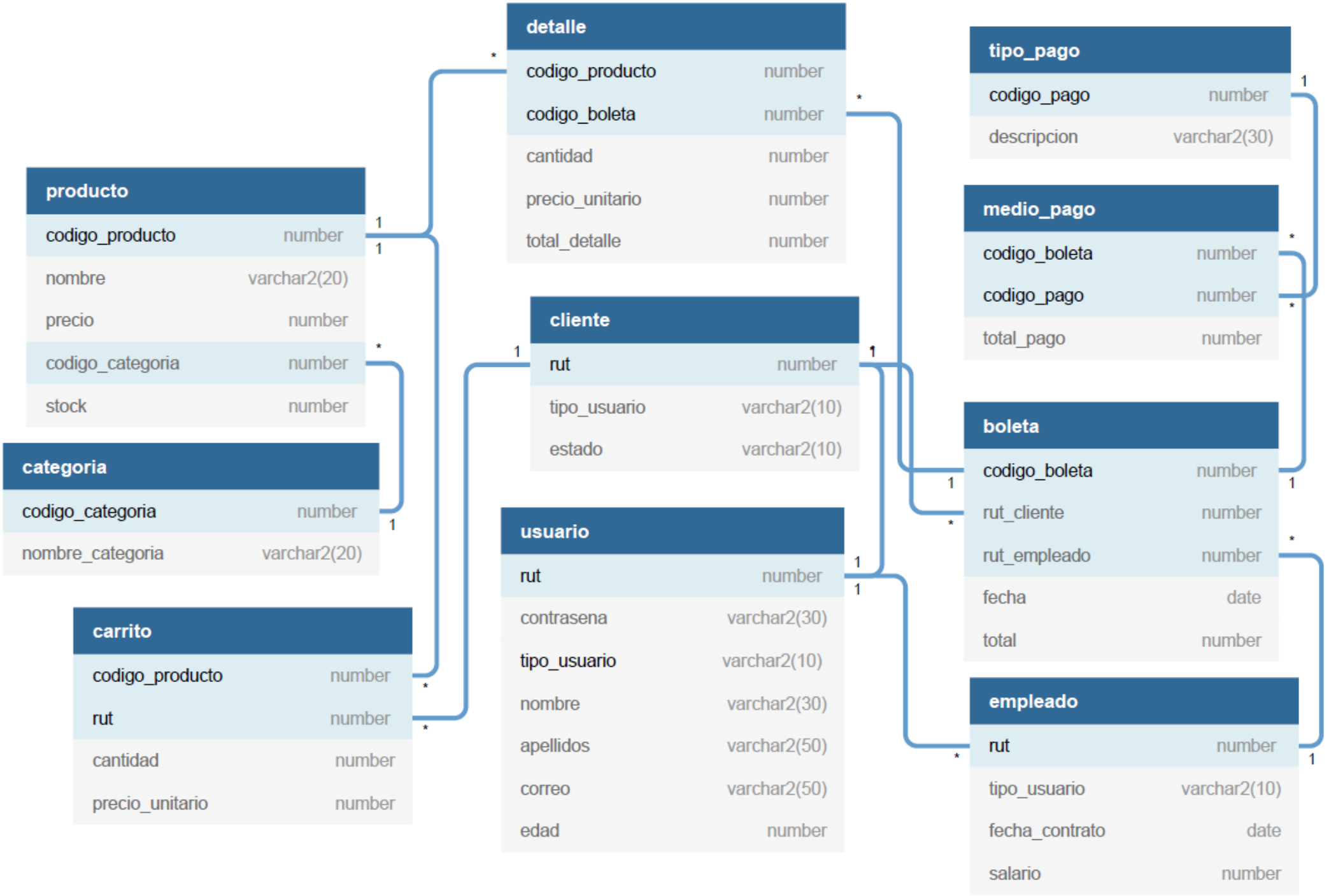
UNIVERSIDAD CATOLICA DEL MAULE

Funciones y Cursores

Programa de hoy

- Modelo relacional final
- Funciones
- Cursores
- Secuencias
- Ejercicios


detalle
codigo_producto number
codigo_boleta number
cantidad number
precio_unitario number
total_detalle number
tipo_pago
codigo_pago number
descripcion varchar2(30)
producto
codigo_producto number
nombre varchar2(20)
precio number
codigo_categoria number
stock number
medio_pago
codigo_boleta number
codigo_pago number
total_pago number
cliente
rut number
tipo_usuario varchar2(10)
estado varchar2(10)
boleta
codigo_boleta number
rut_cliente number
rut_empleado number
fecha date
total number
categoria
codigo_categoria number
nombre_categoria varchar2(20)
usuario
rut number
contrasena varchar2(30)
tipo_usuario varchar2(10)
nombre varchar2(30)
apellidos varchar2(50)
correo varchar2(50)
edad number
carrito
codigo_producto number
rut number
cantidad number
precio_unitario number
empleado
rut number
tipo_usuario varchar2(10)
fecha_contrato date
salario number

Funciones

Anteriormente hemos revisado el trabajo con bloques anónimos y procedimientos almacenados.

