Bases de datos II

06/02/2021 - 10/02/2021

Apuntes de la clase 06/02/2021 - 10/02/2021

IS-601

Emilson Acosta

Clase #6

Generar Diagrama relacional

1. Conectarse a la base de datos
2. Click en file
3. Data modeler → Importar
4. Data directionary
5. Seleccionar la conexión para la cual se generara el diagrama
6. Click en next
7. Elegir esquema es decir en donde se encuentra la base de datos
8. En importar a debe estar: Nuevo modelo relacional
9. Click en next
10. Seleccionar todo
11. Click en next
12. Click en finish

Importar diagrama relacional a PDF

1. Click en file
2. Data modeler
3. Print Diagram
4. To PDF

Clase #7

SELECT INTO dentro de los bloques anónimos o proceso almacenado.

Para usar el select into se necesita declarar una variable para almacenar uno o más valores.

Creación de variable

Se debe usar la misma cantidad de bytes que almacenara la variable si es tipo varchar2 o se puede usar menos para limitar la cantidad de leras que pueda tener un campo, solo funciona cuando se devuelve un valor.

- Comando: NOMBRE VARCHAR(255);

Ejemplo

```
/*
  @author edgar.benedetto@unah.hn
  @date 14/02/2021
  @version 1.0

  CLASE #7.1 Y #7.2
*/
DECLARE
  VAR_CAPITAL VARCHAR(255);
  NUM_CAPITAL_ID CAPITALES.ID%TYPE;
  VAR_PAIS PAISES.NOMBRE%TYPE;
  NUM_PAIS_ID PAISES.ID%TYPE;
  NUM_POBLACION POBLACIONES.POBLACION%TYPE;
BEGIN

  SELECT NOMBRE, ID INTO VAR_CAPITAL, NUM_CAPITAL_ID FROM CAPITALES WHERE ID=2;
```

```
  SELECT NOMBRE,ID INTO VAR_PAIS, NUM_PAIS_ID FROM PAISES WHERE CAPITAL =
NUM_CAPITAL_ID;

  SELECT POBLACION INTO NUM_POBLACION FROM POBLACIONES WHERE PAIS =
NUM_PAIS_ID AND ANYO = 2006;



  DBMS_OUTPUT.PUT_LINE('EL NOMBRE DE LA CAPITAL CON ID=2 ES: '||VAR_CAPITAL);

  DBMS_OUTPUT.PUT_LINE('EL NOMBRE DEL PAIS ES: '||VAR_PAIS);

  DBMS_OUTPUT.PUT_LINE('LA POBLACION EN EL 2006 ERA DE: '||NUM_POBLACION);

END;
```

Clase #8

Cursor: Instrucción que se utiliza cuando se quiere recorrer varios registros de una consulta.

Se puede tener un cursor implícito o SQL, esta instrucción usa cursor implícito o SQL:

- Comando: SELECT NOMBRE, ID INTO VAR_CAPITAL, NUM_CAPITAL_ID FROM CAPITALES WHERE ID=2;

Llamado al cursor implícito:

- Comando: DBMS_OUTPUT.PUT_LINE('LA CANTIDAD DE REGISTROS ES: '||SQL %ROWCOUNT);

Estructura del cursor:

- Comando: CURSOR NombreCursor IS Consulta;

En el cuerpo del bloque anónimo se usa OPEN para poder usar el cursor junto al nombre:

- Comando: OPEN NombreCursor;

Indicar que se posiciones en el siguiente registro:

- Comando: FETCH NombreCursor INTO NombreVariable;

Variable de cursor NOTFOUND que devuelve TRUE o FALSE:

- Comando: EXIT WHEN CUR_POBLACION%NOTFOUND;

Cerrar el cursor:

- Comando: CLOSE NombreCursor;

Variable de cursor ROWCOUNT en un cursor explicito devuelve la posición actual del cursor, en un ciclo FOR no funcionaria.

- Comando: DBMS_OUTPUT.PUT_LINE('LA CANTIDAD DE REGISTROS ES: '|| CUR_POBLACION%ROWCOUNT);

Obtener información de varias tablas con cursores.

