Oracle SQL to Denodo VQL Quick Reference

Revision 20211103

Denodo Inc.: 525 University Avenue, Suite 31, Palo Alto, CA 94301. USA www.denodo.com

Goal

This document is a quick reference for migrating Oracle SQL to Denodo VQL. The document is aimed at administrators and developers that want to efficiently migrate their existing Oracle queries to Denodo VQL queries.

Content

The Knowledge Base article VDP Conformance with Standard SQL contains a reference of the Virtual DataPort conformance with the SQL 92 standard. The document is focused on query capabilities and contains information about: Data Types, SQL Predicates Support and SQL Functions Support. The Query Expressions section lists the expressions defined by the standard SQL and their equivalent in Virtual DataPort, explaining the differences with the standard when appropriate.

Following, a group of tables where the correspondence between Oracle functions and the Denodo equivalents is presented. This list is just a reference since more functions can be included in future Denodo versions.

Functions mapping

Denodo function	Oracle function
cast(data_type, value)	cast(value as data_type)
coalesce(arg0 [, argi]*)	coalesce(arg0[, $argi]{1, n}, null)
coalesce(arg0 [, argi]*)	nvl(arg0 [, nvl(argi]{1, n-1}, $argn[)]{1, n-1})
concat(arg0, arg1 [,argi]*)	arg0 [|| argi]{1, n}
textcat(arg0, arg1 [,argi]*)	arg0 [|| argi]{1, n}
formatdate(date_pattern, datetimevalue [, localename])	to_char(date, date_pattern)
getday(daterelatedvalue)	EXTRACT(DAY FROM date)
gethour(timerelatedvalue)	to_number(to_char(date,''HH24''))
getminute(timerelatedvalue)	to_number(to_char(date,''MI''))
getsecond(timerelatedvalue)	to_number(to_char(date,''SS''))
gettimeinmillis(daterelatedvalue)	round((date - 0 - to_date('19700101000000','YYYYMMDDHH24MISS')) *

	86400000) + MOD(EXTRACT(SECOND FROM CAST(date AS TIMESTAMP))*1000,1000)
getmonth(daterelatedvalue)	EXTRACT(MONTH FROM date)
getyear(daterelatedvalue)	EXTRACT(YEAR FROM date)
getmillisecond(daterelatedvalue)	TO_NUMBER(SUBSTR(TO_CHAR(date, ''FF''), 0, 3))
getdaysbetween(daterelatedvalue1, daterelatedvalue2)	TRUNC(CAST(date2 AS DATE)) - TRUNC(CAST(date1 AS DATE))
getmonthsbetween(daterelatedvalue1, daterelatedvalue2)	TRUNC(MONTHS_BETWEEN(TRUNC(date2), TRUNC(date1)))
getdayofweek(daterelatedvalue)	TO_NUMBER(TO_CHAR(date, ''D''))
getdayofyear(daterelatedvalue)	TO_NUMBER(TO_CHAR(date, ''DDD''))
instr(arg0, arg1)	(INSTR(arg0,arg1) - 1)
degrees(arg0)	(arg0 * 180) / (asin(1) * 2)
radians(arg0)	arg0 * asin(1) * 2 / 180
log(arg0)	log(BASE_PARAM, arg0)
ltrim(arg0)	ltrim(arg0)
now	CURRENT_TIMESTAMP
max(arg0 [,argi]*)	GREATEST(arg0[, argi]{1, n})
min(arg0 [,argi]*)	LEAST(arg0[, argi]{1, n})
regexp(arg0, arg1, arg2)	regexp_replace(arg0, arg1, arg2,1,0,''MATCH_PARAM'')
?removeaccents(arg0)	replace(replace(replace(replace(replace(replace(replace(replace(replace(replace(replace(replace(replace(replace(replace(replace(replace(replace(replace(replace(replace(replace(replace(replace(replace(replace(replace(replace(replace(replace(replace(replace(replace(arg0, ''Ý'', ''Y''), ''Û'', ''U''), ''Ô'',

