# Curso de Base de Datos
# Sesión 5: SQL

2022-2

Ingeniería Informática

# Semana 5
# SQL DML

Profesor del curso:
César Aguilera
Luis Ríos

Elaborado por:
César Aguilera
Luis Ríos

Revisado por:
César Aguilera
Rony Cueva
Luis Rios

# Saberes previos

- SQL DDL, lenguaje de definición de datos
- SQL DML, lenguaje de manipulación de datos

# Agenda

- SQL DML
  - Uso de operadores
  - Manejo de fechas y expresiones aritméticas

# SQL – Operaciones con tipos de datos y funciones

- **NVL:** Si la expresión o función hace referencia a una columna con un valor nulo, el resultado es también nulo. Para convertir un valor nulo en uno NO NULO, al evaluar una expresión se emplea la función **NVL**.

**NVL** evalúa una primera expresión y si NO es nula la devuelve con su valor. En caso contrario devuelve la segunda expresión.

**Sintaxis:** NVL (*expresion1, expresion2*)

Ejemplo:

```
SELECT ename, sal, comm, NVL(comm,0)  expr
FROM emp
WHERE deptno=30 AND job='SALESMAN';
```

| ename | sal | comm | expr |
|---|---|---|---|
| Allen | 1600 | 300 | 300 |
| Jones | 2975 | 500 | 500 |
| Martin | 1250 | 1400 | 1400 |
| Turner | 1500 | 0 | 0 |

# SQL – Operaciones con tipos de datos y funciones

- **ROWNUM:** devuelve el número de la fila de una consulta. Aparece el número de cada fila en la posición de la tabla. Esta función actualiza sus valores usando subconsultas de modo que la consulta:

```
SELECT rownum as posicion, saldo, nomproveedor, razonsocial
FROM
                    (SELECT saldo, nomproveedor, razonsocial
                     FROM Proveedor ORDER BY saldo DESC )
WHERE ROWNUM <= 5;
```

Este ejemplo permite obtener a los 5 proveedores con mayor saldo actualmente.

# SQL – Operaciones con tipos de datos y funciones

## Funciones numéricas

| FUNCIÓN | SINTAXIS | EJEMPLO |
|---|---|---|
| Módulo o resto | MOD(dividendo, divisor) | MOD(7,5) es 2 |
| Raíz cuadrada | SQRT (valor numérico) | SQRT(25) es 5 |
| Redondeo | ROUND(valor, numérico precisión) | ROUND(monto, 2) monto redondeado a 2 decimales |
| Truncamiento | TRUNC(valor numérico, precisión) | TRUNC(monto, 2) monto truncado a 2 decimales |
| Potencia | POWER(valor numérico, potencia) | POWER(monto, 3) monto elevado al cubo |

# SQL – Operaciones con tipos de datos y funciones



## Expresiones aritméticas con fechas

Desde que las bases de datos almacenan fechas como números, se pueden realizar operaciones usando operadores aritméticos como sumas y restas.

Agregando o subtrayendo a una fecha un número de días, se obtiene una fecha: *ColumnaFecha* ± *NúmeroDías*

Restando dos fechas entre si, se obtiene el número de días que existe entre ambas: *ColumnaFecha1 – ColumnaFecha2*

En resumen, se pueden ejecutar las siguientes operaciones:

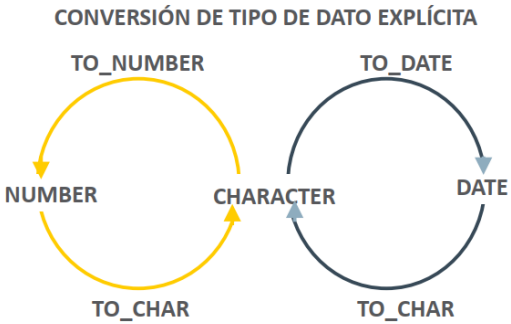
| Operación | Resultado | Descripción |
|---|---|---|
| Fecha + numero | Fecha | Añade un número de días a una fecha |
| Fecha – número | Fecha | Resta un número de días de una fecha |
| Fecha – Fecha | Número de días | Resta una fecha a otra |
| Fecha + número / 24 | Fecha | Añade un número de horas a una fecha |

