Tipos de datos, expresiones y operadores condicionales

Tipos de datos básica y conversión

A la hora de crear tablas, hay que indicar el tipo de datos de cada campo. Necesitamos pues conocer los distintos tipos de datos.

Los tipos de datos principales de ORACLE son los siguientes:

- · VARCHAR(n): dato de tipo carácter, de n caracteres de longitud.
- · NUMBER: dato de tipo numérico de un máximo de 40 dígitos, además del signo y el punto
- decimal. Se puede utilizar notación científica (1.273E2 es igual a 127.3).
- · NUMBER(n,d): dato de tipo numérico con n dígitos en total como máximo y d dígitos decimales
- como mucho. NUMBER(4,2) tiene como máximo valor 99.99.
- · DATE: datos de tipo fecha.

Cadenas de caracteres (`VARCHAR(n)`):

Se delimitan utilizando comillas simples: `'Hola'`, `'Una cadena'`. Además de los operadores de comparación e igualdad (<, >, =, !=, ...) otras funciones útiles para trabajar con cadenas son:

- · cad || cad : concatena dos cadenas.
- · LENGTH(cad): devuelve la longitud de la cadena.
- · LOWER(cad): convierte todas las letras de la cadena a minúsculas.
- · UPPER(cad): ídem a mayúsculas.
- · SUBSTR(cad, comienzo [, cuenta]): extrae la subcadena de cad empezando en la posición comienzo y con longitud (la subcadena) cuenta. El primer caracter de la cadena tiene como índice el número 1.

Números (NUMBER):

Además de las operaciones típicas con valores numéricos (+, -, *, /), otras funciones útiles son:

- · ABS(num): devuelve el valor absoluto.
- · SQRT(num): devuelve la raíz cuadrada.
- · POWER(b,e): devuelve la potencia be.
- · GREATEST(num1, num2, ...): devuelve el mayor valor de la lista de valores.
- · LEAST(num1, num2, ...): devuelve el menor valor de la lista.

Existen otras funciones para grupos de valores (suma, media, máximo, ...) que se verán más adelante.

Fechas (DATE):

El formato de un valor de tipo `DATE` es: `'dia-mes-año'`, donde tanto el día como el año tiene formato numérico y el mes se indica con las tres primeras letras del nombre del mes en el idioma soportado por el servidor ORACLE. Ejemplos: `'1-JAN-96'`, `'28-jul-74'`. Además de esta información, un valor de tipo fecha almacena también la hora en formato `hh:mm:ss`. Las fechas se pueden comparar con los operadores típicos de comparación (<, >, !=, =, ...).

- La función `SYSDATE` devuelve la fecha actual (fecha y hora).
- Con las fechas es posible realizar operaciones aritméticas como sumas y restas de fechas, teniendo en cuenta que a una fecha se le suman días y que la diferencia entre dos fechas se devuelve también en días. Por ejemplo `SYSDATE + 1` devuelve la fecha de mañana.

Oracle permite tanto la conversión de tipos implícita como la explícita. La conversión de tipos implícita significa que cuando Oracle encuentra en un lugar determinado (por ejemplo en una expresión) un dato de un tipo diferente al esperado, entonces aplica una serie de reglas para intentar convertir ese dato al tipo esperado. Por ejemplo, si un atributo de una tabla determinada es de tipo `NUMBER` y se intenta introducir el valor de tipo carácter `'1221'`, entonces automáticamente se convierte en su valor numérico equivalente sin producirse ningún error.

La **conversión de tipos explícita** se realiza básicamente con las siguientes funciones:

- · Conversión número-cadena: TO_CHAR(número [, formato]).
- · Conversión cadena-número: TO_NUMBER(cadena [,formato]).
- · Conversión fecha-cadena: TO_CHAR(fecha [, formato]).
- · Conversión cadena-fecha: TO_DATE(cadena [, formato]).

La opción **`formato`** permite especificar un modelo de formato o **máscara** consistente en una cadena de caracteres que describe el formato en el que se quiere obtener el resultado o en el que se da el parámetro. Algunos ejemplos de la utilización de estas funciones son:

- · TO_CHAR('25-dec-98', 'YY') devuelve '98'.
- · TO_CHAR(SYSDATE, 'dd-mon-yyyy') devuelve '25-dec-1998'.
- · TO_CHAR(123.34,'09999.999') devuelve '00123.340'.

Tipos de completa

De una forma más extensa a continuación se detallan todos los tipos de datos existentes, tanto en SQL Estándar como en Oracle.

