UNIVERSIDAD TECNOLÓGICA DE PANAMÁ

FACULTAD DE INGENIERÍA EN SISTEMAS COMPUTACIONALES

LICENCIATURA EN INGENIERIA EN SISTEMAS DE INFORMACION

ASIGNACION 2

CURSO:

SISTEMA BASE DE DATOS 2

FACILITADOR:

ING. HENRY LEZCANO

ESTUDIANTE:

MILAGROS CAMPOS 8-948-227

GRUPO: 1IF131

II SEMESTRE, 2020

Problema 1

Creación de tabla Propia llamada Estudiante

```
SQL>
SQL> Create table Estudiante(
  2      No_estudiante number not null primary key,
  3      Cedula varchar2(15) not null unique,
  4      Nombre varchar2(10) not null,
  5      Apellido varchar2(10) not null,
  6      Cal_final number not null,
  7      Major Varchar2(25) not null
  8  ) ;

Table created.
```

Creación de tabla donde se cargará lo extraído por el cursor

```
SQL>
SQL> Create table Estudiante_Nueva(
  2      No_estudiante2 number not null primary key,
  3      Nombre2 varchar2(10) not null,
  4      Apellido2 varchar2(10) not null
  5  ) ;

Table created.
```

Ejecución del bloque

```
SQL>
SQL> Set serveroutput on;
SQL> DECLARE
  2   --Variable de salida para almacenar los resultados de ls consulta
  3   v_NoEstudiante   Estudiante.No_Estudiante%Type;
  4   v_NombreEstudiante  Estudiante.Nombre%Type;
  5   v_ApellidoEstudiante  Estudiante.Apellido%type;
  6
  7  --valores de acoplamiento usados en la consulta
  8   v_Major Estudiante.major%TYPE := 'Sistemas computacionales';
  9
 10  --declaracion del cursor
 11   CURSOR c_estudiante IS
 12   SELECT No_estudiante, Nombre, Apellido
 13   FROM Estudiante
 14   WHERE major = v_major;
 15
 16   BEGIN
 17   --identificar las filas en el conjunto activo y preparar el procesamiento de datos
 18
 19   OPEN c_Estudiante;
 20
 21   LOOP
 22
 23   --recuperar cada fila del conjutno active y almacenarlo en variables
 24   FETCH c_Estudiante INTO v_NoEstudiante, v_NombreEstudiante, v_ApellidoEstudiante;
 25
 26   --salir cuando no hayan filas
 27   EXIT WHEN c_Estudiante%NOTFOUND;
 28   insert into Estudiante_Nueva  values (v_NoEstudiante, v_nombreEstudiante, v_apellidoEstudiante );
 29   DBMS_OUTPUT.put_line('ID: ' || v_NoEstudiante || 'Nombre: '||v_NombreEstudiante||' Apellido:  ' || v_ApellidoEstudiante );
 30  END LOOP;
 31   CLOSE c_estudiante;
 32   END;
 33   /
ID: 222Nombre: Martina Apellido:  Melendez

PL/SQL procedure successfully completed.
```

Problema 2

Creación de tablas

```
SQL>  CREATE TABLE rooms(
  2   room_id varchar2(15) not null,
  3   building varchar2(20) not null,
  4   primary key (room_id)
  5   );

Table created.

SQL>  COMMIT;

Commit complete.

SQL>  CREATE TABLE Classes (
  2   course_id varchar2(15) not null primary key,
  3   departement varchar2(20) not null,
  4   course varchar2(20) not null,
  5   room_id varchar2(15),
  6   constraint room_id_fk foreign key (room_id) references rooms (room_id)
  7   );

Table created.

SQL>  COMMIT;

Commit complete.

SQL>
```

Inserción de datos a la tabla Rooms

```
SQL>
SQL>  Insert into rooms values ( '2-201', 'Edificio 2' );

1 row created.

SQL>  Insert into rooms values ( '3-301', 'Edificio 3' );

1 row created.

SQL>  Insert into rooms values ( '1-101', 'Edificio 1' );

1 row created.
```