# SQL – Operaciones con tipos de datos y funciones

## Expresiones aritméticas con fechas

| FUNCIÓN | SINTAXIS | EJEMPLO |
|---|---|---|
| Agregando meses | ADD_MONTHS (Columnafecha, ±Núméromeses) | ADD_MONTHS(fechaingreso,-6) devuelve 6 meses antes de la fecha de ingreso |
| Último día del mes de la fecha | LAST_DAY(ColumnaFecha) | LAST_DAY(fechaingreso) devuelve el último día del mes en que ingresó |
| Siguiente día específico de la semana después de la fecha | NEXT_DAY(ColumnaFecha, DiaSemana) | NEXT_DAY(fechaingreso, ‘ FRIDAY’) devuelve el primer viernes después de que ingresó |
| Número de meses entre dos fecha | MONTHS_BETWEEN(ColFecha1, ColFecha2) | MONTHS_BETWEEN(sysdate, fechaingreso) devuelve los meses trabajados |
| Fecha actual | SYSDATE | SYSDATE() devuelve la fecha actual |

## Expresiones aritméticas con fechas




- Es posible dar formato a los valores de fecha empleando una función de conversión de caracteres y elementos de formato.

  ***Sintaxis:*** **TO_CHAR***(ColumnaFecha,'ElementosFormato')*

- Es posible dar formato a los valores de campos numéricos empleando una función de conversión de caracteres y elementos de formato.

  ***Sintaxis:*** **TO_CHAR***(ColumnaNumérica,'ElementosFormato')*

- *El valor retornado siempre será un VARCHAR2.*

## Expresiones aritméticas con fechas

Ejemplos:

```
SELECT id_empleado, TO_CHAR(fecha_ingreso, 'MM/YYYY') Mes_Ingreso
FROM Empleados
WHERE  ap_paterno= 'Salazar';
```

| Id_empleado | Mes_Ingreso |
|---|---|
| 315 | 07/2004 |

```
SELECT s.nombre, s.apellidopaterno, s.email, s.telefono
FROM  Empleados s
WHERE
to_number(to_char(s.fecha_ingreso,'MM') ) = 4 and
(to_number(to_char(s.fecha_ingreso,'YYYY') ) between 1960 and 1970 );
```

# SQL – Operaciones con tipos de datos y funciones

## Expresiones aritméticas con fechas

| ELEMENTOS QUE DAN FORMATO A FECHAS | SINTAXIS | EJEMPLO |
|---|---|---|
| Día del mes (01-31) | DD | 1 – 31 |
| Día de la semana (tres primeras letras) | DY | THU |
| Nombre en mayúsculas del día de la semana (9 caracteres) | DAY | THUESDAY |
| Nombre en mayúsculas del número del día del mes | DDSPTH | TWELFTH |
| Mes (01 -12) | MM | 01 – 12 |
| Nombre en mayúsculas del mes (9 caracteres) | MONTH | JANUARY |
| Año de dos dígitos | YY | 91 |
| Año de cuatro dígitos | YYYY | 2011 |
| Horas, minutos y segundos | HH:MI:SS | 09:00:00 |
| Supresión de blancos en elementos subsiguientes | FM | |

# SQL – Operaciones con tipos de datos y funciones

- Ejemplos:

*SELECT TO_CHAR(sysdate) FROM DUAL;  SQL>04/02/21*

*SELECT TO_CHAR(sysdate, 'dd/mm/yyyy') FROM DUAL; SQL>24/04/2021*

*SELECT TO_CHAR(sysdate,'Mon') FROM DUAL; SQL>Feb*

*SELECT TO_CHAR(sysdate, 'Month') FROM DUAL; SQL>Febrero*

*SELECT TO_CHAR(sysdate, 'DDD') FROM DUAL; SQL>035*

*SELECT TO_CHAR(sysdate, 'DD') FROM DUAL; SQL>04*

- Ejemplos:

*SELECT TO_CHAR(123.456, '09999') FROM DUAL; SQL>00123*

*SELECT TO_CHAR(123.456, '09999.9') FROM DUAL; SQL>00123.5*

*SELECT TO_CHAR(123456, 'FM999,999,999') FROM DUAL; SQL>123,456*

# SQL – Operaciones con tipos de datos y funciones

## Expresiones aritméticas con cadenas

| FUNCIONES CON CARACTERES | SINTAXIS | EJEMPLO |
|---|---|---|
| Devolver la primera letra en mayúscula y el resto en minúscula | INITCAP(ColumnaCadena) | INITCAP(nombre) devuelve la primera letra de los nombres en mayúscula y el resto en minúscula |
| Devolver todas en mayúsculas | UPPER(ColumnaCadena) | UPPER(nombre) devuelve todos los caracteres del nombre en mayúscula |
| Devolver todas en minúsculas | LOWER(ColumnaCadena) | LOWER(nombre) devuelve todos los caracteres del nombre en minúscula |
| Devolver desde la posición dada N caracteres | SUBSTR(ColumnaCadena, Posición,N) | SUBSTR(empleo,1,3) devuelve desde la primera letra 3 caracteres del empleo |
| Devolver el número de caracteres de una cadena | LENGTH(ColumnaCadena) | LENGTH(nombre) devuelve el número de caracteres del nombre |

# SQL – Operaciones con tipos de datos y funciones

## Expresiones aritméticas con cadenas

| FUNCIONES CON CARACTERES | SINTAXIS | EJEMPLO |
|---|---|---|
| El mayor entre dos valores | GREATEST(Columna1, Columna2) | GREATEST(sal,comi) devuelve el mayor entre salario y comisión |
| El menor entre dos valores | LEAST(Columna1,Columna 2) | LEAST('Adam','Smith') devuelve el menor: Adam |

# Consultas sumarizadas

## Funciones de columna

- SUM () calcula el total de una columna.
- AVG () calcula el valor promedio de una columna.
- MIN() encuentra el valor más pequeño en una columna.
- MAX() encuentra el valor mayor en una columna.
- COUNT() cuenta el número de valores en una columna.
- COUNT(*) cuenta las filas de resultados de la consulta.

# Consultas sumarizadas

## Funciones de columna

El argumento de una función columna puede ser un solo nombre de columna.

¿Cuál es el rendimiento de cuota promedio de los vendedores?

```
SELECT AVG(100*(VENTAS/CUOTA)) FROM REPVENTAS

AVG (100*(VENTAS/CUOTA))
------------------------------------
                              102.60
```

## Funciones de columna

Cálculo del total de una columna (SUM)
La función columna SUM() calcula la suma de una columna de valores de datos. Los datos de la columna deben tener un tipo numérico (entero, decimal, coma flotante o monetario).

¿Cuáles son las cuotas y ventas totales para todos los vendedores?

```
SELECT SUM(CUOTA), SUM(VENTAS) FROM REPVENTAS

SUM(CUOTA)          SUM(VENTAS)
--------------------  --------------------
$2,700,000.00         $2,800,352.00
```

## Funciones de columna

Cálculo del promedio de una columna (AVG)
La función columna AVG() calcula el promedio de una columna de valores de datos. Al igual que una función SUM(), los datos de la columna deben tener un tipo numérico. Ya que la función AVG() suma los valores de la columna y luego los divide por el número de valores.

Calcula el precio medio de los productos del fabricante ACI.

```
SELECT AVG(PRECIO) FROM PRODUCTOS
WHERE ID_FAB='ACI'
```

```
AVG(PRECIO)
-----------
    $804.29
```

## Funciones de columna

### Determinación de valores extremos (MIN y MAX)

Las funciones de columna MIN() y MAX() determinan los valores mayor y menor de una columna, respectivamente. Los datos de la columna pueden contener información numérica, de cadena o de fecha/hora.