Funciones

Anteriormente hemos revisado el trabajo con bloques anónimos y procedimientos almacenados.

```
DECLARE
    [VARIABLES]
BEGIN
    [OPERACIONES][CONSULTAS][LLAMADA PROCEDIMIENTOS][FUNCIONES][ETC]
EXCEPTION
    MANEJO DE EXCEPCIONES
END;
```

```
CREATE OR REPLACE FUNCTION NOMBRE_FUNCION(
    DATOS DE ENTRADA DE LA FUNCION
)
RETURN TIPO_DE_DATO
IS
    [VARIABLES]
BEGIN
    [OPERACIONES][CONSULTAS][LLAMADA PROCEDIMIENTOS][LLAMADA FUNCIONES][ETC]
EXCEPTION
    MANEJO DE EXCEPCIONES
END;
```

Funciones

Anteriormente hemos revisado el trabajo con bloques anónimos y procedimientos almacenados.

El último de los bloques pendiente por revisar corresponde a las funciones.

Funciones

Anteriormente hemos revisado el trabajo con bloques anónimos y procedimientos almacenados.

El último de los bloques pendiente por revisar corresponde a las funciones.

Las funciones poseen una definición y uso similar a los procedimientos, presentando una variación sobre dos puntos esenciales.

Funciones

Anteriormente hemos revisado el trabajo con bloques anónimos y procedimientos almacenados.

El último de los bloques pendiente por revisar corresponde a las funciones.

Las funciones poseen una definición y uso similar a los procedimientos, presentando una variación sobre dos puntos esenciales.

- Retornan un único valor.
- La función retorna el valor sobre el espacio de trabajo donde se llamó.

Funciones (Sintaxis)

```
CREATE OR REPLACE FUNCTION NOMBRE_FUNCION(
    DATOS DE ENTRADA DE LA FUNCION
)
RETURN TIPO_DE_DATO
IS
    [VARIABLES]
BEGIN
    [OPERACIONES][CONSULTAS][LLAMADA PROCEDIMIENTOS][LLAMADA FUNCIONES][ETC]
    RETURN VALOR
EXCEPTION
    MANEJO DE EXCEPCIONES
END;
```

Funciones (Sintaxis)

```
CREATE OR REPLACE FUNCTION NOMBRE_FUNCION(
    DATOS DE ENTRADA DE LA FUNCION
)
RETURN TIPO_DE_DATO
IS
    [VARIABLES]
BEGIN
    [OPERACIONES][CONSULTAS][LLAMADA PROCEDIMIENTOS][LLAMADA FUNCIONES][ETC]
    RETURN VALOR
EXCEPTION
    MANEJO DE EXCEPCIONES
END;
```

Como recomendación de uso, la función debería ser utilizada para dividir el problema en sub problemas, realizar cálculos, transformaciones, comprobaciones u otras operaciones que van en apoyo a la utilización de procedimientos.

Funciones (Sintaxis)

```
CREATE OR REPLACE FUNCTION NOMBRE_FUNCION(
    DATOS DE ENTRADA DE LA FUNCION
)
RETURN TIPO_DE_DATO
IS
    [VARIABLES]
BEGIN
    [OPERACIONES][CONSULTAS][LLAMADA PROCEDIMIENTOS][LLAMADA FUNCIONES][ETC]
    RETURN VALOR
EXCEPTION
    MANEJO DE EXCEPCIONES
END;
```

Como recomendación de uso, la función debería ser utilizada para dividir el problema en sub problemas, realizar cálculos, transformaciones, comprobaciones u otras operaciones que van en apoyo a la utilización de procedimientos.

Ejemplos comunes pueden ser funciones para verificar la existencia de un usuario, validar un rut, verificar stock disponible, entre otros.