Inserción de datos a la tabla de Classes

```
SQL> Insert into classes values ( 'HIS432', 'HIS', '101', '2-201');

1 row created.

SQL>  Insert into classes values ( 'MAT651', 'MAT', '102','3-301');

1 row created.

SQL>  Insert into classes values ( 'ING723', 'ING', '103', ‘1-101');

1 row created.

SQL>
```

Ejecución del bloque

```
SQL> Set serveroutput on;
SQL> DECLARE
  2  v_RoomID classes.room_id%TYPE;
  3  v_Building rooms.building%TYPE;
  4  v_Department classes.departement%TYPE;
  5  v_Course classes.course%TYPE;
  6  CURSOR c_Building IS
  7  SELECT building
  8  FROM rooms, classes
  9  WHERE rooms.room_id = classes.room_id
 10  AND departement = v_Department
 11  AND course = v_Course;
 12  BEGIN
 13  -- Asignar las variables de Acoplamiento antes de abril el cursor
 14  v_Department := 'HIS';
 15  v_Course := 101;
 16  -- Abril el Cursor
 17  OPEN c_Building;
 18  -- Reasignar las variables de acoplamiento - No tienen efecto alguno, ya que e
 19  LOOP
 20  --recuperar cada fila del conjutno active y almacenarlo en variables
 21  FETCH c_Building INTO v_Building;
 22  --salir cuando no hayan filas
 23  EXIT WHEN c_Building%NOTFOUND;
 24  DBMS_OUTPUT.put_line('Curso dictado en: '|| v_Building);
 25
 26  END LOOP;
 27  v_Department := 'XXX';
 28  v_Course := -1;
 29  END;
 30  /
Curso dictado en: Edificio 2

PL/SQL procedure successfully completed.

SQL>
SQL>
```

Problema 3

Ejecución del bloque

```
SQL>
SQL> set serveroutput on;
SQL> DECLARE
  2  v_RoomID classes.room_id%TYPE;
  3  v_Building rooms.building%TYPE;
  4
  5  CURSOR c_Building( v_Department classes.departement%TYPE,
  6  v_Course classes.course%TYPE) IS
  7  SELECT building
  8  FROM rooms, classes
  9  WHERE rooms.room_id = classes.room_id
 10  AND departement = v_Department
 11  AND course = v_Course;
 12  BEGIN
 13  -- Abril el Cursor
 14  OPEN c_Building ('HIS','101');
 15  -- Reasignar las variables de acoplamiento - No tienen efecto alguno, ya que e
 16  LOOP
 17  --recuperar cada fila del conjutno active y almacenarlo en variables
 18  FETCH c_Building INTO v_Building;
 19  --salir cuando no hayan filas
 20  EXIT WHEN c_Building%NOTFOUND;
 21  DBMS_OUTPUT.put_line('Curso dictado en: '|| v_Building);
 22
 23  END LOOP;
 24
 25  END;
 26  /
Curso dictado en: Edificio 2

PL/SQL procedure successfully completed.

SQL>
```

Problema 4

Creación de tabla Temp_table

```
SQL> create table temp_table (
  2  char_col varchar2(50),
  3  num_col varchar2(50)
  4  );

Table created.

SQL>
```

Ejecución de bloque

```
SQL> set serveroutput on;
SQL> DECLARE
  2  v_RoomData rooms%ROWTYPE;
  3  BEGIN
  4   -- Extraer la información sobre la clase ID -1
  5   SELECT * INTO v_RoomData
  6   FROM rooms
  7   WHERE room_id = '-1';
  8   /* La siguiente orden no se ejecutará nunca, ya que el control pasa inmediatamente al gestor de excepciones */
  9   IF SQL%NOTFOUND THEN
 10   INSERT INTO temp_table ( char_col) VALUES ( 'Not Found');
 11   END IF;
 12  EXCEPTION
 13   WHEN NO_DATA_FOUND THEN
 14   INSERT INTO temp_table ( char_col) VALUES ( 'Not Found, Excpetion Handler');
 15  END;
 16  /

PL/SQL procedure successfully completed.

SQL>
```

```
SQL> select* from temp_table;

CHAR_COL
--------------------------------------------------
NUM_COL
--------------------------------------------------
Not Found, Excpetion Handler


Not Found, Excpetion Handler
```

