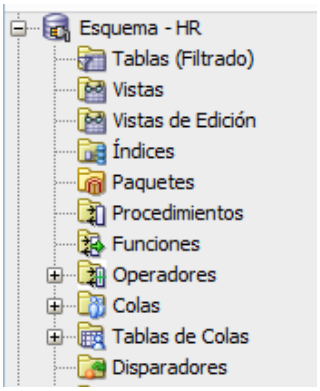
Afuera Centro de Estudios Superiores	**Prueba de Evaluación Continua**	

Ciclo Formativo GRADO SUPERIOR:	**DESARROLLO DE APLICACIONES MULTIPLATAFORMA**	**IFCS02**
Módulo Profesional Clave: **01**	**BASES DE DATOS**	
Nº de Expediente:	Nif:	Fecha: 20/5/2022
Nombre y Apellidos: Daniel Izquierdo Bonilla		

- Este examen se entrega en un único fichero de texto (word/odt) en el aula virtual **a partir de esta plantilla**. Durante este examen no está permitido el uso de internet en el PC (excepto para la descarga de los scripts y la entrega del doc) ni el uso de móviles (deben estar bien guardados) u otro tipo de dispositivos.
- Se usará el PC solo para ejecutar SQL-DEVELOPER. Se permite también el uso de un prontuario en papel (1 folio).
- Se debe guardar absoluto silencio y no está permitida ninguna interacción con el resto de alumnos. Si tienes alguna duda o necesidad, levanta la mano y el profesor te atenderá en cuanto pueda.
- Lee con detenimiento y atención este documento y las preguntas, si hace falta varias veces. Tómate tu tiempo (hay suficiente y de sobra para completarlo) y completa las preguntas en esta misma plantilla en formato electrónico.
- Pon el nombre en la cabecera del documento y nombra el documento final como APELLIDO_nombre.doc

PREPARACIÓN DEL ENTORNO:

Trabajaremos con el esquema HR, Se encuentran en el aula virtual los ficheros de script **HR_0.dropFP.sql**, **HR_1.drop.sql**, **HR_2.cre.sql, HR_3.popul.sql** y **HR_4.idx.sql** de borrado, creación de tablas, población de datos e índices sobre ese esquema HR.
IMPORTANTE: tras ejecutar las sentencias de DROP de los dos primeros ficheros **se debe comprobar que se han borrado cualquier otra tabla, vista, función, procedimiento o trigger previamente existente** de tal manera que no quede ninguno previo a la creación de tablas y población: (Pueden borrarse con el SQLDEVELOPER usando el botón derecho), debiendo quedar así el entorno;

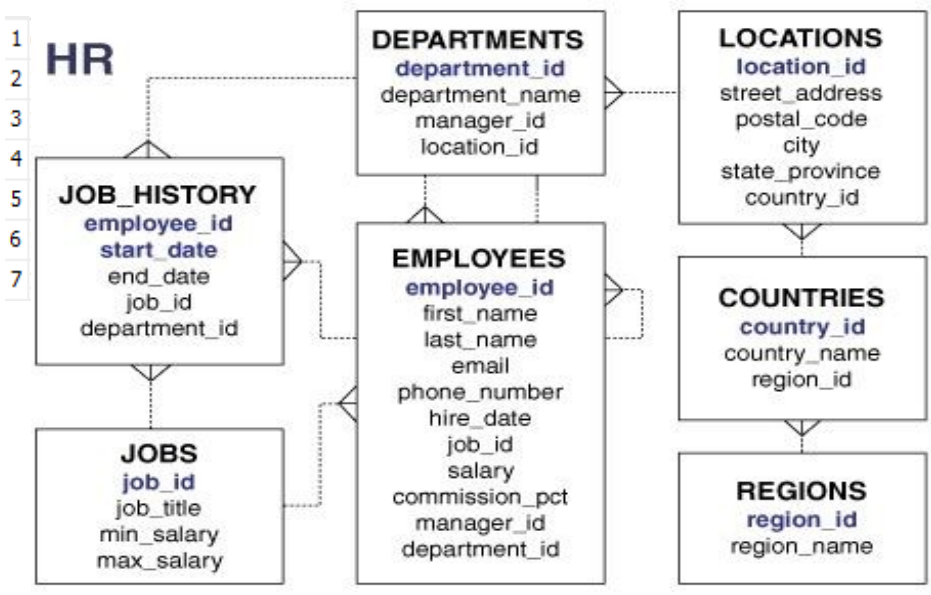


El profesor comprobará que todo queda borrado y podrá proseguirse.