Cursor con in inner join en el select

- Comando: CURSOR CDATOS_PRODUCTO IS SELECT PRODUCTS.PRODUCT_NAMEN, CATEGORIES.CATEGORY_NAME, PRODUCTS.MODEL_YEAR FROM PRODUCTS INNER JOIN CATEGORIES ON PRODUCTS.CATEGORY_ID = CATEGORIES.CATEGORY_ID;

Obtener el tipo de dato y mapear la cantidad de filas/columnas de un cursor con multiples tablas usando %ROWTYPE, convirtiendo a la variable en una matriz.

- Comando: REGISTRO CDATOS_PRODUCTO%ROWTYPE;

Se puede acceder al valor de cada columna a través del nombre del campo.

- Comando: REGISTRO.MODEL_YEAR;

Uso de Alias en Prostege SQL

- **Comando:** SELECT PRODUCT_NAME NOMBRE FROM PRODUCTS;

Impresión de campos con alias

- **Comando:** DBMS_OUTPUT.PUT_LINE('EL NOMBRE DEL PRODUCTO ES: '|| REGISTRO.NOMBRE);

Ejemplo

--- Clase #8.2

```
--- Obtener datos a un cursor de distintas tablas


DECLARE

  --- NOMBRE_PRODUCTO PRODUCTS.PRODUCT_NAME%TYPE;

  --- MODELO_PRODUCTO PRODUCTS.MODEL_YEAR%TYPE;

  --- NOMBRE_CATEGORIA CATEGORIES.CATEGORY_NAME%TYPE;

  CURSOR CDATOS_PRODUCTO IS SELECT PRODUCTS.PRODUCT_NAME,
CATEGORIES.CATEGORY_NAME, PRODUCTS.MODEL_YEAR,

  CATEGORIES.CATEGORY_ID CAT_ID, PRODUCTS.CATEGORY_ID PROD_CAT_ID
FROM PRODUCTS INNER JOIN CATEGORIES ON PRODUCTS.CATEGORY_ID =
CATEGORIES.CATEGORY_ID;


  --- Uniendo las 3 variables en una sola

  REGISTRO CDATOS_PRODUCTO%ROWTYPE;


BEGIN

 OPEN CDATOS_PRODUCTO;


 LOOP

  FETCH  CDATOS_PRODUCTO INTO REGISTRO; ---NOMBRE_PRODUCTO,
NOMBRE_CATEGORIA, MODELO_PRODUCTO;


  EXIT WHEN CDATOS_PRODUCTO%NOTFOUND;


  DBMS_OUTPUT.PUT_LINE('EL NOMBRE DEL PRODUCTO ES: '||
REGISTRO.PRODUCT_NAME);
```

```
   DBMS_OUTPUT.PUT_LINE('EL MODELO DEL PRODUCTO ES DEL: '||
REGISTRO.MODEL_YEAR);

   DBMS_OUTPUT.PUT_LINE('EL NOMBRE DE LA CATEGORIA DEL PRODUCTO ES: '||
REGISTRO.CATEGORY_NAME);

   DBMS_OUTPUT.PUT_LINE('EL ID DE LA CATEGORIA DEL PRODUCTO ES: '||
REGISTRO.CAT_ID);

   DBMS_OUTPUT.PUT_LINE('EL ID DE LA CATEGORIA DEL PRODUCTO ES: '||
REGISTRO.PROD_CAT_ID);

   DBMS_OUTPUT.PUT_LINE('LA CANTIDAD DE REGISTROS ES: '||CDATOS_PRODUCTO
%ROWCOUNT);


   --- Salto de linea

   DBMS_OUTPUT.PUT_LINE(CHR(13));

  END LOOP;


  CLOSE CDATOS_PRODUCTO;

END;
```

Crear estructura tipo record para almacenar registros que devuelve un cursor

La variable tipo record debe tener la misma cantidad de campos que los que se van a retornar.

- Comando:
```
TYPE FILA IS RECORD(
NOMBRE_PRODUCTO PRODUCT.PRODUCT_NAME%TYPE,
NOMBRE_CATEGORIA CATEGORIES.CATEGORY_NAME%TYPE,
ANIO_MODELO NUMBER,
COD_CATEGORIA NUMBER
);
```

Bulk Collect: Se utiliza cuando se quieren obtener todos los registros de una sola vez

PLS INTEGER: Tipo de datos en Oracle para poder definir de forma unico los registros de una estructura especifica.