Descripción	Tipos Estándar SQL	Oracle SQL

Texto		
Texto de anchura fija	**CHARACTER(*n*)** **CHAR(n)**	**CHAR(n)**
Texto de anchura variable	**CHARACTER VARYING(n)** **VARCHAR (n)**	**VARCHAR2(n)**
Texto de anchura fija para caracteres nacionales	**NATIONAL CHARACTER(n)** **NATIONAL CHAR(n)** **NCHAR(n)**	**NCHAR(n)**
Texto de anchura variable para caracteres nacionales	**NATIONAL CHARACTER VARYING(*n*)** **NATIONAL CHAR VARYING(n)** **NCHAR VARYING(n)**	**NVARCHAR2(n)**
Números		
Enteros pequeños (2 bytes)	**SMALLINT**	
Enteros normales (4 bytes)	**INTEGER** **INT**	
Enteros largos (8 bytes)	**BIGINT** (en realidad no es estándar, pero es muy utilizado en muchas bases de datos)	
Descripción	**Tipos Estándar SQL**	**Oracle SQL**
Enteros precisión decimal		**NUMBER(n)**

Decimal de coma variable	**FLOAT** **DOUBLE** **DOUBLE PRECISSION** **REAL**	**NUMBER**
Decimal de coma fija	**NUMERIC(*m,d*)** **DECIMAL(*m,d*)**	**NUMBER(*m,d*)**
Fechas		
Fechas	**DATE**	**DATE**
Fecha y hora	**TIMESTAMP**	**TIMESTAMP**
Intervalos	**INTERVAL**	**INTERVAL**
Booleanos y binarios		
Lógicos	**BOOLEAN** **BOOL**	
Binarios	**BIT** **BIT VARYING(n)** **VARBIT(n)**	
Datos de gran tamaño		
Texto gran longitud	**CHARACTER LARGE OBJECT** **CLOB**	**LONG** (en desuso) **CLOB**
Binario de gran longitud	**BINARY LARGE OBJECT** **BLOB**	**RAW (en desuso)** **LONG RAW** (en desuso) **BLOB**

De aquí en adelante solo se hará referencia a los tipos de datos utilizados en Oracle.

TEXTOS

VARCHAR . Para textos de longitud variable. Su tamaño depende de la base de datos (en Oracle es de 4000). En Oracle se llama **VARCHAR2**, pero es posible seguir utilizando VARCHAR.
CHAR. Para textos de longitud fija (en Oracle hasta 2000 caracteres).

NCHAR. Para el almacenamiento de caracteres nacionales de texto fijo

NVARCHAR. Para el almacenamiento de caracteres nacionales de longitud variable. En Oracle se llama **NVARCHAR2**.

En todos estos tipos se indican los tamaños entre paréntesis tras el nombre del tipo. Conviene poner suficiente espacio para almacenar los valores. En el caso de los VARCHAR2, no se malgasta espacio por poner más espacio del deseado ya que si el texto es más pequeño que el tamaño indicado, el resto del espacio se ocupa.

NÚMEROS

En Oracle, el tipo **NUMBER** es un formato versátil que permite representar todo tipo de números. Su rango recoge números de entre 10^{-130} y $9{,}99999999999 * 10^{128}$. Fuera de estos rangos Oracle devuelve un error.
Los números decimales (números de coma fija) se indican con **NUMBER(*p*,*s*)**, donde *p* es la precisión máxima y *s* es la escala (número de decimales a la derecha de la coma). Por ejemplo, NUMBER (8,3) indica que se representan números de ocho cifras de precisión y tres decimales. Los decimales en Oracle se presenta con el **punto y no con la coma.**
Para números enteros se indica **NUMBER(*p*)** donde *p* es el número de dígitos. Eso es equivalente a NUMBER(*p*,0).
Para números de coma flotante (equivalentes a los **float** o **double** de muchos lenguajes de programación) simplemente se indica el texto **NUMBER** sin precisión ni escala.

PRECISIÓN Y ESCALA

La cuestión de la precisión y la escala es compleja. Para entenderla mejor, se muestran estos ejemplos:

Formato	Número escrito por el usuario	Se almacena como…
NUMBER	345255.345	345255.345
NUMBER(9)	345255.345	345255
NUMBER(9,2)	345255.345	345255.35
NUMBER(9,-2)	345255.345	345300

NUMBER(7,2)	345255.345	Da error de precisión

En definitiva, la precisión debe incluir todos los dígitos del número (puede llegar hasta 38 dígitos). La escala sólo indica los decimales que se respetarán del número, pero si es negativa indica ceros a la izquierda del decimal.