Problema 5

Problema 5.1

Creación de la tabla Registro estudiante

```
SQL> Create table registro(
  2  department varchar2(20) not null,
  3  course varchar2(20) not null,
  4  numero_estudiante number,
  5  constraint  no_estudiante_fk foreign key (numero_estudiante ) references Estudiante (no_estudiante ),
  6  course_id varchar2(15),
  7  constraint course_id_fk foreign key (course_id) references classes (course_id)
  8  );

Table created.
```

Ejecución del bloque

```
SQL>
SQL> Set serveroutput on;
SQL> DECLARE
  2  /* Declaración de variables para almacenar información acerca de los estudiantes que cursan la especialidad
  3   v_NoEstudiante   Estudiante.No_Estudiante%Type;
  4   v_NombreEstudiante  Estudiante.Nombre%Type;
  5   v_ApellidoEstudiante  Estudiante.Apellido%type;
  6
  7  -- Cursor para recuperar la informacion sobre los estudiantes de Historia
  8  CURSOR c_HistoryStudents IS
  9  SELECT no_estudiante, Nombre, Apellido
 10  FROM Estudiante
 11  WHERE major = 'Sistemas Computacionales';
 12  BEGIN
 13  -- Abre el cursor e inicializa el conjunto activo
 14  OPEN c_HistoryStudents;
 15  LOOP
 16  -- Recupera la información del siguiente estudiante
 17  FETCH c_HistoryStudents INTO v_NoEstudiante, v_NombreEstudiante, v_ApellidoEstudiante;
 18  -- Salida del bucle cuando no hay más filas por recuperar
 19  EXIT WHEN c_HistoryStudents%NOTFOUND ;
 20  /* Procesa las filas recuperadas. En este caso matricula a cada estudiante en Historia 301, insertándolo en
 21  Registra también el nombre y el apellido en la tabla temp_table */
 22  INSERT INTO registro ( Numero_estudiante, department, course)
 23  VALUES ( v_NoEstudiante, 'HIS', '101');
 24  INSERT INTO temp_table ( num_col, char_col)
 25  VALUES ( v_NoEstudiante, v_NombreEstudiante|| ' '|| v_ApellidoEstudiante);
 26  END LOOP;
 27  -- Libera los recursos utilizados por el curso
 28  CLOSE c_HistoryStudents;
 29  -- Confirmamos el trabajo
 30  COMMIT;
 31  END;
 32  /

PL/SQL procedure successfully completed.
```

Resultados

```
SQL> select *from temp_table;

CHAR_COL
--------------------------------------------------
NUM_COL
--------------------------------------------------
Not Found, Excpetion Handler


Not Found, Excpetion Handler


Martina Melendez
222


SQL> select * from registro;

DEPARTMENT           COURSE               NUMERO_ESTUDIANTE COURSE_ID
-------------------- -------------------- ----------------- ---------------
HIS                  101                                222

SQL>
```

Problema 5.2

Ejecución del bloque

```
SQL> Set serveroutput on;
SQL> DECLARE
  2   /* Declaración de variables para almacenar información acerca de los estudiantes que cursan la especialidad de Historia */
  3  v_NoEstudiante   Estudiante.No_Estudiante%Type;
  4   v_NombreEstudiante  Estudiante.Nombre%Type;
  5   v_ApellidoEstudiante  Estudiante.Apellido%type;
  6
  7  -- Cursor para recuperar la información sobre los estudiantes de Historia
  8   CURSOR c_HistoryStudents IS
  9   SELECT No_estudiante, Nombre, Apellido
 10   FROM Estudiante
 11   WHERE major = 'Ciencias computacionales';
 12  BEGIN
 13   -- Abre el cursor e inicializa el conjunto activo
 14   OPEN c_HistoryStudents;
 15   LOOP
 16   -- Recupera la información del siguiente estudiante
 17   FETCH c_HistoryStudents INTO v_NoEstudiante, v_NombreEstudiante, v_ApellidoEstudiante ;
 18   /* Procesa las filas recuperadas. En este caso matrícula a cada estudiante en Historia 301, insertándolo en la tabla Registro.
 19  Registra también el nombre y el apellido en la tabla temp_table */
 20   INSERT INTO Registro( numero_estudiante, department, course)
 21   VALUES ( v_NoEstudiante, 'HIS', 101);
 22   INSERT INTO temp_table ( num_col, char_col)
 23  VALUES ( v_NoEstudiante, v_NombreEstudiante|| ' '|| v_ApellidoEstudiante);
 24  -- Salida del bucle cuando no hay más filas por recuperar
 25   EXIT WHEN c_HistoryStudents%NOTFOUND ;
 26   END LOOP;
 27   -- Libera los recursos utilizados por el curso
 28   CLOSE c_HistoryStudents;
 29  -- Confirmamos el trabajo
 30  COMMIT;
 31  END;
 32  /
```