Estructura tipo tabla, y la tabla sera de tipo fila o RECORD

- Comando: TYPE TABLA_CATEGORIAS IS TABLE OF FILA INDEX BY PLS_INTEGER;

La cláusula del BULK COLLECT se utiliza antes del INTO del SELECT el cual luego se toma como cursor implícito para el cual se debe usar la variable SQL, cuando se usa el cursor implícito el ROWCOUNT muestra la cantidad total de registros que hay en la consulta.

- Comando: SELECT CATEGORIES.CATEGORY_ID, CATEGORIES.CATEGORY_NAME BULK COLLECT INTO DATOS_CATEGORIA FROM CATEGORIES;
- Comando: DBMS_OUTPUT.PUT_LINE('LA CANTIDAD DE REGISTROS ES: '||SQL %ROWCOUNT);

Ejemplo

```
--- Clase #8.3

--- Crear estructura tipo record para almacenar registros que devuelve un cursor


DECLARE
  --- Variable tipo RECORD
  TYPE FILA IS RECORD(
    NOMBRE_PRODUCTO PRODUCTS.PRODUCT_NAME%TYPE,
    NOMBRE_CATEGORIA CATEGORIES.CATEGORY_NAME%TYPE,
    ANIO_MODELO NUMBER,
    COD_CATEGORIA NUMBER
  );
```

```
  CURSOR CDATOS_PRODUCTO IS SELECT PRODUCTS.PRODUCT_NAME,
CATEGORIES.CATEGORY_NAME, PRODUCTS.MODEL_YEAR,
  CATEGORIES.CATEGORY_ID CAT_ID FROM PRODUCTS INNER JOIN CATEGORIES
ON PRODUCTS.CATEGORY_ID = CATEGORIES.CATEGORY_ID;

  --- Variable tipo Fila la cual es tipo record ---> DATOS_CURSOR es tipo RECORD
  DATOS_CURSOR FILA;

BEGIN
 OPEN CDATOS_PRODUCTO;

 LOOP
  FETCH  CDATOS_PRODUCTO INTO DATOS_CURSOR;

  DBMS_OUTPUT.PUT_LINE('EL NOMBRE DEL PRODUCTO ES: '||
DATOS_CURSOR.NOMBRE_PRODUCTO);
  DBMS_OUTPUT.PUT_LINE('EL MODELO DEL PRODUCTO ES DEL: '||
DATOS_CURSOR.ANIO_MODELO);
  DBMS_OUTPUT.PUT_LINE('EL NOMBRE DE LA CATEGORIA DEL PRODUCTO ES: '||
DATOS_CURSOR.NOMBRE_CATEGORIA);
  DBMS_OUTPUT.PUT_LINE('EL ID DE LA CATEGORIA DEL PRODUCTO ES: '||
DATOS_CURSOR.COD_CATEGORIA);
  DBMS_OUTPUT.PUT_LINE('LA CANTIDAD DE REGISTROS ES: '||
CDATOS_PRODUCTO%ROWCOUNT);
```

```
    EXIT WHEN CDATOS_PRODUCTO%NOTFOUND;


    --- Salto de linea
    DBMS_OUTPUT.PUT_LINE(CHR(13));
  END LOOP;


  CLOSE CDATOS_PRODUCTO;
END;


/*
   Obtener los nombres de todas las categorias y los codigos de las mismas.
   Bulk Collect --> se utiliza cuando se quieren obtener todos los registros de una sola vez
*/
DECLARE
   TYPE FILA IS RECORD(
     CODIGO_CATEGORIA CATEGORIES.CATEGORY_ID%TYPE,
     NOMBRE_CATEGORIA CATEGORIES.CATEGORY_NAME%TYPE
   );


   --- PLS INTEGER --> Tipo de datos en Oracle para poder definir de forma unico los registros de una estructura especifica.
   --- Estructura tipo tabla, y la tabla sera de tipo fila o RECORD.
   TYPE TABLA_CATEGORIAS IS TABLE OF FILA INDEX BY PLS_INTEGER;
```

```
  --- Variable para almacenar todos los registros del Bulk Collect
  DATOS_CATEGORIA TABLA_CATEGORIAS;

  --- Variable para recorrer los registros obtenidos
  ITERACION NUMBER(10):= 0;
BEGIN
  SELECT CATEGORIES.CATEGORY_ID, CATEGORIES.CATEGORY_NAME BULK
COLLECT INTO DATOS_CATEGORIA FROM CATEGORIES;

  DBMS_OUTPUT.PUT_LINE('LA CANTIDAD DE REGISTROS ES: '||SQL%ROWCOUNT);

  --- Ciclo WHILE
  WHILE (ITERACION<SQL%ROWCOUNT) LOOP
   DBMS_OUTPUT.PUT_LINE('EL CODIGO DE LA CATEGORIA ES: '||
DATOS_CATEGORIA(ITERACION+1).CODIGO_CATEGORIA);
   DBMS_OUTPUT.PUT_LINE('EL NOMBRE DE LA CATEGORIA ES: '||
DATOS_CATEGORIA(ITERACION+1).NOMBRE_CATEGORIA);

   ITERACION:=ITERACION+1;
  END LOOP;
END;
```