Una vez borrados ejecutaremos los scripts **HR_2.cre.sql** y HR**_3.popul.sql** de creación de tablas población respectivamente y finalmente **HR_4.idx.sql** para los índices. Para comprobar que la carga ha sido correcta podemos ejecutar la siguiente sentencia que nos dará el número de filas de cada tabla:

SELECT TABLE_NAME, num_rows from user_tables order by table_name;

Puede comprobarse si coincide con la lista de cuenta de registros. Como ayuda, se muestra el modelo de HR:



TAREAS: (en cada parte se anexará **texto con el código** y **captura con resultados de ejemplo**. Es importante incluir en el texto incluir un comentario con el nombre del autor. También en la captura que se vea claramente que el programa ha compilado correctamente.

---

1. **BLOQUE ANÓNIMO SENCILLO** (1 punto)
   Hacer un bloque anónimo que devuelva el nombre de la ciudad con más empleados

```
-- Ej1
-- Daniel Izquierdo Bonilla
set serveroutput on
declare
    ciudad LOCATIONS.CITY%type;
    cuenta INTEGER;
begin
    select city, count(EMPLOYEE_ID) as num_emp into ciudad,cuenta
    from LOCATIONS
             join DEPARTMENTS on LOCATIONS.LOCATION_ID = DEPARTMENTS.LOCATION_ID
             join EMPLOYEES on DEPARTMENTS.DEPARTMENT_ID = EMPLOYEES.DEPARTMENT_ID
    group by city
    order by num_emp desc fetch first 1 rows only;


    dbms_output.put_line('Ciudad: ' || ciudad || ' | Empleados: '||cuenta);
end;
```

```
        select city, count(EMPLOYEE_ID) as num_emp into ciudad,cuenta
        from LOCATIONS
                 join DEPARTMENTS on LOCATIONS.LOCATION_ID = DEPARTMENTS.LOCATION_ID
                 join EMPLOYEES on DEPARTMENTS.DEPARTMENT_ID = EMPLOYEES.DEPARTMENT_ID
        group by city
        order by num_emp desc fetch first 1 rows only;


        dbms_output.put_line('Ciudad: ' || ciudad || ' | Empleados: '||cuenta);
    end;
[2022-05-20 13:52:25] completed in 15 ms
Ciudad: South San Francisco | Empleados: 45
```

