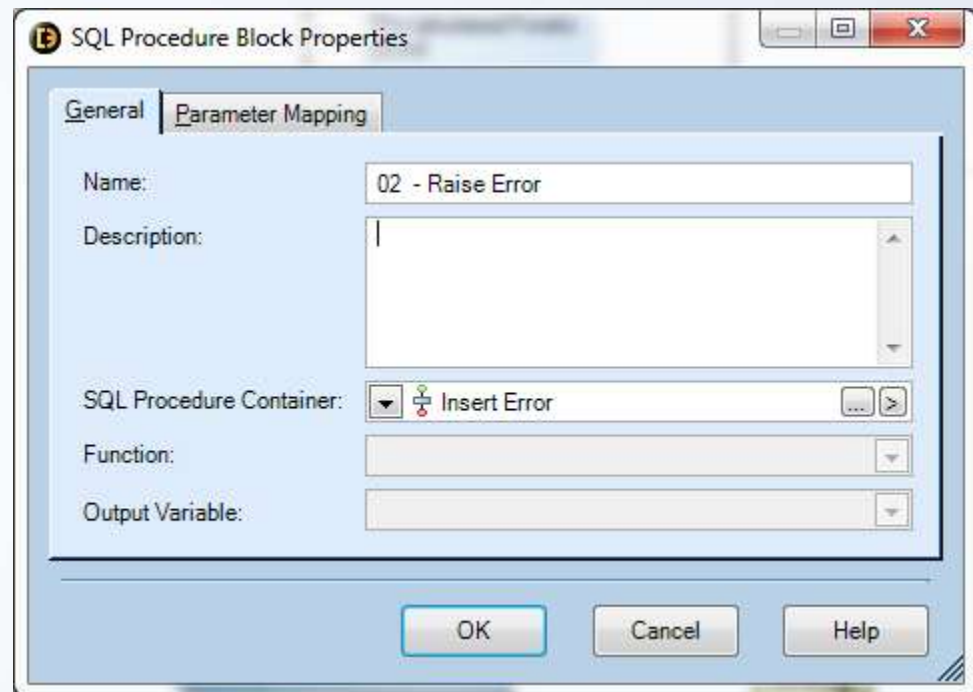
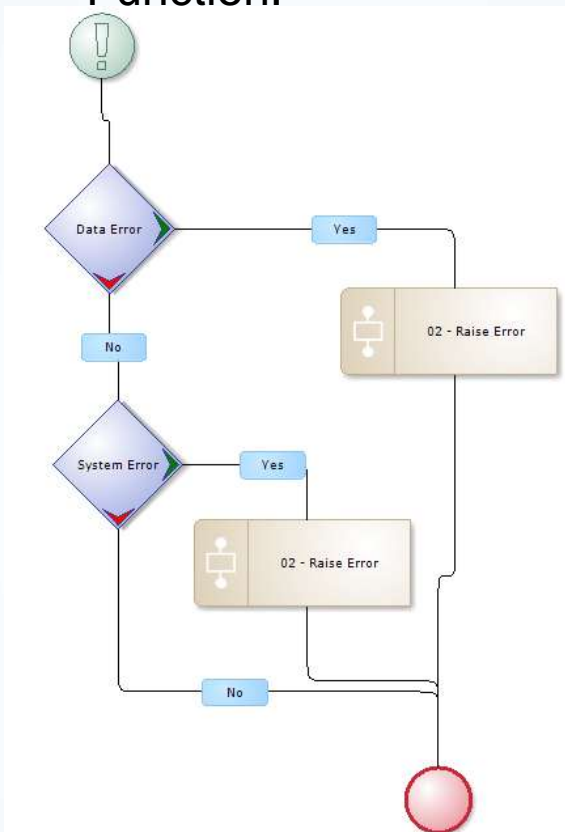
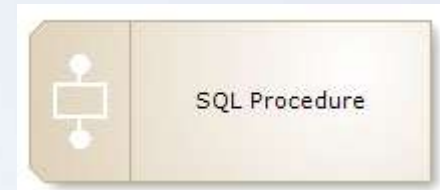
## Assignment Region