Clase #8.4

Cursores y arreglos asociativos

Tabla creada a base de un mapeo de una tabla de los registros

- Comando: TYPE TABLA_CATEGORIAS IS TABLE OF CATEGORIES%ROWTYPE INDEX BY PLS_INTEGER;

Arreglo asociativo: Es un elemento donde se guardan valores.

PRIOR: La tarea de esta función es Retornar el número de índice anterior al que se mande como parámetro**.**

- **Comando:** DBMS_OUTPUT.PUT_LINE(DATOS.PRIOR(3));

NEXT: La tarea de esta función es Retornar el número de índice siguiente al que se mande como parámetro**.**

- **Comando:** DBMS_OUTPUT.PUT_LINE(DATOS.NEXT(3));

DELETE: La tarea de esta función es eliminar el número de índice que se mande como parámetro**.**

- **Comando:** DATOS.DELETE(1);

Variante de DELETE borra a partir de rangos de índices

- **Comando:** DATOS.DELETE(2,4);

Clase #9

Funciones para el manejo de fechas

Tipo de dato DATE: Solo almacena la fecha

Tipo de dato TIMESTAMP: Almacena fecha y hora.

Cuando no se especifica formato, se le asigna el formato por defecto de ORACLE.

TO_DATE() : Permite indicar la fecha e interpretarla en un formato indicado. El primer parámetro es la fecha a interpretar y el segundo parámetro es el formato.

TO_TIMESTAMP() : Permite indicar la fecha y hora en un formato especifico o se puede dejar en el formato por defeco de Oracle.Para indicar la hora se usa HH para un formato de 12 horas ,HH24 para un formato 24 horas, MI es para minutos y SS para segundos.

TO_CHAR() : Para cambiar el formato de impresión de la fecha y hora.

USER: Obtiene el usuario con el que se realizo la conexión con la base de datos

SUBSTR: Obtener una subcadena de una cadena. Primer parámetro es la cadena original, segundo parámetro es de donde empieza la subcadena y el tercer parámetro es cuantos caracteres obtendrá.

Ejemplo

```
DECLARE
        FECHA DATE;
        FECHA_HORA TIMESTAMP;
BEGIN
        FECHA := SYSDATE;
        DBMS_OUTOUT.PUT_LINE('LA FECHA OBTENIDA CON SISTEMA ES: '||FECHA);
        FECHA := TO_DATE('2018-05-28', 'YYYY-MM-DD');
        DBMS_OUTOUT.PUT_LINE('LA FECHA ASIGNADA CON TO_DATE ES: '||FECHA);
        FECHA_HORA := SYSTIMESTAMP;
        DBMS_OUTOUT.PUT_LINE('LA FECHA Y HORA OBTENIDA CON SISTEMA ES: '||
FECHA_HORA );
        FECHA_HORA := TO_TIMESTAMP('25-02-2020 11:30:16','DD-MM-YYYY
HH24:MI:SS');
        DBMS_OUTOUT.PUT_LINE('LA FECHA Y HORA OBTENIDA CON TO_TIMESTAMP
ES: '||FECHA_HORA );
        DBMS_OUTOUT.PUT_LINE('USO DE TO_CHAR PARA CAMBIAR EL FORMATO DE
IMPRESIÓN DE LA FECHA: '||TO_CHAR(FECHA, 'DD-MM-YYYY' ));
        DBMS_OUTOUT.PUT_LINE('USO DE TO_CHAR PARA CAMBIAR EL FORMATO DE
IMPRESIÓN DE LA FECHA Y HORA: '||TO_CHAR(FECHA_HORA, 'DD-MM-YYYY HH-MI-SS
AM' ));
```

