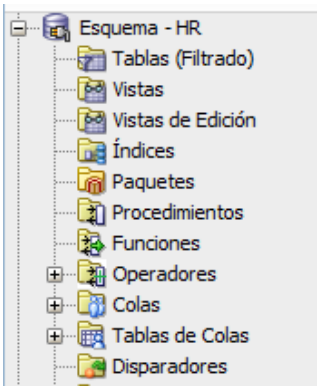
Afuera Centro de Estudios Superiores	**Prueba de Evaluación Continua**	

Ciclo Formativo GRADO SUPERIOR:	**DESARROLLO DE APLICACIONES MULTIPLATAFORMA**	**IFCS02**
Módulo Profesional Clave: **01**	**BASES DE DATOS**	
Nº de Expediente:	Nif:	Fecha: 13/5/2022
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;

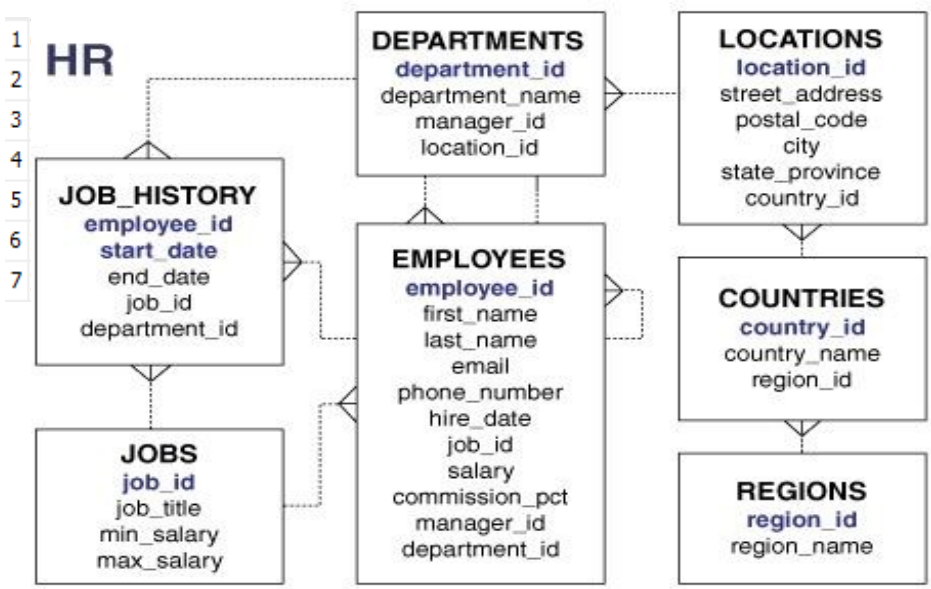


El profesor comprobará que todo queda borrado y podrá proseguirse.

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 apellido del empleado de mayor antigüedad

```
Ej1
-- Daniel Izquierdo Bonilla
set serveroutput on
declare
    ape employees.last_name%type;
begin
    select last_name into ape from employees order by hire_date fetch first 1 rows only;

    dbms_output.put_line( A: 'Apellido: ' || ape);
end;
```

2. **TRATAMIENTO DE EXCEPCIONES** (2 puntos)

   Hacer un bloque anónimo que pida un id de departamento y devuelva su nombre, el nombre de su jefe (manager) y su número de empleados. Debe gestionar las siguientes excepciones:

   a) Que no exista departamento con ese id (interna)
   b) Que no tenga empleados (definida) → debe dar aviso pero mostrar datos
   c) Otras (OTHERS)

```
-- Ej2
-- Daniel Izquierdo Bonilla
set serveroutput on
declare
    idPedido employees.department_id%type;
    cuenta   integer;
    nomDep   departments.department_name%type;
    nomMan   employees.first_name%type;
    numDep exception;
begin
    idPedido := &dime_id_pedido;

    select count(*) into cuenta from departments where department_id = idPedido;

    if cuenta is null or cuenta = 0
    then
        raise numDep;
    end if;

    select count(*) into cuenta from employees where department_id = idPedido;

    if cuenta is null or cuenta = 0
    then
        dbms_output.put_line( A: 'El departamento no tiene empleados');
    end if;

    select employees.first_name, departments.department_name
    into nomMan, nomDep
    from employees
             join departments on departments.department_id = employees.department_id
    where employees.employee_id = (select manager_id from departments where departments.department_id = idPedido);

    dbms_output.put_line( A: 'Departamento: ' || nomDep || ' || Manager: ' || nomMan || ' || N° empleados: ' || cuenta);

exception
    when numDep then dbms_output.put_line( A: 'No existe empleado con la ID introducida');
    when others then dbms_output.put_line( A: 'Error inesperado');
end;
```