2. **TRATAMIENTO DE EXCEPCIONES** (2 puntos)
   Hacer un bloque anónimo que pida un id de empleado y devuelva su nombre, apellidos, salario e id del departamento. Debe gestionar las siguientes excepciones:
   a) Que no exista el empleado con ese id (interna)
   b) Que no tenga departamento (definida) → debe dar aviso pero mostrar datos
   c) Que su salario sea más de 10.000 (definida) → debe dar aviso pero NO mostrar datos
   d) Otras (OTHERS)

```
-- Ej2
-- Daniel Izquierdo Bonilla
set serveroutput on
declare
    idEmp   employees.EMPLOYEE_ID%type;
    cuenta  integer;
    ape     EMPLOYEES.LAST_NAME%type;
    nom     employees.first_name%type;
    salario EMPLOYEES.salary%type;
    dep     DEPARTMENTS.DEPARTMENT_NAME%type;
    numDep exception;
    salary10000 exception;
begin
    idEmp := &dime_id_empleado;

    select SALARY into cuenta from employees where EMPLOYEE_ID = idEmp;

    if cuenta > 10000
    then
        raise salary10000;
    end if;

    select count(DEPARTMENT_ID) into cuenta from EMPLOYEES where EMPLOYEE_ID = idEmp;

    if cuenta = 0
    then
        raise numDep;
    end if;

    select employees.first_name, LAST_NAME, SALARY, DEPARTMENT_NAME
    into nom, ape, salario, dep
    from employees
             join DEPARTMENTS on DEPARTMENTS.DEPARTMENT_ID = EMPLOYEES.DEPARTMENT_ID
    where EMPLOYEE_ID = idEmp;



    dbms_output.put_line('Nombre: ' || nom || ' Apellido: ' || ape || ' Salario: ' ||
salario || ' Departamento: ' ||
                         dep);

exception
    when NO_DATA_FOUND then DBMS_OUTPUT.PUT_LINE('Empleado no econtrado');
    when salary10000 then DBMS_OUTPUT.PUT_LINE('El salario del empleado con ID
introducida es superior a 10000');
    when numDep then dbms_output.put_line('El empleado con la ID introducida no tiene
departamento');
    when others then dbms_output.put_line('Error inesperado');
end;
```

```
        dbms_output.put_line('Nombre: ' || nom || ' Apellido: ' || ape || ' Salario: ' || salario || ' Departamento: ' ||
                             dep);

    exception
        when NO_DATA_FOUND then DBMS_OUTPUT.PUT_LINE('Empleado no econtrado');
        when salary10000 then DBMS_OUTPUT.PUT_LINE('El salario del empleado con ID introducida es superior a 10000');
        when numDep then dbms_output.put_line('El empleado con la ID introducida no tiene departamento');
        when others then dbms_output.put_line('Error inesperado');
    end;
[2022-05-20 13:50:44] completed in 16 ms
Nombre: John Apellido: Chen Salario: 8200 Departamento: Finance
```

3. **CURSORES** (2 puntos)

   Hacer un bloque anónimo que muestre el nombre, ciudad y código postal de los departamentos cuyo país se pide por teclado. El cursor debe declararse, abrirse, usar FETCH y cerrarse (es decir, no se permite FOR en este caso)

```
-- Ej3
-- Daniel Izquierdo Bonilla
set serveroutput on
declare
    pais      COUNTRIES.COUNTRY_NAME%type;
    nomDep    DEPARTMENTS.DEPARTMENT_NAME%type;
    ciudad    LOCATIONS.CITY%type;
    codPostal LOCATIONS.POSTAL_CODE%type;
    cursor c_ej is select DEPARTMENTS.DEPARTMENT_NAME, LOCATIONS.CITY,
LOCATIONS.POSTAL_CODE
                   from DEPARTMENTS
                            left join LOCATIONS on DEPARTMENTS.LOCATION_ID =
LOCATIONS.LOCATION_ID
                            left join COUNTRIES on LOCATIONS.COUNTRY_ID =
COUNTRIES.COUNTRY_ID
                   where COUNTRY_NAME = pais;
begin
    pais := '&dime_pais';

    open c_ej;
    fetch c_ej into nomDep, ciudad, codPostal;
    while c_ej%found
        loop
            dbms_output.put_line('Departamento: ' || nomDep || ' Ciudad: ' || ciudad ||
' Codigo Postal: ' || codPostal);
            fetch c_ej into nomDep, ciudad, codPostal;
        end loop;
    close c_ej;
end;
```

```
        open c_ej;
        fetch c_ej into nomDep, ciudad, codPostal;
        while c_ej%found
            loop
                dbms_output.put_line('Departamento: ' || nomDep || ' Ciudad: ' || ciudad || ' Codigo Postal: ' || codPostal);
                fetch c_ej into nomDep, ciudad, codPostal;
            end loop;
        close c_ej;
    end;
[2022-05-20 13:49:30] completed in 16 ms
Departamento: Human Resources Ciudad: London Codigo Postal:
Departamento: Sales Ciudad: Oxford Codigo Postal: OX9 9ZB
```