Resultados

```
SQL> select* from temp_table;

CHAR_COL
--------------------------------------------------
NUM_COL
--------------------------------------------------
Not Found, Excpetion Handler


Not Found, Excpetion Handler


Martina Melendez
222


CHAR_COL
--------------------------------------------------
NUM_COL
--------------------------------------------------




SQL> select* from registro;

DEPARTMENT           COURSE               NUMERO_ESTUDIANTE COURSE_ID
-------------------- -------------------- ----------------- --------------
HIS                  101                                222
HIS                  101

SQL>
```

Problema 5.3

Ejecución del bloque y resultados

Símbolo del sistema - sqlplus sys as sysdba

```
SQL> Set serveroutput on;
SQL> DECLARE
  2   -- Cursor para recuperar la información sobre los estudiantes de Historia
  3   CURSOR c_HistoryStudents IS
  4   SELECT No_estudiante, Nombre, Apellido
  5   FROM Estudiante
  6   WHERE major = 'Sistemas computacionales';
  7   -- Declaración el registro para almacenar información extraída
  8   v_StudentData c_HistoryStudents%ROWTYPE;
  9  BEGIN
 10   -- Abre el cursor e inicializa el conjunto activo
 11   OPEN c_HistoryStudents;
 12   -- Recupera la información del siguiente estudiante
 13   FETCH c_HistoryStudents INTO v_StudentData;
 14   -- El bucle continua mientras haya mas filas que extraer
 15   WHILE c_HistoryStudents%FOUND LOOP
 16   /* Procesa las filas recuperadas. En este caso matricula a cada estudiante en Historia 301, insertándolo en la tabla
 17  registered_students. Registra también el nombre y el apellido en la tabla temp_table */
 18   INSERT INTO Registro ( numero_estudiante, department, course)
 19   VALUES ( v_StudentData.No_Estudiante, 'HIS', 101);
 20   INSERT INTO temp_table ( num_col, char_col)
 21   VALUES ( v_StudentData.No_Estudiante, v_StudentData.Nombre || ' '|| v_StudentData.Apellido);
 22   -- Recuperar la fila siguiente. La condición %FOUND se comprobara antes de que el bucle continúe
 23   FETCH c_HistoryStudents INTO v_StudentData;
 24   END LOOP;
 25   -- Libera los recursos utilizados por el curso
 26   CLOSE c_HistoryStudents;
 27  -- Confirmamos el trabajo
 28  COMMIT;
 29  END;
 30  /

PL/SQL procedure successfully completed.

SQL> select * from registro;

DEPARTMENT           COURSE               NUMERO_ESTUDIANTE COURSE_ID
-------------------- -------------------- ----------------- --------------
HIS                  101                                222
HIS                  101
HIS                  101                                222

SQL>
```