Funciones (Ejemplo)

```
CREATE OR REPLACE FUNCTION SUMA(
    NUMERO1 IN NUMBER,
    NUMERO2 IN NUMBER
)RETURN NUMBER
IS
    RESULTADO NUMBER;
BEGIN
    RESULTADO := NUMERO1 + NUMERO2;
    RETURN RESULTADO;
END;
```

Funciones (Ejemplo)

```
CREATE OR REPLACE FUNCTION SUMA(
    NUMERO1 IN NUMBER,
    NUMERO2 IN NUMBER
)RETURN NUMBER
IS
    RESULTADO NUMBER;
BEGIN
    RESULTADO := NUMERO1 + NUMERO2;
    RETURN RESULTADO;
END;
```

```
DECLARE
    RESULTADO NUMBER;
BEGIN
    RESULTADO := SUMA(2,2);
    DBMS_OUTPUT.PUT_LINE(RESULTADO);
END;
```

Funciones (Ejemplo)

```
CREATE OR REPLACE FUNCTION SUMA(
    NUMERO1 IN NUMBER,
    NUMERO2 IN NUMBER
)RETURN NUMBER
IS
    RESULTADO NUMBER;
BEGIN
    RESULTADO := NUMERO1 + NUMERO2;
    RETURN RESULTADO;
END;
```

```
DECLARE
    RESULTADO NUMBER;
BEGIN
    RESULTADO := SUMA(2,2);
    DBMS_OUTPUT.PUT_LINE(RESULTADO);
END;
```

Ejercicio SUMA utilizando función

Funciones (Ejemplo)

```
CREATE OR REPLACE FUNCTION SUMA(
    NUMERO1 IN NUMBER,
    NUMERO2 IN NUMBER
)RETURN NUMBER
IS
    RESULTADO NUMBER;
BEGIN
    RESULTADO := NUMERO1 + NUMERO2;
    RETURN RESULTADO;
END;
```

```
CREATE OR REPLACE PROCEDURE SUMA(
    NUMERO1 IN NUMBER,
    NUMERO2 IN NUMBER,
    RESULTADO OUT NUMBER
)
IS
BEGIN
    RESULTADO := NUMERO1 + NUMERO2;
END;
```

```
DECLARE
    RESULTADO NUMBER;
BEGIN
    RESULTADO := SUMA(2,2);
    DBMS_OUTPUT.PUT_LINE(RESULTADO);
END;
```

Ejercicio SUMA utilizando función

```
DECLARE
    RESULTADO NUMBER;
BEGIN
    SUMA(1,2,RESULTADO);
    DBMS_OUTPUT.PUT_LINE(RESULTADO);
END;
```

Ejercicio SUMA utilizando procedimiento

Cursores

Como bien sabemos, la utilización de una instrucción SELECT devuelve las filas pertenecientes a una tabla.

Cursores

Como bien sabemos, la utilización de una instrucción SELECT devuelve las filas pertenecientes a una tabla.

PRODUCTO				
CODIGO_PRODUCTO	NOMBRE	PRECIO	CODIGO_CATEGORIA	STOCK
1	NOTEBOOK LENOVO	$380.000	1	20
2	CELULAR MOTOROLA	$110.000	2	15
3	AUDIFONOS MACROTEL	$3.500	3	12
4	NOTEBOOK SAMSUNG	$500.000	1	3
5	MONITOR 17" AOC	$67.990	4	80
6	MONITOR 21" DELL	$81.990	4	20
7	NOTEBOOK HP	$280.990	1	1
8	CELULAR IPHONE 6S	$560.000	2	6
9	CELULAR LG	$450.000	2	7
10	MOUSE BLUETOOTH	$28.990	5	9
11	MONITOR 17" SAMSUNG	$150.000	4	7
12	MONITOR 17" LENOVO	$250.000	4	7
13	MONITOR 17" LG	$125.000	4	4
14	MONITOR 15" LENOVO	$200.000	4	3
15	MOTO G 2013	$130.000	2	45
16	MOTO G 2015	$180.000	2	32
17	MOTO X PLAY	$370.000	2	12
18	IPAD MINI 2	$200.000	6	4
19	IPAD AIR	$260.000	6	2
20	NOTEBOOK MSI	$850.000	1	4
21	MOUSE MICROSOFT	$8.500	5	5

Cursores

Como bien sabemos, la utilización de una instrucción SELECT devuelve las filas pertenecientes a una tabla.

Un cursor, es una estructura que permite recorrer fila a fila un resultado de una instrucción SELECT.