¿Cuáles son las cuotas asignadas mínimas y máxima.?

SELECT **MIN**(CUOTA), **MAX**(CUOTA) FROM REPVENTAS

| MIN(CUOTA) | MAX(CUOTA) |
|---|---|
| $200,000.00 | $350,000.00 |

# Consultas sumarizadas

## Funciones de columna

### Determinación de valores extremos (MIN y MAX)

¿Cuál es la fecha de pedido más antigua en la base de datos.?

```
SELECT MIN(FECHA_PEDIDO) FROM PEDIDOS

        MIN(FECHA_PEDIDO)
        ---------------------------------
                14-ABRIL-2014
```

¿Cuál es el mejor rendimiento de ventas de todas los vendedores?

```
SELECT MAX(100*(VENTAS/CUOTA)) FROM REPVENTAS

        MAX(100*(VENTAS/CUOTA))
        ---------------------------------------
                                135.44
```

Ingeniería Informática

## Funciones de columna

### Cuenta de valores de datos (COUNT)

La función de columna COUNT() cuenta el número de valores de datos que hay en una columna. Los datos de la columna pueden ser de cualquier tipo. La función COUNT() siempre devuelve un entero, independientemente del tipo de datos de la columna.

¿Cuántos clientes hay?

```
SELECT COUNT(NUM_CLIE) FROM CLIENTES

        COUNT(NUM_CLIE)
        ---------------
                    21
```

## Funciones de columna

### Cuenta de valores de datos (COUNT)

¿Cuántos vendedores superan su cuota?

```
SELECT COUNT(NOMBRE)
FROM REPVENTAS
WHERE VENTAS > CUOTA
```

```
COUNT(NOMBRE)
-------------
           7
```

¿Cuántos pedidos de más de $25,000 hay en los registros?

```
SELECT COUNT(IMPORTE)
FROM PEDIDOS
WHERE IMPORTE > 25000.00
```

```
COUNT(IMPORTE)
--------------
            4
```

# Consultas sumarizadas

## Funciones de columna

### Cuenta de valores de datos (COUNT)

SQL soporta una función de columna especial COUNT(*) que cuenta filas en lugar de valores de datos. He aquí la misma consulta, reescrita utilizando la función COUNT(*)

```
SELECT COUNT(*)
FROM PEDIDOS
WHERE IMPORTE > 25000.00
        COUNT(*)
        ---------------
                4
```

## Funciones de columna

### Valores NULL y funciones de columna

Las funciones de columna SUM(), AVG(), MIN(), MAX() y COUNT() aceptan cada una de ellas una columna de valores de datos como argumento y producen un único valor como resultado.

```
SELECT COUNT(*), COUNT(VENTAS), COUNT(CUOTA)
FROM REPVENTAS
```

| COUNT(*) | COUNT(VENTAS) | COUNT(CUOTA) |
|---|---|---|
| 10 | 10 | 9 |

## Funciones de columna

### Valores NULL y funciones de columna

La tabla REPVENTAS contiene diez filas, por lo que COUNT(*) devuelve una cuenta de diez. La columna VENTAS contiene diez valores no NULL, por lo que la función COUNT(VENTAS) también devuelve una cuenta de diez. La columna CUOTA es NULL para el vendedor más reciente. La función COUNT(CUOTA) ignora este valor NULL y devuelve un menor valor.

Debido a estas anomalías, la función COUNT(*) es utilizada casi siempre en lugar de la función COUNT(), a menos que específicamente se desee excluir del total los valores NULL de una columna en particular.