4. **FUNCIONES** (1 punto) (no usa la BD HR)
   Hacer una función llamada TRIANGULO() con tres argumentos enteros que devuelva un VARCHAR2 que diga si es equilátero (tres lados iguales) isósceles (solo 2) o escaleno (los tres distintos) Debe retornar 'ERROR' si cualquiera de los lados es mayor o igual que la suma de los otros dos.

```
-- Ej4
-- Daniel Izquierdo Bonilla
create or replace function TRIANGULO(lado1 INTEGER, lado2 INTEGER, lado3 INTEGER) return
varchar2
    is
    lado_invalido exception;
begin
    if (lado1 + lado2) < lado3 or (lado3 + lado2) < lado1 or (lado1 + lado3) < lado2
then
        raise lado_invalido;
    end if;

    if lado1 = lado2 and lado1 = lado3 then
        return 'Es Equilatero';
    elsif lado1 = lado2 or lado1 = lado3 or lado2 = lado3 then
        return 'Es Isosceles';
    end if;

    return 'Es Escaleno';
exception
    when lado_invalido then return 'ERROR';
end;
```

```
[2022-05-20 14:01:56] completed in 0 ms
Es Isosceles
HR> begin
        DBMS_OUTPUT.PUT_LINE(TRIANGULO(3,6,4));
    end;
[2022-05-20 14:02:01] completed in 0 ms
Es Escaleno
HR> begin
        DBMS_OUTPUT.PUT_LINE(TRIANGULO(11,6,4));
    end;
[2022-05-20 14:05:35] completed in 0 ms
ERROR
```

5. **PROCEDIMIENTOS** (2 puntos)

   Hacer un procedimiento LISTA_DEPARTAMENTOS (ciudad) que liste el nombre y el número de empleados de los departamentos que se ubican en una determinada ciudad, que será su parámetro. (Si se quiere puede emplearse un cursor FOR en este caso) (Se recomienda pensar y probar la SELECT previamente y luego hacer el procedimiento)

```
-- Ej5
-- Daniel Izquierdo Bonilla
CREATE OR REPLACE PROCEDURE LISTA_EMPLEADOS(ciudad LOCATIONS.CITY%TYPE) IS
BEGIN
    FOR registro IN (SELECT DEPARTMENTS.DEPARTMENT_NAME AS nom,
COUNT(EMPLOYEES.EMPLOYEE_ID) AS emp
         FROM DEPARTMENTS
                  LEFT JOIN EMPLOYEES ON DEPARTMENTS.DEPARTMENT_ID =
EMPLOYEES.DEPARTMENT_ID
                  JOIN LOCATIONS ON DEPARTMENTS.LOCATION_ID = LOCATIONS.LOCATION_ID
         WHERE LOCATIONS.CITY = ciudad
         GROUP BY DEPARTMENTS.DEPARTMENT_NAME)
        LOOP
            DBMS_OUTPUT.PUT_LINE('Departamento: ' || registro.nom || ' - Empleados: ' ||
registro.emp);
        END LOOP;
END LISTA_EMPLEADOS;
```

```
    BEGIN
        FOR registro IN (SELECT DEPARTMENTS.DEPARTMENT_NAME AS nom, COUNT(EMPLOYEES.EMPLOYEE_ID) AS emp
             FROM DEPARTMENTS
                      LEFT JOIN EMPLOYEES ON DEPARTMENTS.DEPARTMENT_ID = EMPLOYEES.DEPARTMENT_ID
                      JOIN LOCATIONS ON DEPARTMENTS.LOCATION_ID = LOCATIONS.LOCATION_ID
             WHERE LOCATIONS.CITY = ciudad
             GROUP BY DEPARTMENTS.DEPARTMENT_NAME)
            LOOP
                DBMS_OUTPUT.PUT_LINE('Departamento: ' || registro.nom || ' - Empleados: ' || registro.emp);
            END LOOP;
    END LISTA_EMPLEADOS;
[2022-05-20 13:53:02] completed in 54 ms
```

```
HR> begin
        LISTA_EMPLEADOS('Oxford');
    end;
[2022-05-20 13:53:19] completed in 16 ms
Departamento: Sales - Empleados: 33
```

6. **TRIGGERS** (2 puntos)