Cursores

Como bien sabemos, la utilización de una instrucción SELECT devuelve las filas pertenecientes a una tabla.

Un cursor, es una estructura que permite recorrer fila a fila un resultado de una instrucción SELECT.

Su utilización se realiza sobre bloques anónimos, procedimientos o funciones.

Cursores

Como bien sabemos, la utilización de una instrucción SELECT devuelve las filas pertenecientes a una tabla.

Un cursor, es una estructura que permite recorrer fila a fila un resultado de una instrucción SELECT.

Su utilización se realiza sobre bloques anónimos, procedimientos o funciones.

Para entender mejor esta idea se presenta a continuación la sintaxis de un cursor y la explicación.

Cursores (Sintaxis)

La sintaxis de un cursor, es bastante sencilla.

```
CURSOR NOMBRE_CURSOR
IS SENTENCIA_SELECT;
```

Para entender la idea del cursor utilizamos la instrucción SELECT * FROM PRODUCTO.

Cursores (Sintaxis)

La sintaxis de un cursor, es bastante sencilla.

```
CURSOR NOMBRE_CURSOR
IS SENTENCIA_SELECT;
```

Para entender la idea del cursor utilizamos la instrucción SELECT * FROM PRODUCTO.

```
CURSOR PRODUCTOS
IS SELECT * FROM PRODUCTO;
```

Cursores (Sintaxis)

La sintaxis de un cursor, es bastante sencilla.

```
CURSOR NOMBRE_CURSOR
IS SENTENCIA_SELECT;
```

Para entender la idea del cursor utilizamos la instrucción SELECT * FROM PRODUCTO.

```
CURSOR PRODUCTOS
IS SELECT * FROM PRODUCTO;
```

Al declarar esto, estaremos almacenando en el cursor “PRODUCTOS” todas las filas que retorne la instrucción SELECT.

Cursores (Sintaxis)

PRODUCTO				
CODIGO_PRODUCTO	NOMBRE	PRECIO	CODIGO_CATEGORIA	STOCK
1	NOTEBOOK LENOVO	$380.000	1	20
2	CELULAR MOTOROLA	$110.000	2	15
3	AUDIFONOS MACROTEL	$3.500	3	12
4	NOTEBOOK SAMSUNG	$500.000	1	3
5	MONITOR 17" AOC	$67.990	4	80
6	MONITOR 21" DELL	$81.990	4	20
7	NOTEBOOK HP	$280.990	1	1
8	CELULAR IPHONE 6S	$560.000	2	6
9	CELULAR LG	$450.000	2	7
10	MOUSE BLUETOOTH	$28.990	5	9
11	MONITOR 17" SAMSUNG	$150.000	4	7
12	MONITOR 17" LENOVO	$250.000	4	7
13	MONITOR 17" LG	$125.000	4	4
14	MONITOR 15" LENOVO	$200.000	4	3
15	MOTO G 2013	$130.000	2	45
16	MOTO G 2015	$180.000	2	32
17	MOTO X PLAY	$370.000	2	12
18	IPAD MINI 2	$200.000	6	4
19	IPAD AIR	$260.000	6	2
20	NOTEBOOK MSI	$850.000	1	4
21	MOUSE MICROSOFT	$8.500	5	5

Cursores (Sintaxis)

CURSOR
PRODUCTOS

PRODUCTO				
CODIGO_PRODUCTO	NOMBRE	PRECIO	CODIGO_CATEGORIA	STOCK
1	NOTEBOOK LENOVO	$380.000	1	20
2	CELULAR MOTOROLA	$110.000	2	15
3	AUDIFONOS MACROTEL	$3.500	3	12
4	NOTEBOOK SAMSUNG	$500.000	1	3
5	MONITOR 17" AOC	$67.990	4	80
6	MONITOR 21" DELL	$81.990	4	20
7	NOTEBOOK HP	$280.990	1	1
8	CELULAR IPHONE 6S	$560.000	2	6
9	CELULAR LG	$450.000	2	7
10	MOUSE BLUETOOTH	$28.990	5	9
11	MONITOR 17" SAMSUNG	$150.000	4	7
12	MONITOR 17" LENOVO	$250.000	4	7
13	MONITOR 17" LG	$125.000	4	4
14	MONITOR 15" LENOVO	$200.000	4	3
15	MOTO G 2013	$130.000	2	45
16	MOTO G 2015	$180.000	2	32
17	MOTO X PLAY	$370.000	2	12
18	IPAD MINI 2	$200.000	6	4
19	IPAD AIR	$260.000	6	2
20	NOTEBOOK MSI	$850.000	1	4
21	MOUSE MICROSOFT	$8.500	5	5

Cursores (Sintaxis)

Una vez almacenado nuestro SELECT en el cursor “PRODUCTOS”, puede recorrerse fila por fila utilizando un conjunto de instrucciones específicas.