## Funciones de columna

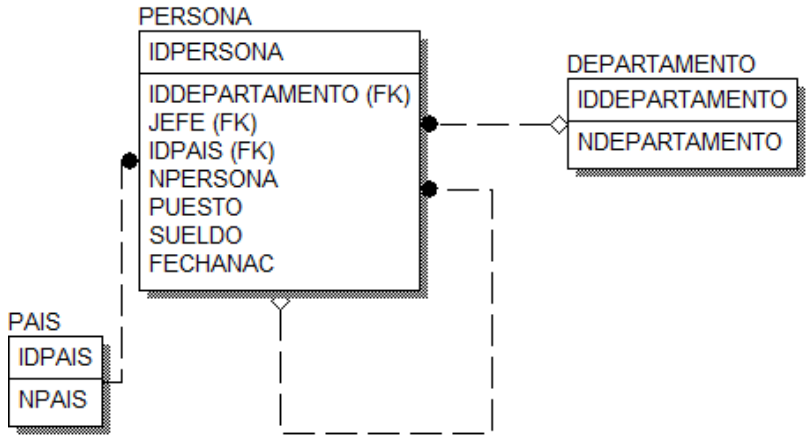
### Valores NULL y funciones de columna

Todas las funciones de grupo ignoran los valores nulos en las columnas.

```
SELECT  NVL (comision, 0)
FROM empleados;
```

La función NVL forza al grupo de funciones a incluir los valores NO nulos.

# Caso propuesto




## Insertar datos para toda la tabla

- SQL> INSERT INTO PAIS VALUES (20,'PERU');
- SQL> INSERT INTO DEPARTAMENTO (IDDEPARTAMENTO,NDEPARTAMENTO) VALUES ('01','LIMA');

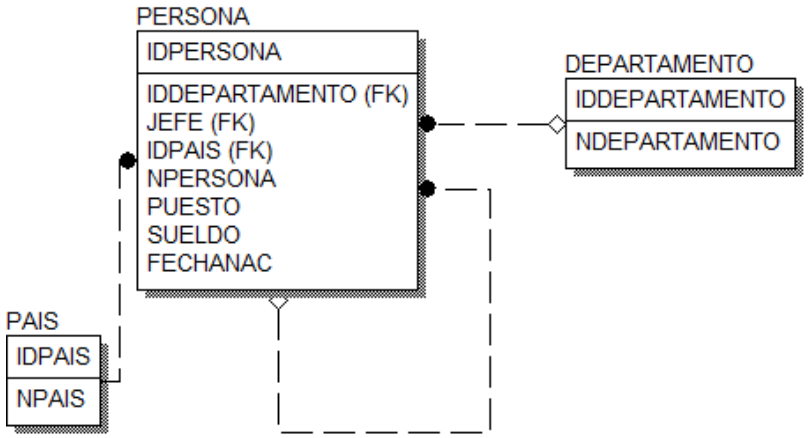
## Insertar datos en columnas concretas

- SQL>INSERT INTO PERSONA VALUES ('0001','01',NULL,'PE','JUAN PEREZ','EMPLEADO',1500,NULL);

## Confirmar los datos

- SQL> COMMIT;

# Caso propuesto



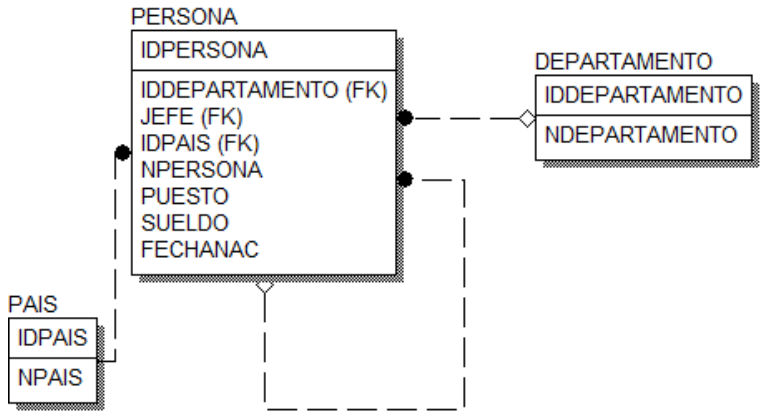

## Borrar datos

- SQL> DELETE FROM PERSONA WHERE IDPERSONA='0002';

## Revertir cambios

- SQL> ROLLBACK;

# Caso propuesto




**Borrar datos – ON DELETE RESTRICT**

Esta cláusula evita que se borren datos en la tabla padre, siempre y cuando existan en la tabla hija datos en las columnas referenciadas, coincidan con dicho valor.