DBMS_OUTOUT.PUT_LINE('EL USUARIO CONTECTADO ES: : '||USER);

END;

SELECT ORDER_DATE, TO_CHAR(TO_DATE(ORDER_DATE, 'YYYYMMDD'), 'YYYY-MM-DD'), TO_TIMESTAMP(ORDER_DATE, 'YYYYMMDD HH:MI:SS') FROM ORDERS;

SELECT CATEGORY_NAME, SUBSTR(CATEGORY_NAME, 1, 5) FROM CATEGORIES;

Clase #10

Gestion de excepciones: Cuando se quieren capturar los errores que se generen en PL/SQL, se pueden usar en los triggers, funciones y procedimientos.Se usa la palabra reservada EXCEPTION para capturar los errores que se puedan generar, se puede usar el nombre de la excepción.

SQLCODE: Retorna el código del error el cual es númerico y Oracle lo genera.

SQLERRM: Retorna el mensaje del error.

OTHERS: En las excepciones others permite controlar las excepciones que no se hayan declarado previamente.

Los códigos de error pueden asignarse a una variable, almacenando el código y el mensaje

Excepción	Disparada cuando…
ACCESS_INTO_NULL	Se intenta asignar valores a los atributos de un Objeto no inicializado.
COLLECTION_IS_NULL	Se intenta aplicar métodos predefinidos (exceptuando EXISTS), sobre colecciones de Objetos como varrays o tablas de memoria no inicializados; o también cuando se intenta asignar valores a los elementos de una colección de Objetos que no ha sido inicializada.
CURSOR_ALREADY_OPEN	Se intenta abrir un cursor que ya está abierto.
DUP_VAL_ON_INDEX	Se intenta almacenar valores duplicados en una columna de Bbdd que tiene una constraint tipo Unique sobre esa columna.

INVALID_CURSOR	Se intenta realizar una operación no válida sobre un cursor, como por ejemplo cerrar un cursor que no ha sido abierto.
INVALID_NUMBER	En una sentencia SQL, la conversión de un string de caracteres a un número falla, porque el string no representa un número válido. En sentencias aisladas dentro de un procedimiento PL/SQL, se dispara la excepción VALUE_ERROR para este mismo caso.
LOGIN_DENIED	Se intenta entrar en Oracle con un username o password no válidos.
NO_DATA_FOUND	Una sentencia SELECT INTO no devuelve valores, o bien se referencia a un elemento borrado en una tabla de memoria, o bien se referencia a un elemento no inicializado en una tabla de memoria.
NOT_LOGGED_ON	El programa PL/SQL realiza una llamada a Bbdd sin estar conectado a Oracle.

PROGRAM_ERROR	PL/SQL tiene un problema interno.
ROWTYPE_MISMATCH	La variable del cursor del host, y la variable del cursor de PL/SQL, implicados en una sentencia de asignación, tienen tipos incompatibles.
STORAGE_ERROR	PL/SQL se queda sin memoria, o la memoria está estropeada.
SUBSCRIPT_BEYOND_COUNT	Se referencia a un elemento de un varray o tabla de memoria, utilizando un índice mayor que el número más largo que tiene como índice el varray o tabla de memoria.
SUBSCRIPT_OUTSIDE_LIMIT	Se referencia a un elemento de un varray o tabla de memoria, utilizando un índice que está fuera del rango legal (Por ejemplo –1)
TIMEOUT_ON_RESOURCE	Se produce un time-out cuando Oracle está esperando por un recurso.
TOO_MANY_ROWS	Una sentencia SELECT INTO devuelve más de una fila.

TIMEOUT_ON_RESOURCE	Se produce un time-out cuando Oracle está esperando por un recurso.
TOO_MANY_ROWS	Una sentencia SELECT INTO devuelve más de una fila.

VALUE_ERROR	Se produce algún error de tipo aritmético, conversión, truncate, o constraint de tipo size.
ZERO_DIVIDE	Se intenta dividir entre cero.