Cursores (Sintaxis)

Una vez almacenado nuestro SELECT en el cursor “PRODUCTOS”, puede recorrerse fila por fila utilizando un conjunto de instrucciones específicas.

Atributo	Tipo	Descripción
OPEN	Instrucción	Abre un cursor
FETCH	Instrucción	Carga la fila actual y sus valores
CLOSE	Instrucción	Cierra un cursor
%ISOPEN	Booleano	Verifica si el cursor está abierto
%NOTFOUND	Booleano	Verifica si la fila es nula
%FOUND	Booleano	Verifica si la fila no es nula
%ROWCOUNT	Entero	Proporciona el número total de filas hasta el momento

Cursores (Sintaxis)

Una vez almacenado nuestro SELECT en el cursor “PRODUCTOS”, puede recorrerse fila por fila utilizando un conjunto de instrucciones específicas.

Atributo	Tipo	Descripción
OPEN	Instrucción	Abre un cursor
FETCH	Instrucción	Carga la fila actual y sus valores
CLOSE	Instrucción	Cierra un cursor
%ISOPEN	Booleano	Verifica si el cursor está abierto
%NOTFOUND	Booleano	Verifica si la fila es nula
%FOUND	Booleano	Verifica si la fila no es nula
%ROWCOUNT	Entero	Proporciona el número total de filas hasta el momento

Para ver como funciona utilizamos un bloque anónimo.

Cursores (Ejemplo)

```
DECLARE
    CURSOR PRODUCTOS IS SELECT * FROM PRODUCTO;
BEGIN

END;
```

El cursor es una variable, por lo tanto debe definirse previamente.

Cursores (Ejemplo)

```
DECLARE
    CURSOR PRODUCTOS IS SELECT * FROM PRODUCTO;
BEGIN

END;
```

El cursor es una variable, por lo tanto debe definirse previamente.

Para utilizar un cursor, primero debe abrirse utilizando la instrucción OPEN.

Cursores (Ejemplo)

```
DECLARE
    CURSOR PRODUCTOS IS SELECT * FROM PRODUCTO;
BEGIN

END;
```

El cursor es una variable, por lo tanto debe definirse previamente.

Para utilizar un cursor, primero debe abrirse utilizando la instrucción OPEN.

```
DECLARE
    CURSOR PRODUCTOS IS SELECT * FROM PRODUCTO;
BEGIN
    OPEN PRODUCTOS;
END;
```

Cursores (Ejemplo)

Para recorrer el cursor, es necesario iterar utilizando la estructura LOOP (equivalente a un FOR o WHILE).

Cursores (Ejemplo)

Para recorrer el cursor, es necesario iterar utilizando la estructura LOOP (equivalente a un FOR o WHILE).

Dentro de la iteración, utilizamos la instrucción FETCH para extraer las columnas de la fila en curso.

Cursores (Ejemplo)

Para recorrer el cursor, es necesario iterar utilizando la estructura LOOP (equivalente a un FOR o WHILE).

Dentro de la iteración, utilizamos la instrucción FETCH para extraer las columnas de la fila en curso.

```
DECLARE
    CURSOR PRODUCTOS IS SELECT * FROM PRODUCTO;
BEGIN
    OPEN PRODUCTOS;
    LOOP
        FETCH PRODUCTOS;
    END LOOP;
END;
```

Cursores (Ejemplo)

Para recorrer el cursor, es necesario iterar utilizando la estructura LOOP (equivalente a un FOR o WHILE).

Dentro de la iteración, utilizamos la instrucción FETCH para extraer las columnas de la fila en curso.

```
DECLARE
    CURSOR PRODUCTOS IS SELECT * FROM PRODUCTO;
BEGIN
    OPEN PRODUCTOS;
    LOOP
        FETCH PRODUCTOS;
    END LOOP;
END;
```

Pero, ¿cómo obtenemos las columnas de la fila actual?.

Cursores (Ejemplo)

Para obtenerlas, es necesario declarar variables que capturen las columnas del cursor.

