Data Selection and Filtering

(Part II)

by Ahmed Baraka

Introduction to Oracle Data Guard

In this lecture, we are going to talk about the basic concepts of Oracle Data Guard

Objectives

By the end of this lecture, you should be able to:

- Use built-in functions to perform column tests
- Use built-in functions to manipulate columns data

Oracle GoldenGate Built-in Functions

- Used for data transformation
- They have @ prefix
- Can be used for:
  - Column tests and evaluations
  - Date format and mappings
  - String and number conversion
  - Strings manipulation
  - Strings or numbers comparison

Built-in Functions: Testing Data

Function	Description
`If`	One value returned based on a conditional statement
`Eval`	One value returned based on a series of tests
`Case`	One value returned based on a series of tests
`ColStat`	Indicator that a column is missing, null, or invalid.
`ColTest`	Tests if a column value is PRESENT, MISSING, NULL, or INVALID
`ValOneOf`	Returns TRUE if a column contains one of a list of values

`IF` Function

- Returns a value based on a condition
- Syntax:

```
@IF (condition, value_if_non-zero, value_if-zero)
```

- Example:

```
TOTAL_COL = @IF (TOTAL > 0, TOTAL, 0)
```

`ColStat` Function

- Indicator that a column is missing, null, or invalid
- Syntax:

```
@COLSTAT ({MISSING | NULL | INVALID})
```

- Examples:

```
ADDRESS = @COLSTAT (NULL)
```

```
UNIT_PRICE = @IF (PRICE < 0, @COLSTAT(NULL))
```

`ColTest` Function

- Indicator that a column is missing, null, or invalid
- Syntax:

```
@COLTEST (source_column, test_condition)
test_condition: PRESENT, NULL, MISSING, INVALID
```

- Example:

```
AMOUNT = @IF (@COLTEST (AMT, MISSING, INVALID), 0, AMT)
```

Testing Data Example


```
LARGE_QUANTITY =
 @IF (@COLTEST (QUANTITY, PRESENT) AND
     QUANTITY > 2000,
     QUANTITY, @COLSTAT (NULL))
```

Testing Data Examples


```
@CASE (PRODUCT_TYPE, '56', 'COMPUTER', '57', 'MONITOR')
```

```
@IF (@ValOneOf (STATE, 'CA', 'NY'), 'COAST', 'MIDDLE')
```

```
@EVAL (GRADE >= 90, 'EXCELLENT',
       GRADE < 90, 'LESS THAN EXCELLENT')
```

Built-in Functions: Handling Dates

Function	Description
`Date`	Returns dates and times in a variety of formats
`DateDiff`	Calculates the difference between two dates or datetimes
`DateNow`	Returns the current date and time

`@Date` Function

- Returns date in the provided format

```
DATE_COL = @DATE ('YYYY-MM-DD', 'YY', INV_YEAR, 'MM',
INV_MM, 'DD', INV_DD)
```

```
DATE_COL = @DATE ('YYYY-MM-DD HH:MI:00', 'YYMMDD',
INV_DATE, 'HHMI', INV_TIME)
```

```
DATE_COL = @DATE ('YYYY-MM-DD HH:MI:SS',
'YYYYMMDDHHMISS', NUMERIC_DATE)
```

Built-in Functions: Handling Strings

Function	Description
`Compute`	Returns the result of Arithmetic expressions
`NumStr`	Converts a string into a number
`StrCat`	Concatenates two or more strings
`StrCmp`	Compares two strings
`StrNCmp`	Compares two strings up to a certain number of characters
`StrEq`	Tests to see whether two strings are equal
`StrExt`	Extracts selected characters from a string
`StrFind`	Finds a string within a string
`StrLen`	Returns the length of a string

Built-in Functions: Handling Strings (cont)

Function	Description
`StrLTrim`	Trims the leading spaces
`StrRTrim`	Trims the trailing spaces in a column
`StrTrim`	Trims both leading and trailing spaces
`StrSub`	Substitutes one string for another within a column
`StrNum`	Converts a number into a string, with justification and zero-fill options
`StrUp`	Changes a string to uppercase