Ejemplo

```
DECLARE
        NOMBRE PRODUCTS.PRODUCT_NAME%TYPE;
        FECHA DATE;
        CODIGO_ERROR NUMBER;
        MSJ_ERROR VARCHAR2(1000);

BEGIN
        FECHA:='HOLA MUNDO';
        CODIGO_ERROR:=SQLCODE;
        MSJ_ERROR:= SQLERRM;
        SELECT PRODUCT_NAME INTO NOMBRE FROM PRODUCTS ---WHERE
PRODUCT_ID=10000;
        DBMS_OUTPUT.PUT_LINE('EL NOMBRE DEL PRODUCTO CON ID 1 ES: '||
NOMBRE);
        EXCEPTION
                WHEN TOO_MANY_ROWS THEN
                        CODIGO_ERROR:=SQLCODE;
                        MSJ_ERROR:= SQLERRM;
                        DBMS_OUTPUT.PUT_LINE('SE RETORNAN MUCHOS REGISTROS
        EN LA SETENCIA SELECT, EL CODIGO DEL ERROR ES: '||CODIGO_ERROR);
                        DBMS_OUTPUT.PUT_LINE(MSJ_ERROR);
                WHEN OTHERS
                        IF(SQLCODE=-1858) THEN
                                CODIGO_ERROR:=SQLCODE;
```

```
                MSJ_ERROR:= SQLERRM;
                DBMS_OUTPUT.PUT_LINE('SE ESTAN ASIGNANDO TIPOS DE
DATOS INCOMPATIBLES');
                DBMS_OUTPUT.PUT_LINE(' EL CODIGO DEL ERROR ES: '||
        CODIGO_ERROR);
                DBMS_OUTPUT.PUT_LINE('EL MENSAJE DE ERROR ES: '||
        MSJ_ERROR);
        ENF IF;
        IF(SQLCODE= 100) THEN
                CODIGO_ERROR:=SQLCODE;
                MSJ_ERROR:= SQLERRM;
                DBMS_OUTPUT.PUT_LINE('EL SELECT INTO NO ENCONTRO
DATOS');
                DBMS_OUTPUT.PUT_LINE(' EL CODIGO DEL ERROR ES: '||
        CODIGO_ERROR);
                DBMS_OUTPUT.PUT_LINE('EL MENSAJE DE ERROR ES: '||
        MSJ_ERROR);
        END IF;

END;
```

Clase #11

Gestión de transacciones vinculado con la gestión de excepciones

Transacción Constituye un conjunto de operaciones SQL que se realizan sobre la base, modificando los datos que ya se tienen almacenados en la base de datos, se puede realizar sobre una o muchas tablas, afectando a una o muchas tablas. Al ser la transacción exitosa Se debe garantizar que los cambios se almacenen de forma permanente en la base de datos, si la

transacción finaliza con error se deben deshacer los cambios que no se quieren almacenar en la base de datos.

ROLLBACK Deshace los cambios efectuados desde la ultima vez que se realizó COMMIT.

COMMIT Aprueba los cambios de forma permanente en la base de datos.

DECLARE Parte opcional de un bloque anonimo.

SAVEPOINT Puntos de guardado los cuales se puede usar o declarar en mas de una ocasión reemplazando el anterior , se utiliza en los ROLLBACK para retornar a un punto de guardado.

Ejemplo

```
BEGIN


DECLARE

--- SEGUNDA PARTE

        UPDATE BRANDS SET BRAND_NAME= BRAND_NAME||' 2019' WHERE
BRAND_ID=1;

        UPDATE BRANDS SET BRAND_NAME= BRAND_NAME||' 2019' WHERE
BRAND_ID=2;

        COMMIT;

        SAVEPOINT DATOS_MODIFICADOS;

        UPDATE BRANDS SET BRAND_ID=1, BRAND_NAME= BRAND_NAME||' 2021'
WHERE BRAND_ID=14;

        COMMIT;

        SAVEPOINT DATOS_MODIFICADOS;

        UPDATE BRANDS SET BRAND_NAME= BRAND_NAME||' 2020' WHERE
BRAND_ID=12 OR BRAND_ID=13;

        COMMIT;
```

```
        SAVEPOINT DATOS_MODIFICADOS;
/* PRIMER PARTE
        INSERT INTO BRAND VALUES (10, 'BACINI');
        COMMIT;
        INSERT INTO BRAND VALUES (11, 'BAIKA);
        COMMIT;
        INSERT INTO BRAND VALUES (12, 'MONGOOSE');
        COMMIT;
        INSERT INTO BRAND VALUES (9, 'MONGOOSE');
        COMMIT;
*/
        EXCEPTION
                WHEN DUP_VAL_ON_INDEX THEN
                        DBMS_OUTPUT.PUT_LINE('ERROR DE LLAVE PRIMARIA');
                        ROLLBACK TO SAVEPOINT DATOS_MODIFICADOS;
END;
```