Cursores (Ejemplo)

Para obtenerlas, es necesario declarar variables que capturen las columnas del cursor.

```
DECLARE
    CODIGOP NUMBER;
    NOMBREP VARCHAR2(30);
    PRECIOP NUMBER;
    CODIGOC NUMBER;
    STOCKP NUMBER;
    CURSOR PRODUCTOS IS SELECT * FROM PRODUCTO;
BEGIN
    OPEN PRODUCTOS;
    LOOP
        FETCH PRODUCTOS INTO CODIGOP,NOMBREP,PRECIOP,CODIGOC,STOCKP;
    END LOOP;
END;
```

Cursores (Ejemplo)

Para obtenerlas, es necesario declarar variables que capturen las columnas del cursor.

```
DECLARE
    CODIGOP NUMBER;
    NOMBREP VARCHAR2(30);
    PRECIOP NUMBER;
    CODIGOC NUMBER;
    STOCKP NUMBER;
    CURSOR PRODUCTOS IS SELECT * FROM PRODUCTO;
BEGIN
    OPEN PRODUCTOS;
    LOOP
        FETCH PRODUCTOS INTO CODIGOP,NOMBREP,PRECIOP,CODIGOC,STOCKP;
    END LOOP;
END;
```

Ahora, es posible trabajar con las columnas de la fila en curso.

Cursores (Ejemplo)

Una vez que el cursor recorre todas las filas del SELECT es necesario establecer una instrucción de parada y finalmente, cerrar el cursor.

Cursores (Ejemplo)

Una vez que el cursor recorre todas las filas del SELECT es necesario establecer una instrucción de parada y finalmente, cerrar el cursor.

```
DECLARE
    CODIGOP NUMBER;
    NOMBREP VARCHAR2(30);
    PRECIOP NUMBER;
    CODIGOC NUMBER;
    STOCKP NUMBER;
    CURSOR PRODUCTOS IS SELECT * FROM PRODUCTO;
BEGIN
    OPEN PRODUCTOS;
    LOOP
        FETCH PRODUCTOS INTO CODIGOP,NOMBREP,PRECIOP,CODIGOC,STOCKP;

        EXIT WHEN PRODUCTOS%NOTFOUND;
    END LOOP;
    CLOSE PRODUCTOS;
END;
```

Cursores (Ejemplo)

Entre la instrucción FETCH y EXIT WHEN es donde podemos trabajar con las columnas de la fila en curso.

```
DECLARE
    CODIGOP NUMBER;
    NOMBREP VARCHAR2(30);
    PRECIOP NUMBER;
    CODIGOC NUMBER;
    STOCKP NUMBER;
    CURSOR PRODUCTOS IS SELECT * FROM PRODUCTO;
BEGIN
    OPEN PRODUCTOS;
    LOOP
        FETCH PRODUCTOS INTO CODIGOP,NOMBREP,PRECIOP,CODIGOC,STOCKP;
        DBMS_OUTPUT.PUT_LINE('LEYENDO LA FILA: '||PRODUCTOS%ROWCOUNT);
        IF STOCKP > 10 THEN
            DBMS_OUTPUT.PUT_LINE('CODIGO: '||CODIGOP||' - NOMBRE: '||NOMBREP);
        END IF;
        EXIT WHEN PRODUCTOS%NOTFOUND;
    END LOOP;
    CLOSE PRODUCTOS;
END;
```

Cursores (Ejemplo)

El resultado del cursor anterior:

Cursores (Ejemplo)