**Ejemplo:**

- CONSTRAINT FK_PERSONA_PAIS FOREIGN KEY(IDPAIS) REFERENCE PAIS (IDPAIS)

Si queremos borrar la línea de la tabla PAIS que contiene en la columna IDPAIS el valor de 15, utilizaríamos la siguiente instrucción:

- DELETE PAIS WHERE IDPAIS = 15

El sistema nos devolvería un error indicándonos que no es posible llevar acabo la operación, porque existe información de una tabla hija que referencia el valor de esta columna.

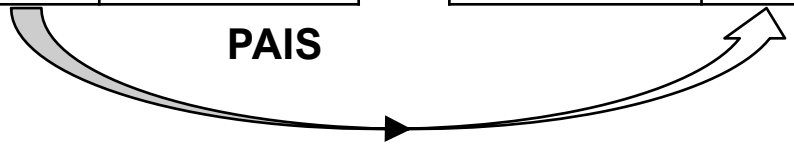
# Caso propuesto

| IDPAIS | NPAIS |
|---|---|
| 14 | ARGENTINA |
| 15 | BRASIL |
| 16 | ALBANIA |
| 17 | CROACIA |

**PAIS**

| IDPERSONA | IDPAIS | NPERSONA |
|---|---|---|
| 20120013 | 14 | CECILIA CASTRO |
| 20120018 | 16 | JUAN PEREZ |
| 20121015 | 17 | ADAM SMITH |
| 20121178 | 15 | MARCIA SANTOS |

**PERSONA**



- CONSTRAINT FK_PERSONA_PAIS FOREIGN KEY(IDPAIS) REFERENCE PAIS (IDPAIS) **ON DELETE CASCADE**

Si quisiéramos borrar la línea de la tabla PAIS que contiene en la columna IDPAIS el valor de 15, utilizando la siguiente instrucción:

- DELETE FROM PAIS WHERE IDPAIS=15;

# Caso propuesto

El resultado que obtenemos en cuanto a las tablas de nuestro ejemplo sería el siguiente:

| IDPAIS | NPAIS |
|---|---|
| 14 | ARGENTINA |
| 16 | ALBANIA |
| 17 | CROACIA |

**PAIS**

| IDPERSONA | IDPAIS | NPERSONA |
|---|---|---|
| 20120013 | 14 | CECILIA CASTRO |
| 20120018 | 16 | JUAN PEREZ |
| 20121015 | 17 | ADAM SMITH |

**PERSONA**

Para mantener la integridad referencial en los datos, también se borran las filas de la tabla hija que contenga la clave de la fila borrada en el padre.

# Caso propuesto

Actualizar datos

- SQL> UPDATE PERSONA SET PUESTO = ‘JEFE’ WHERE IDPERSONA=‘0001’;

- SQL> UPDATE PERSONA SET PUESTO = ‘JEFE’, SUELDO = SUELDO*1.2
  WHERE IDPERSONA=‘0001’;