- Assigns values to user-defined variables using the SQL Expression diagram



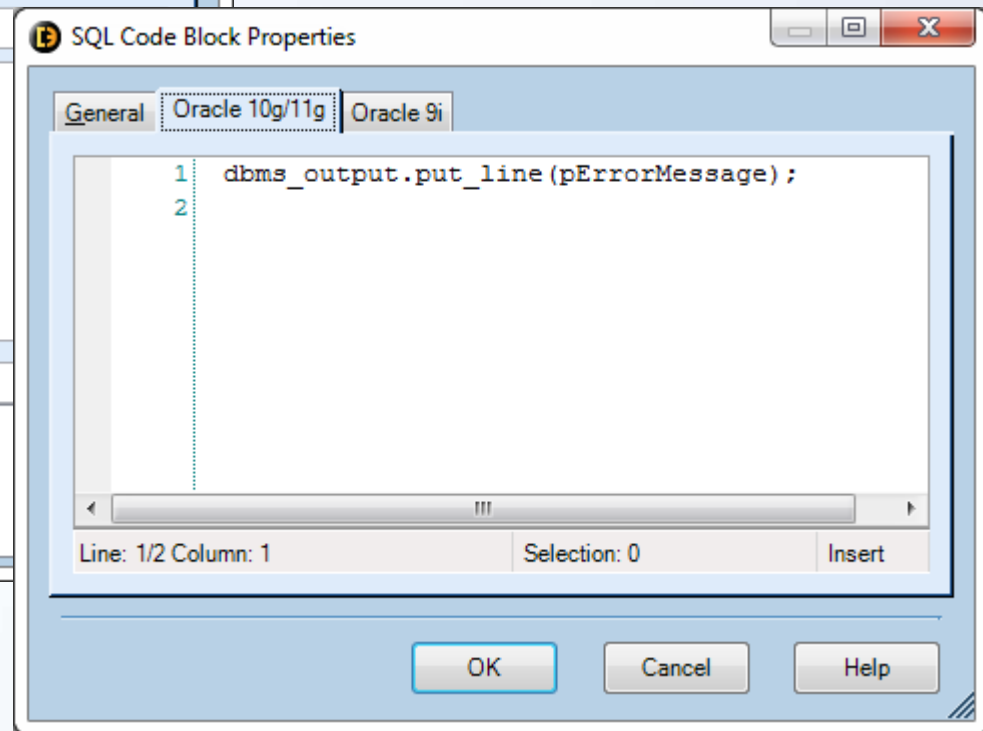
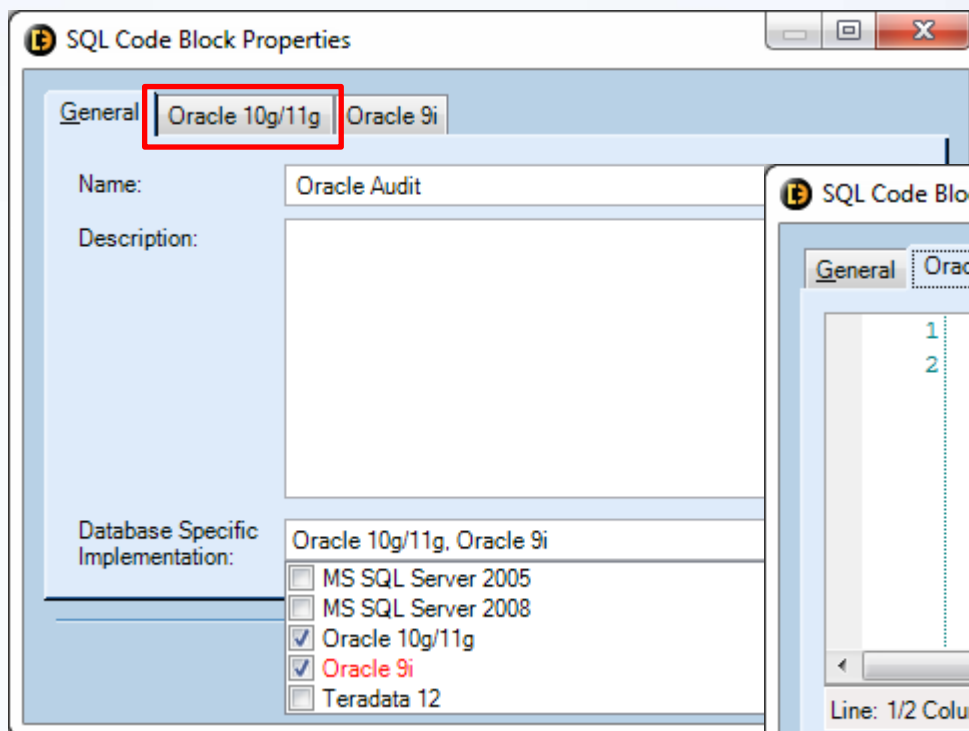
## SQL Procedure Block