El resultado del cursor anterior:

```
DECLARE
    CODIGOP NUMBER;
    NOMBREP VARCHAR2(30);
    PRECIOP NUMBER;
    CODIGOC NUMBER;
    STOCKP NUMBER;
    CURSOR PRODUCTOS IS SELECT * FROM PRODUCTO;
BEGIN
    OPEN PRODUCTOS;
    LOOP
        FETCH PRODUCTOS INTO CODIGOP,NOMBREP,PRECIOP,CODIGOC,STOCKP;
        DBMS_OUTPUT.PUT_LINE('LEYENDO LA FILA: '||PRODUCTOS%ROWCOUNT);
        IF STOCKP > 10 THEN
            DBMS_OUTPUT.PUT_LINE('CODIGO: '||CODIGOP||' - NOMBRE: '||NOMBREP);
        END IF;
        EXIT WHEN PRODUCTOS%NOTFOUND;
    END LOOP;
    CLOSE PRODUCTOS;
END;
```

```
LEYENDO LA FILA: 1
CODIGO: 1 - NOMBRE: NOTEBOOK LENOVO
LEYENDO LA FILA: 2
CODIGO: 2 - NOMBRE: CELULAR MOTOROLA
LEYENDO LA FILA: 3
CODIGO: 3 - NOMBRE: AUDIFONOS MACROTEL
LEYENDO LA FILA: 4
LEYENDO LA FILA: 5
CODIGO: 5 - NOMBRE: MONITOR 17" AOC
LEYENDO LA FILA: 6
CODIGO: 6 - NOMBRE: MONITOR 21" DELL
LEYENDO LA FILA: 7
LEYENDO LA FILA: 8
LEYENDO LA FILA: 9
LEYENDO LA FILA: 10
LEYENDO LA FILA: 11
LEYENDO LA FILA: 12
LEYENDO LA FILA: 13
LEYENDO LA FILA: 14
LEYENDO LA FILA: 15
CODIGO: 15 - NOMBRE: MOTO G 2013
LEYENDO LA FILA: 16
CODIGO: 16 - NOMBRE: MOTO G 2015
LEYENDO LA FILA: 17
CODIGO: 17 - NOMBRE: MOTO X PLAY
LEYENDO LA FILA: 18
LEYENDO LA FILA: 19
LEYENDO LA FILA: 20
LEYENDO LA FILA: 21
LEYENDO LA FILA: 21
```

Secuencias

Una secuencia es un objeto de la base de datos para generar enteros únicos.

Secuencias

Una secuencia es un objeto de la base de datos para generar enteros únicos.

Esto es conocido en otros motores de base de datos como AUTOINCREMENT.

Secuencias

Una secuencia es un objeto de la base de datos para generar enteros únicos.

Esto es conocido en otros motores de base de datos como AUTOINCREMENT.

La secuencia puede ser de gran utilidad para generar claves primarias.

Secuencias (Sintaxis)

```
CREATE SEQUENCE NOMBRE_SECUENCIA;
```

Esta es la sintaxis simple para la creación de una secuencia. Por defecto, se crea una secuencia ascendente que comienza en 1, incrementando de 1 en 1 sin un límite definido.

Secuencias (Sintaxis)

```
CREATE SEQUENCE NOMBRE_SECUENCIA;
```

Esta es la sintaxis simple para la creación de una secuencia. Por defecto, se crea una secuencia ascendente que comienza en 1, incrementando de 1 en 1 sin un límite definido.

Sin embargo, la secuencia ofrece una serie de opciones para su creación y funcionamiento.

Secuencias (Sintaxis)

```
CREATE SEQUENCE NOMBRE_SECUENCIA;
```

Esta es la sintaxis simple para la creación de una secuencia. Por defecto, se crea una secuencia ascendente que comienza en 1, incrementando de 1 en 1 sin un límite definido.

Sin embargo, la secuencia ofrece una serie de opciones para su creación y funcionamiento.

Opción	Descripción
INCREMENT BY	Especifica entre el intervalo de números
START WITH	Especifica el primer número de la secuencia
MAXVALUE	Especifica el valor máximo que la secuencia puede generar
MINVALUE	Especifica el valor mínimo que la secuencia puede generar
CYCLE	Especifica que la secuencia se reinicia al llegar al máximo valor
ORDER	Permite que la secuencia de números sea generada en orden de petición

Secuencia (Ejemplo)