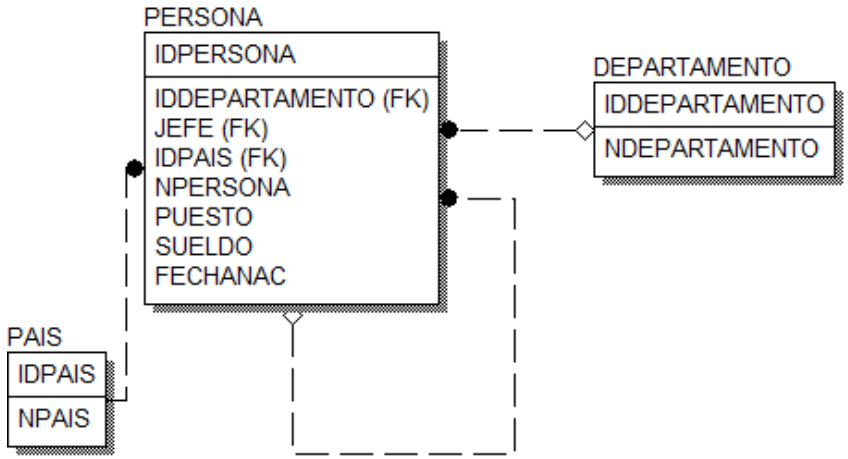
UPDATE no genera ningún resultado. Para saber qué registros se van a cambiar, hay que examinar primero el resultado de una consulta de selección que utilice el mismo criterio y después ejecutar la consulta de actualización.

Si en una consulta de actualización suprimimos la cláusula WHERE todos los registros de la tabla señalada serán actualizados.

UPDATE PERSONA SET Sueldo = Sueldo * 1.2;

- COMMIT;
- ROLLBACK;

# Caso propuesto




Insertar fechas

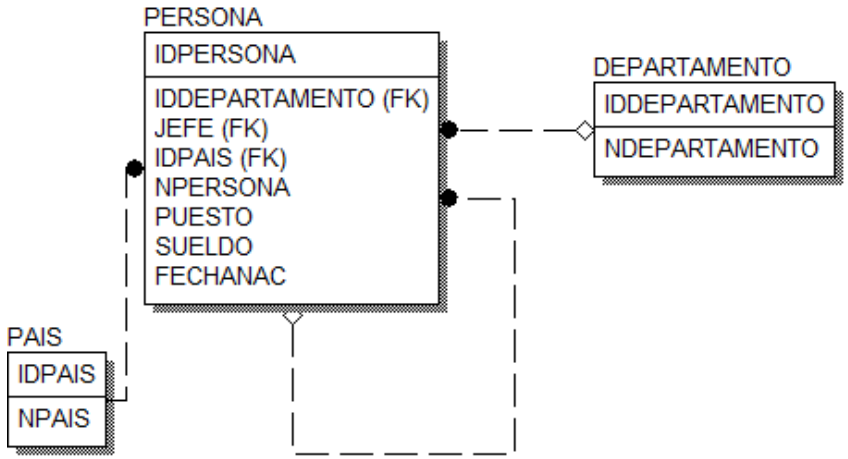
- SQL> INSERT INTO PERSONA VALUES ('0003','01',NULL,'PE','JORGE VIZCAINO','EMPLEADO',1500,TO_DATE('01-12-1975','DD-MM-YYYY'));

Además

Hay un tipo de consulta, que permite rellenar datos de una tabla copiando el resultado de una consulta.

- SQL> INSERT INTO PERSONATEMPORAL (idpersona, npersona, sueldo, fechanac)
  SELECT idpersona, npersona, sueldo, fechanac FROM PERSONA
  WHERE TO_CHAR(FECHANAC,'MM-YYYY')='12-1975';

# Caso propuesto




## Insertar fechas

- SQL> INSERT INTO PERSONA VALUES ('0003','01',NULL,'PE','JORGE VIZCAINO','EMPLEADO',1500,TO_DATE('01-12-1975','DD-MM-YYYY'));

## Convertir fecha a cadena

- SQL> SELECT * FROM PERSONA
  WHERE TO_CHAR(FECHANAC,'MM-YYYY')='12-1975';

# Resumen

En esta sesión, debe haber aprendido lo siguiente:

- SQL DDL son comandos que se utilizan para crear la estructura física de la base de datos
- SQL DQL es un comando que se utiliza para consultar los datos de la base de datos
- SQL DML son comandos que se utilizan para modificar los datos de la base de datos