Handling String Examples


```
FULLNAME = @StrCat (LASTNAME, "," ,FIRSTNAME)
```

```
AREA_CODE = @StrExt (PHONE, 1, 3)
```

```
FILTER (@COMPUTE (UNIT_PRICE * QUANTITY) > 10000)
```

Built-in Functions: Miscellaneous

Function	Description
`GetEnv`	Returns information about the GoldenGate environment, trail file header, trail record header, last replicated operation, and lag
`GetVal`	Extracts values from a stored procedure or query so that they can be used as input to a FILTER or COLMAP
`Range`	Divides a workload into multiple groups of data
`Token`	Retrieves token data that is stored in the user token area of the Oracle GoldenGate record header

About `SQLEXEC` Function

- To execute SQL queries (including DMLs), stored procedures, or database commands
- Extend the capabilities of the Oracle GoldenGate
- Supports: Oracle, SQL Server, Teradata Sybase and DB2

SQLEXEC within a TABLE or MAP Parameters

- Support queries and stored procedures but not DB commands
- SQLEXEC can pass and accept parameters
- To execute a procedure:

```
SQLEXEC (SPNAME <sp_name>, [ID <logical_name,]
{PARAMS <param_spec> | NOPARAMS})
```

- To run a query or DML:

```
SQLEXEC (ID <logical_name>, QUERY ' <sql_query> ',
 {PARAMS <param_spec> | NOPARAMS})
```

A Procedure Example for SQLEXEC Function


```
CREATE OR REPLACE PROCEDURE GET_TITLE
(P_EMPLOYEE_ID IN VARCHAR2,
 P_TITLE OUT VARCHAR2)
BEGIN
 SELECT TITLE_COL INTO P_TITLE
 FROM EMPLOYEE_TITLES
 WHERE EMPLOYEE_ID = P_EMPLOYEE_ID;
END;
```

SQLEXEC Function: Using the Procedure Example


```
Map HR.EMPLOYEES, Target HR.EMPS, &
 SQLEXEC (SPNAME GET_TITLE, &
 PARAMS (P_EMPLOYEE_ID = EMPLOYEE_ID)), &
 ColMap (USEDEFAULTS, EMPLOYEE_ID = EMP_ID, &
 EMP_TITLE = @GETVAL(GET_TITLE.P_TITLE));
```

- In multiline code, ampersand is required

SQLEXEC Function: Running a Query

- Use `@GETVAL` to retrieve the return parameter

```
Map HR.EMPLOYEES, Target HR.EMPS,
 SQLEXEC (id GET_TITLE,
  QUERY ' SELECT TITLE_COL FROM EMPLOYEE_TITLES
       WHERE EMPLOYEE_ID = :V_EMPLOYEE_ID ',
       PARAMS (V_EMPLOYEE_ID = EMPLOYEE_ID)),
       COLMAP (USEDEFAULTS, EMPLOYEE_ID = EMP_ID,
       EMP_TITLE = @GETVAL(GET_TITLE.TITLE_COL));
```

SQLEXEC as a Standalone Statement

- When used as a standalone parameter statement in the Extract or Replicat parameter file, SQLEXEC can execute a stored procedure, query, or database command.

Purpose	Syntax
Execute a stored procedure	`SQLEXEC 'call proc_name()'`
Execute a query	`SQLEXEC ' sql_query '`
Execute a database command	`SQLEXEC 'database_command'`

SQLEXEC as a Standalone Statement Examples


```
SQLEXEC 'call order_count ()'
SQLEXEC ' select x from dual '
SQLEXEC 'SET TRIGGERS OFF'
SQLEXEC 'call order_count ()' EVERY 30 SECONDS
```

Summary

In this lecture, you should have learnt how to:

- Use built-in functions to perform colmun tests
- Use built-in functions to maniplulate columns data