Clase #12

Creacion de secuencias

Secuencia Objeto que se utiliza para usar valores secuenciales, los cuales sirven para campos auto incrementables dado que en Oracle no hay datos auto incrementables puede o no ser un valor de llave primaria. Para poder identificarlos se les pone este formato de nombre SQ_NombreTabla. Se necesita tener privilegios para poder crear secuencias.

NEXTVAL Obtiene el siguiente valor de la secuencia. Al usarla por primera vez obtiene el valor de inicio.

CURRVAL Obtiene el valor actual de la secuencia, teniendo que haber utilizado antes el NEXTVAL para poder ejecutar esta función, aplica para cada sesión es decir cada vez que se

haga la conexión a una base de datos se debe haber ejecutado NEXTVAL antes de ejecutar CURVAL.

Ejemplo

--- CON EL USER SYSTEM

GRANT CREATE ANY SEQUENCE, DROP ANY SEQUENCE TO ADMINSITRADORES;

--- EN BASE DE DATOS BD_BICICLETAS

CRETE SEQUENCE SQ_TABLA_CATEGORIAS

START WITH 5

INCREMENT BY 1;

INSERT INTO CATEGORIES VALUES (SQ_TABLA_CATEGORIAS.NEXTVAL, 'BICICLETAS PROFECIONALES');

DECLARE

VALOR_ACTUAL_SQ NUMBER;

BEGIN

VALOR_ACTUAL := SQ_TABLA_CATEGORIAS.CURRVAL;

DBMS_OUTPUT.PUT_LINE('EL VALOR ACTUAL DE LA SECUENCIA ES: '|| VALOR_ACTUAL);

COMMIT;

END;

Clase #13.1

Triggers Disparador automatico que se asocia a una tabla de bases de datos, se ejecuta antes o después de realizar una opreacion INSERT, UPDATE o DELETE. Se pueden usar dos

	OPERACIÓN SQL		
VARIABLE	INSERT	UPDATE	DELETE
:NEW	✓	✓	X
:OLD	X	✓	✓

variables NEW y OLD. El nombre tiene este formato TG_NombreOperacion, no reciben parámetros.

NEW Obtiene los valores nuevos a insertar o a modificar

OLD Obtiene los valores que se eliminaron o se modificaron

Ejemplo

```
--- EN EL USUARIO SYSTEM

GRANT CREATE ANY TRIGGER, ALTER ANY TRIGGER, DROP ANY TRIGGER TO ADMINISTRADORES;

CREATE O REPLACE TRIGGER TG_SQ_TABLA_CATEGORIAS

BEFORE INSERT ON CATEGORIES

FOR EACH ROW

BEGIN

    :NEW .CATEGORY_ID:= SQ_TABLA_CATEGORIAS.NEXTVAL;

END;


DECLARE

BEGIN

    INSERT INTO CATEGORIES (CATEGORY_NAME) VALUES (‘BICICLETA HIBRIDA’);

    INSERT INTO CATEGORIES (CATEGORY_NAME) VALUES (‘BICICLETA PARALELA’);

    COMMIT;

END;
```

Clase #13.2

Triggers

TRIM Quita los espacios en blanco que tenga al inicio y al final una cadena

UPPER Pone en mayúsculas los caracteres de la cadena que se envia como parámetro

Ejemplo

```
CREATE O REPLACE TRIGGER TG_UPDATE_BRANDS
BEFORE UPDATE ON BRANDS
FOR EACH ROW
BEGIN
        :NEW .BRAND_NAME:=TRIM(UPPER(:NEW.BRAND_NAME));
END;
BEGIN
        UPDATE BRANDS SET BRAND_NAME= ‘   th bike  ’ WHERE BRAND_ID=13;
        COMMIT;
END;
--- INSERT INTO BRANDS VALUES (15, ‘ Alpina  ’);
```

Clase #14

Bitácora con Triggers

Realizar un COMMIT dentro del Trigger para lo cual se necesita una transacción autónoma.