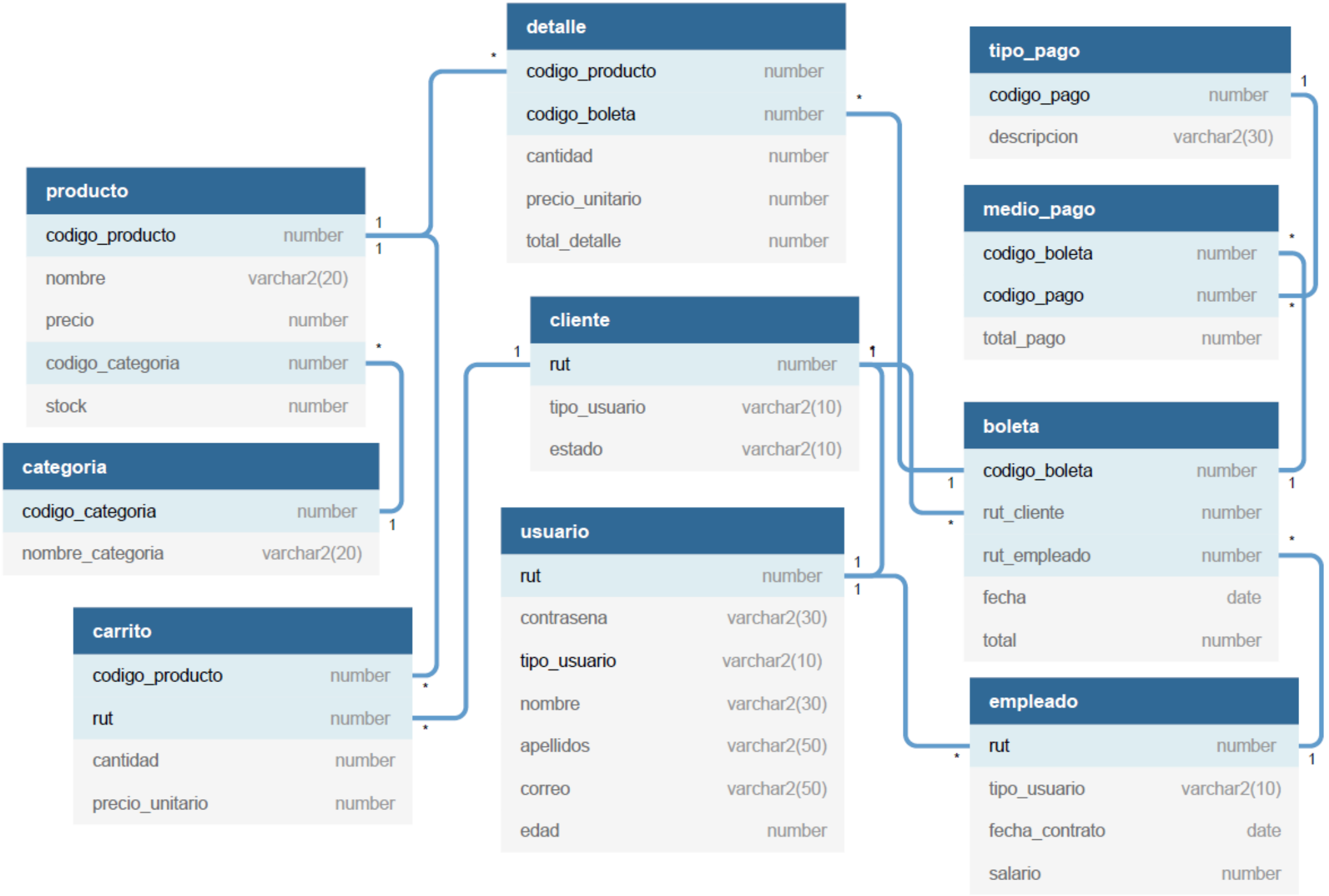
```
CREATE SEQUENCE GENERADOR_PK_PRODUCTO
START WITH 1
INCREMENT BY 1
MAXVALUE 10000;

DECLARE
    NUMERO NUMBER;
BEGIN
    NUMERO := GENERADOR_PK_PRODUCTO.NEXTVAL;
    DBMS_OUTPUT.PUT_LINE('EL NÚMERO OBTENIDO ES: '||NUMERO);
END;
```

Ejercicios

Considerando el modelo final de la siguiente diapositiva:

Realice los procedimientos de inserción, modificación y borrado para las tablas cliente, boleta, detalle, producto, carrito.

Utilice procedimientos, funciones, secuencias y cursores (si es que lo requiere).


detalle
codigo_producto number
codigo_boleta number
cantidad number
precio_unitario number
total_detalle number
tipo_pago
codigo_pago number
descripcion varchar2(30)
producto
codigo_producto number
nombre varchar2(20)
precio number
codigo_categoria number
stock number
medio_pago
codigo_boleta number
codigo_pago number
total_pago number
cliente
rut number
tipo_usuario varchar2(10)
estado varchar2(10)
boleta
codigo_boleta number
rut_cliente number
rut_empleado number
fecha date
total number
categoria
codigo_categoria number
nombre_categoria varchar2(20)
usuario
rut number
contrasena varchar2(30)
tipo_usuario varchar2(10)
nombre varchar2(30)
apellidos varchar2(50)
correo varchar2(50)
edad number
carrito
codigo_producto number
rut number
cantidad number
precio_unitario number
empleado
rut number
tipo_usuario varchar2(10)
fecha_contrato date
salario number


UNIVERSIDAD CATOLICA DEL MAULE
CHILE
ucm
UNIVERSIDAD CATOLICA DEL MAULE