- Calls a stored procedure in the SQL Rule, another defined SQL Procedure or an External Call Format Function.



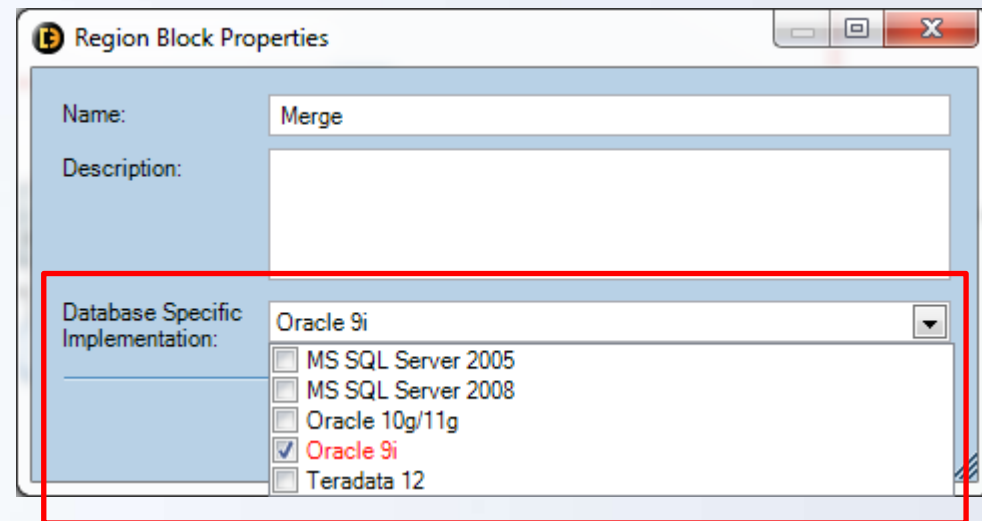
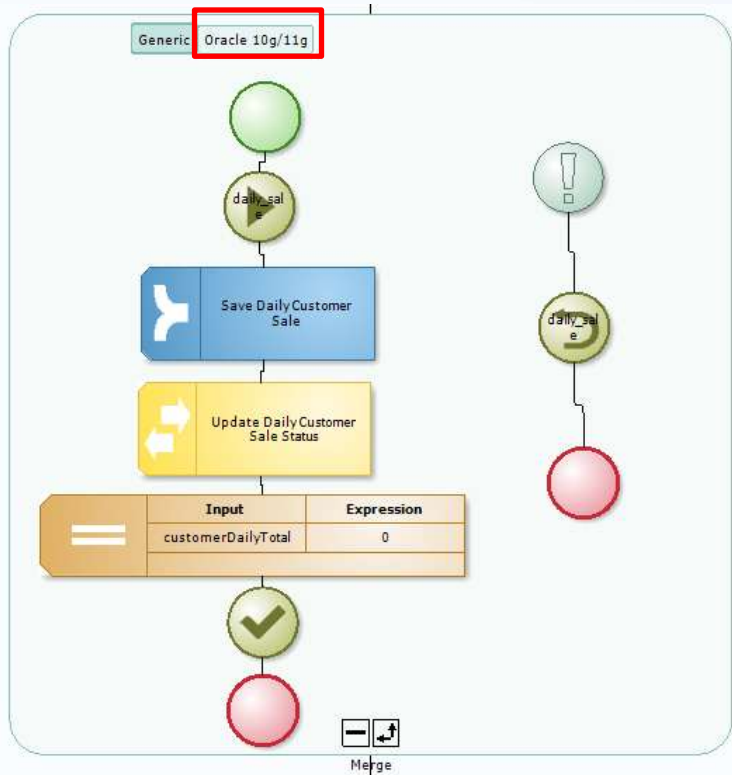
## SQL Code Block

- Executes SQL code specific for supported databases.