3. **CURSORES** (2 puntos)

Hacer un bloque anónimo que muestre el nombre, apellido y puesto de trabajo (job_title) de los empleados cuyo EMAIL comience con una determinada letra (que 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
    letra char;
    nom   employees.first_name%type;
    ape   employees.last_name%type;
    job   jobs.job_title%type;
    mails employees.email%type;
    cursor c_ej is select first_name, last_name, job_title, email
                   from employees
                            join jobs on employees.job_id = jobs.job_id
                   where substr(email, 1, 1) = letra;
begin
    letra := '&dime_letra';
    letra := UPPER(letra);

    open c_ej;
    fetch c_ej into nom, ape, job, mails;
    while c_ej%found
        loop
            dbms_output.put_line( A: 'Nombre: ' || nom || ' Apellido: ' || ape || ' Trabajo: ' || job ||
                                  ' Email: ' || mails);
            fetch c_ej into nom, ape, job, mails;
        end loop;
    close c_ej;
end;
```

4. **FUNCIONES** (1 punto) (no usa la BD HR)

Hacer una función llamada CUBO que devuelva el cubo de un numero real r, por ejemplo CUBO(4) debe retornar 256, CUBO(2,5) debe retornar 15.625, etc. Debe retornar un error (excepción) si el argumento es mayor que 1000.

```
-- Ej4
-- Daniel Izquierdo Bonilla
create or replace function CUBO(numero float) return varchar2
    is
    num_max exception;
begin
    if numero > 1000 then
        raise num_max;
    end if;
    return numero * numero * numero;

exception
    when num_max then return 'Se ha producido un error, el numero no puede ser mayor de 1000';
end;
```

5. **PROCEDIMIENTOS** (2 puntos)

Hacer un procedimiento LISTA_EMPLEADOS (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
    ciudad2 locations.city%type;
    numEmp  integer;
    nom     employees.first_name%type;
    cursor c_ej is select DEPARTMENTS.DEPARTMENT_NAME, count(EMPLOYEES.FIRST_NAME) as empleados
                   from DEPARTMENTS
                            left join EMPLOYEES on DEPARTMENTS.department_id = departments.department_id
                            join locations on departments.location_id = locations.location_id
                   where city = 'Oxford'
                   group by DEPARTMENT_NAME;
begin
    open c_ej;
    fetch c_ej into nom, numEmp;
    while c_ej%found
        loop
            dbms_output.put_line( A: 'Nombre: ' || nom || ' Empleados: ' || numEmp);
            fetch c_ej into nom, numEmp;
        end loop;
    close c_ej;
end;
```

6. **TRIGGERS** (2 puntos)

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

Hacer un trigger que ante cualquier cambio de puesto de trabajo (JOB_ID) o departamento en la tabla empleados inserte un registro en la tabla CONTROL con los valores correspondientes. También debe insertar un registro en el caso de que se inserte un nuevo empleado. TIPO_EVENTO será “Cambio trabajo”, “Cambio departamento” o “Nuevo empleado” en cada caso. ANTIGUO y NUEVO serán los valores antiguos y nuevos de los cambios (en caso de inserción de nuevo empleado, no se rellenarán)

(En este ejercicio debe anexarse además del código del trigger, la captura del select * de la tabla CONTROL después de cambiar un puesto de trabajo, un departamento e insertar un nuevo empleado)

```
create table CONTROL (
    nombre VARCHAR2(100),
    apellido VARCHAR2(100),
    tipo_cambio VARCHAR2(100),
    antiguo VARCHAR2(100) null,
    nuevo VARCHAR2(100) null
);
```

```
CREATE OR REPLACE TRIGGER audit_cambios
    BEFORE INSERT OR UPDATE
    ON employees
    FOR EACH ROW
BEGIN
    if inserting then
        insert INTO CONTROL
        VALUES (FIRST_NAME,LAST_NAME,'Nuevo Empleado',null,null);
    else if updating('job_id') then
        insert INTO CONTROL
        VALUES (FIRST_NAME,LAST_NAME,'Cambio trabajo',:old.JOB_ID,:new.JOB_ID);
    else if updating('department_id') then
        insert INTO CONTROL
        VALUES (FIRST_NAME,LAST_NAME,'Cambio departamento',:old.DEPARTMENT_ID,:new.DEPARTMENT_ID);
    end if;
END;
```

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.