   Antes de hacer el trigger crear con DDL una tabla REGISTRO con los siguientes campos: NOMBRE, APELLIDO, TIPO_CAMBIO, ANTIGUO, NUEVO (Todos VARCHAR)

   Hacer un trigger que ante cualquier cambio de salario o email en la tabla empleados inserte un registro en la tabla REGISTRO con los valores correspondientes. También debe insertar un registro

en el caso de que se inserte un nuevo empleado o que se elimine uno. TIPO_EVENTO será “Cambio salario”, “Cambio email”, “Alta empleado” o “Baja empleado” en cada caso. ANTIGUO y NUEVO serán los valores antiguos y nuevos de los cambios (en caso de inserción o borrado de un empleado, no se rellenará)

(En este ejercicio debe anexarse además del código del trigger, la captura del select * de la tabla CONTROL después de hacer un update, insert o delete en EMPLOYEES)

```
-- Ej6
-- Daniel Izquierdo Bonilla
CREATE TABLE REGISTRO
(
    NOMBRE      VARCHAR2(100),
    APELLIDO    VARCHAR2(100),
    TIPO_CAMBIO VARCHAR2(100),
    ANTIGUO     VARCHAR2(100),
    NUEVO       VARCHAR2(100)
);

CREATE OR REPLACE TRIGGER CONTROL_CAMBIOS
    BEFORE UPDATE OR INSERT OR DELETE
    ON EMPLOYEES
    FOR EACH ROW
BEGIN
    IF INSERTING THEN
        INSERT INTO REGISTRO (NOMBRE, APELLIDO, TIPO_CAMBIO) VALUES (:new.FIRST_NAME,
:new.LAST_NAME, 'Alta empleado');
    ELSIF UPDATING ('SALARY') THEN
        INSERT INTO REGISTRO VALUES (:new.FIRST_NAME, :new.LAST_NAME, 'Cambio salario',
:old.SALARY, :new.SALARY);
    ELSIF UPDATING ('EMAIL') THEN
        INSERT INTO REGISTRO
        VALUES (:new.FIRST_NAME, :new.LAST_NAME, 'Cambio email', :old.EMAIL,
:new.EMAIL);
    ELSIF DELETING THEN
        INSERT INTO REGISTRO (NOMBRE, APELLIDO, TIPO_CAMBIO) VALUES (:old.FIRST_NAME,
:old.LAST_NAME, 'Baja empleado');
    END IF;
END;
```

```
        IF INSERTING THEN
            INSERT INTO REGISTRO (NOMBRE, APELLIDO, TIPO_CAMBIO) VALUES (:new.FIRST_NAME, :new.LAST_NAME, 'Alta empleado');
        ELSIF DELETING THEN
            INSERT INTO REGISTRO (NOMBRE, APELLIDO, TIPO_CAMBIO) VALUES (:old.FIRST_NAME, :old.LAST_NAME, 'Baja empleado');
        ELSIF UPDATING ('SALARY') THEN
            INSERT INTO REGISTRO VALUES (:new.FIRST_NAME, :new.LAST_NAME, 'Cambio trabajo', :old.JOB_ID, :new.JOB_ID);
        ELSIF UPDATING ('EMAIL') THEN
            INSERT INTO REGISTRO
            VALUES (:new.FIRST_NAME, :new.LAST_NAME, 'Cambio departamento', :old.DEPARTMENT_ID, :new.DEPARTMENT_ID);
        END IF;
    END;
[2022-05-20 13:31:54] completed in 16 ms
```

	NOMBRE	APELLIDO	TIPO_CAMBIO	ANTIGUO	NUEVO
1	William	Smith	Baja empleado	<null>	<null>
2	Trenna	Rajs	Cambio salario	5000	5500
3	William	Smith	Baja empleado	<null>	<null>

NOTA1: Recordad que un código que tenga errores de compilación no puntuará.
NOTA2: Este control de simulación tiene un ejercicio menos que el examen de 3ª EV.
NOTA3: Está terminantemente prohibido mirar los trabajos de los compañeros.