Problema 5.4

Ejecución del bloque

```
SQL> Set serveroutput on;
SQL> DECLARE
  2   -- Cursor para recuperar la información sobre los estudiantes de Historia
  3   CURSOR c_HistoryStudents IS
  4  SELECT No_estudiante, Nombre, Apellido
  5   FROM Estudiante
  6  WHERE major = 'Sistemas computacionales';
  7  BEGIN
  8   /* Inicio del bucle. Aquí se ejecuta una orden OPEN
  9   implícita sobre c_HistoryStudents */
 10   FOR v_StudentData IN c_HistoryStudents LOOP
 11   -- Aquí se ejecuta una orden FETCH implícita
 12   /* Procesa las filas recuperadas. En este caso matricula a cada estudiante en Historia 301, insertándolo en la tabla
 13  registered_students. Registra también el nombre y el apellido en la tabla temp_table */
 14  INSERT INTO Registro ( numero_estudiante, department, course)
 15   VALUES ( v_StudentData.No_Estudiante, 'HIS',101);
 16   INSERT INTO temp_table ( num_col, char_col)
 17   VALUES ( v_StudentData.No_Estudiante, v_StudentData.Nombre || ' '|| v_StudentData.Apellido);
 18   -- Antes de continuar con el bucle, aquí se hace una comprobacion implícita de c_HistoryStudents %NOTFOUND.
 19   END LOOP;
 20  -- Ahora el bucle ha terminado, se hace cierre implícito del cursor c_HistoryStudents
 21  -- Confirmamos el trabajo
 22  COMMIT;
 23  END;
 24  /

PL/SQL procedure successfully completed.
```

Resultados

```
SQL> select* from temp_table;

CHAR_COL
--------------------------------------------------
NUM_COL
--------------------------------------------------
Not Found, Excpetion Handler


Not Found, Excpetion Handler


Martina Melendez
222


CHAR_COL
--------------------------------------------------
NUM_COL
--------------------------------------------------



Martina Melendez
222

Martina Melendez
222


6 rows selected.

SQL> select* from Registro;

DEPARTMENT           COURSE               NUMERO_ESTUDIANTE COURSE_ID
-------------------- -------------------- ----------------- ---------------
HIS                  101                                222
HIS                  101
HIS                  101                                222
HIS                  101                                222

SQL>
```

Problema 6

Creación de tabla propia

```
Connected.
SQL> Create table Juguetes (
  2     Id_juguete varchar2(8) not null primary key,
  3      Nombre_juguete  varchar2(30) not null,
  4      Descripcion varchar2(200),
  5     Precio number not null
  6  ) ;

Table created.
```

Inserción de tuplas

```
SQL> insert into  juguetes values (‘100’, 'Barbie mariposa', ‘Cambia de color su cabello en agua y sus alas se mueven’, 15);

1 row created.

SQL> insert into  juguetes values (‘101’, 'Piscina', ‘para 6 personas ’, 50);

1 row created.

SQL> insert into  juguetes values (‘102’, 'Bicicleta', ‘ ‘, 65);

1 row created.

SQL>
```

Tabla donde se insertara

```
SQL>
SQL> Create table Juguete_nuevo (
  2       Nombre_juguete22 varchar2(30) not null,
  3      Descripcion varchar2(200),
  4     Precio2 number not null
  5  ) ;

Table created.
```

Ejecución de bloque

```
SQL> set serveroutput on;
SQL> DECLARE
  2      v_nombre Juguetes.nombre_juguete%TYPE;
  3      v_precio   Juguetes.precio%TYPE;
  4      v_descripcion Juguetes.descripcion%TYPE;
  5
  6      v_rebajas number := 30;
  7      CURSOR c_rebajas IS
  8          SELECT nombre_juguete, precio, descripcion
  9          FROM juguetes
 10          WHERE precio >= v_rebajas;
 11  BEGIN
 12      OPEN c_rebajas;
 13      UPDATE Juguetes
 14              set precio = precio - (precio*0.2)
 15              where precio >= 30;
 16  Insert into Juguete_nuevo values (v_nombre, v_precio, v_descripcion);
 17      LOOP
 18          FETCH c_rebajas INTO v_nombre, v_precio, v_descripcion;
 19          EXIT WHEN c_rebajas%NOTFOUND;
 20      END LOOP;
 21      CLOSE c_rebajas;
 22      COMMIT;
 23  END;
 24  /
DECLARE
```