	''O''), ''Î'', ''I''), ''Ê'', ''E''), ''Â'', ''A''), ''Ü'', ''U''), ''Ö'', ''O''), ''Ï'', ''I''), ''Ë'', ''E''), ''Ä'', ''A''), ''Ù'', ''U''), ''Ò'', ''O''), ''Ì'', ''I''), ''È'', ''E''), ''À'', ''A''), ''Ú'', ''U''), ''Ó'', ''O''), ''Í'', ''I''), ''É'', ''E''), ''Á'', ''A''), ''ý'', ''y''), ''ÿ'', ''y''), ''û'', ''u''), ''ô'', ''o''), ''î'', ''i''), ''ê'', ''e''), ''â'', ''a''), ''ü'', ''u''), ''ö'', ''o''), ''ï'', ''i''), ''ë'', ''e''), ''ä'', ''a''), ''ù'', ''u''), ''ò'', ''o''), ''ì'', ''i''), ''è'', ''e''), ''à'', ''a''), ''ú'', ''u''), ''ó'', ''o''), ''í'', ''i''), ''é'', ''e''), ''á'', ''a'')
repeat(arg0, count)	rpad(arg0, length(arg0) * count, arg0)
round(arg0 [, argi])	round(arg0[, argi]{1,n})
rtrim(arg0)	rtrim(arg0)
substr(arg0, arg1)	SUBSTR(arg0,arg1)
substr(arg0, arg1,arg2)	SUBSTR(arg0,arg1,arg2)
to_date(date_pattern, value [, argi])	to_date(value, date_pattern[,arg2]{0,1})
formatdate(date_pattern, daterelatedvalue [, arg2])	to_char(date, date_pattern[,arg2]{0,1})
addday(daterelatedvalue, inc)	date+inc
addhour(timerelatedvalue,inc)	date+inc/24
addminute(timerelatedvalue,inc)	date+inc/(24*60)
addsecond(timerelatedvalue,inc)	date+inc/(24*60*60)
addmonth(daterelatedvalue,inc)	add_months(date,inc)
addweek(daterelatedvalue,inc)	date+(7*inc)
addyear(daterelatedvalue,inc)	add_months(date,(12*inc))
firstdayofmonth(daterela	add_months(last_day(date),-1)+1

tedvalue)	
firstdayofweek(daterelatedvalue)	next_day(date-7, 1)
lastdayofmonth(daterelatedvalue)	last_day(date)
lastdayofweek(daterelatedvalue)	next_day(date-1, 7)
nextweekday(daterelatedvalue,weekDay)	next_day(date,weekDay)
previousweekday(daterelatedvalue,weekDay)	next_day(date-8,weekDay)
trunc(daterelatedvalue[, pattern])	trunc(date[,pattern]{1,n})
current_date	trunc(current_date)
xmlquery(arg0, arg1)	XMLQUERY(arg0 PASSING BY VALUE arg1 RETURNING CONTENT)
xpath(arg0, arg1)	XMLQUERY(arg0 PASSING BY VALUE arg1 RETURNING CONTENT)
len(arg0)	LENGTH(arg0)
CASE arg0 WHEN arg1 THEN arg 2 [WHEN argi THEN argj] [ELSE argk]	decode(arg0, arg1, arg2 [, argi, argj] [, argk])
FROM view1 RIGHT OUTER JOIN view2 ON view1.column = view2.column	FROM view1, view2 WHERE view1.column(+) = view2.column
FROM view1 LEFT OUTER JOIN view2 ON view1.column = view2.column	FROM view1, view2 WHERE view1.column = view2.column(+)

In the above list, there are different types of data and time functions:

- Functions that are related with datetime types that includes dates (date, localdate, timestamp, timestamptz), will be referred as daterelatedvalue
- Functions that are related with datetime types that includes times (date, time, timestamp, timestamptz), will be referred as timerelatedvalue
- Functions that receive any datetime values, except intervals, will be referred as datetimevalue

Aggregation functions

Denodo Function	Oracle function
stdev(arg0)	STDDEV(arg0)
stdevp(arg0)	STDDEV_POP(arg0)
var(arg0)	VARIANCE(arg0)
varp(arg0)	VAR_POP(arg0)

Operators

Denodo function	Oracle operator
arg0 is FALSE	arg0 = 0
arg0 is TRUE	arg0 = 1
xmlexists(arg0, arg1)	XMLEXISTS(arg0 PASSING BY VALUE arg1)
regexp_like(arg0, arg1)	regexp_like(arg0,arg1,''MATCH_PARAM'')
regexp_ilike(arg0, arg1)	regexp_like(arg0,arg1,''iMATCH_PARAM'')