FECHAS Y HORAS

DATE
El tipo **DATE** permite almacenar fechas. Las fechas se pueden escribir en formato día, mes y año entre comillas. El separador puede ser una barra de dividir, un guión y casi cualquier símbolo.
Para almacenar la fecha actual la mayoría de bases de datos proporcionan funciones (como **SYSDATE** en Oracle) que devuelven ese valor.

TIMESTAMP
Es una extensión del anterior, almacena valores de día, mes y año, junto con hora, minuto y segundos (incluso con decimales). Con lo que representa un instante concreto en el tiempo. Un ejemplo de TIMESTAMP sería '2/2/2004 18:34:23,34521'. En este caso si el formato de fecha y hora del sistema está pensado para el idioma español, el separador decimal será la coma (y no el punto).
intervalos
Sirven para almacenar intervalos de tiempo (no fechas, sino una suma de elementos de tiempo). En el caso de Oracle son:

DATOS DE GRAN TAMAÑO

Son tipos pensados para almacenar datos de tamaño muy grande. No pueden poseer índices ni ser parte de claves.
CLOB
Utilizado para almacenar datos de texto de gran tamaño (hasta 4 GB de texto)

BLOB
Utilizado para guardar datos binarios de hasta 4 GB de tamaño

Expresiones y operadores condicionales

Las condiciones son expresiones lógicas (devuelven verdadero o falso), algunos de los sitios donde pueden aparecer expresiones lógica son:

- La cláusula `WHERE` selecciona un subconjunto de tuplas, justo aquellas que cumplen la condición especificada.
- Cláusula `CHECK` que sirve para establecer condiciones sobre los valores almacenados en una tabla.

Las condiciones se construyen utilizando los operadores de comparación y los operadores lógicos. A continuación se describen los operadores más importantes junto con ejemplos de su utilización.

- **=, <>, <=, >=, < y >.**

Ejemplos:

- `horas >= 10.5`
- `nombre = 'PEPE'`
- `fecha < '1-ene-93'`

- **[NOT] IN lista_valores** : Comprueba la pertenencia a la lista de valores. Generalmente, los valores de la lista se habrán obtenido como resultado de un comando `SELECT` (comando de consulta).

Ejemplo:

- `nombre` **`NOT IN`** `('PEPE', 'LOLA')`

- **oper {ANY | SOME} lista_valores** : Comprueba que se cumple la operación `oper` con

algún elemento de la lista de valores. `oper` puede ser **<**, **>**, **<=**, **>=**, **<>**.

Ejemplo:

- `nombre = ANY ('PEPE', 'LOLA')`

- **oper ALL lista_valores** : Comprueba que se cumple la operación `oper` con todos los

elementos de la lista de valores. `oper` puede ser **<**, **>**, **<=**, **>=**, **<>**.

Ejemplo:

- `nombre <> ALL ('PEPE', 'LOLA')`

- **[NOT] BETWEEN x AND y**: Comprueba la pertenencia al rango `x - y`.

Ejemplo:

- `horas BETWEEN 10 AND 20` que equivale a `horas >= 10 AND horas <= 20`

○ **[NOT] EXISTS lista_valores** : Comprueba si la lista de valores contiene algún elemento.

Ejemplos:

- `EXISTS ('ALGO')` devuelve verdadero.
- `NOT EXISTS ('ALGO')` devuelve falso.

○ **[NOT] LIKE:** Permite comparar cadenas alfanuméricas haciendo uso de símbolos comodín.

Estos símbolos **comodín** son los siguientes:

_ : sustituye a un único carácter.

`%`: sustituye a varios caracteres.

Ejemplos:

- `nombre LIKE 'Pedro%'`
- `codigo NOT LIKE 'cod1_'`

Si dentro de una cadena se quieren utilizar los caracteres `'%'` o `'_'` tienen que ser *escapados* utilizando el símbolo `'\'`.

○ **IS [NOT] NULL** : Cuando el valor de un atributo, o es desconocido, o no es aplicable esa información, se hace uso del valor nulo (NULL). Para la comparación de valores nulos se utiliza el operador `IS [NOT] NULL`.

Ejemplo:

- `telefono IS NULL`

Los operadores lógicos junto con el uso de paréntesis permiten combinar condiciones simples obteniendo otras más complejas. Los operadores lógicos son:

- **OR**: `nombre = 'PEPE' OR horas BETWEEN 10 AND 20`
- **AND**: `horas > 10 AND telefono IS NULL`
- **NOT**: `NOT (nombre IN ('PEPE','LUIS'))`