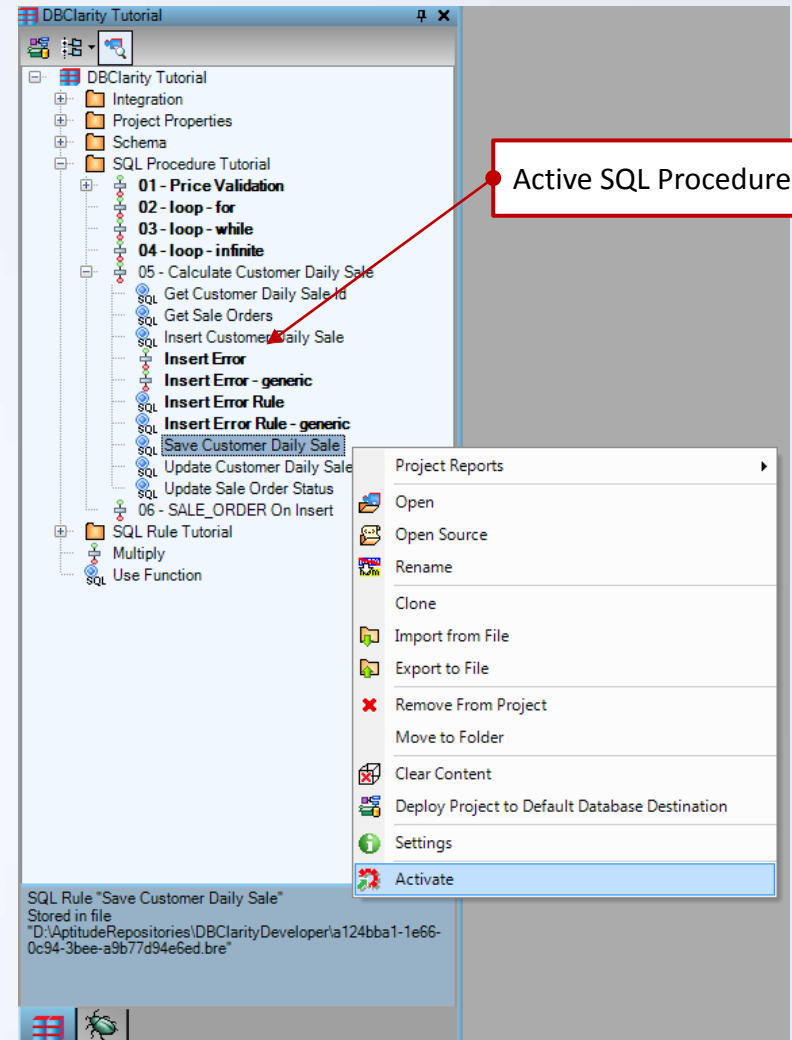
## Region Block

- Similar features to the SQL Rule region blocks
- Represents a database-specific region executed dependently on the database



# SQL Procedure – Deployment

- SQL Procedure is a definition of a stored procedure – deployed to the database if active
- The definitions of stored procedures are executed while deploying the project to a database



# More Information

- Download the product and find more information, including product workshops and training videos, visit the Microgen website [www.microgen.com/dbclarity](http://www.microgen.com/dbclarity)
- Send your product questions and feedback to the Microgen team at [dbclarity.feedback@microgen.com](mailto:dbclarity.feedback@microgen.com)
- Contact Microgen Support for all DBClarity Developer related support queries at [dbclarity.support@microgen.com](mailto:dbclarity.support@microgen.com)
- Follow DBClarity Developer on Twitter @**MCGN\_DBClarity**

This document is designed to provide a training overview of Microgen DBClarity Developer™. Information in this document is subject to change without notice and does not represent a commitment on the part of Microgen.

The information contained in this document is proprietary and confidential to Microgen plc and must not, therefore, be disclosed to any third-party without the express written permission of Microgen. In addition, no part of this document may be reproduced or transmitted in any form or by any means electronic or mechanical including photocopying, recording or information storage and retrieval systems, for any purpose other than the recipient's personal use without the express written permission of Microgen.