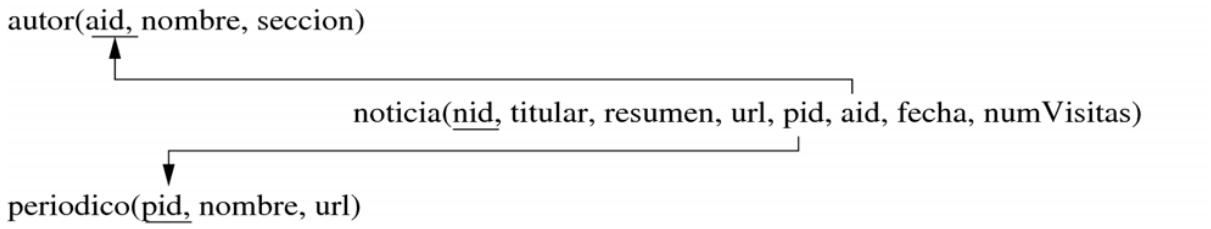
Ejercicio 3. (6 puntos) Dado el siguiente modelo relacional de una base de datos que almacena información de noticias de periódicos digitales:



Contesta a los siguientes apartados:

a. (0,5 puntos) Escribe una consulta en lenguaje SQL que muestre el nombre de los autores de noticias publicadas el 1/06/2018 con más de 1.000 visitas, así como el nombre de los periódicos en los que han publicado las noticias. Debes evitar que aparezcan resultados repetidos.

```
SELECT DISTINCT a.nombre, p.nombre
FROM autor a JOIN noticia n ON a.aid = n.aid
        JOIN periodico p ON n.pid = p.pid
WHERE n.fecha = TO_DATE('01/06/2018','DD/MM/YYYY')
    AND numVisitas > 1000;
```

b. (0,5 puntos) Escribe una consulta en lenguaje SQL que muestre, para cada autor que haya publicado más de dos noticias en un mismo día, su nombre y el número total de visitas recibidas por esas noticias.

```
SELECT a.nombre, sum(numVisitas)
FROM autor a JOIN noticia n ON a.aid = n.aid
GROUP BY a.aid, a.nombre, n.fecha -- Fecha no puede contener información de la hora.
HAVING count(*) > 2;
```

c. (0,5 puntos) Escribe una consulta en lenguaje SQL que muestre el nombre de todos los autores y el nombre de los periódicos donde han publicado noticias. Si un autor no ha publicado en ningún periódico, debe mostrar el texto '(ninguno)' en la columna del nombre del periódico.

```
SELECT DISTINCT a.nombre, NVL(p.nombre,'(ninguno)')
FROM autor a LEFT JOIN (noticia n JOIN periodico p ON n.pid = p.pid) ON a.aid = n.aid;
```

d. (0,75 puntos) Escribe una consulta en lenguaje SQL que muestre las fechas en las que todos los medios han publicado alguna noticia, junto con el número de noticias que se han publicado en esa fecha y el número total de visitas recibidas.

```
SELECT n.fecha, count(*), sum(n.numVisitas)
FROM noticia n
GROUP BY n.fecha
HAVING NOT EXISTS
  (SELECT p.pid
  FROM periodico p
  WHERE p.pid NOT IN
    (SELECT p2.pid
    FROM periodico p2
    JOIN noticia n2 ON p2.pid = n2.pid
    WHERE n2.fecha = n.fecha));

-- Solución alternativa:

SELECT n.fecha, count(*), sum(n.numVisitas)
FROM noticia n
GROUP BY n.fecha
HAVING count(distinct pid) = (select count(*) from periodico);
```

e. (0,75 puntos) Escribe una consulta en lenguaje SQL que muestre el nombre de los autores que no han publicado noticias en ningún periódico en el que haya publicado el autor 'Pedro Santillana'.

```
SELECT a.nombre
FROM autor a
WHERE a.aid NOT IN (SELECT n.aid
                    FROM noticia n
                    WHERE n.pid IN (SELECT n2.pid
                                    FROM noticia n2 JOIN autor a2 ON n2.aid = a2.aid
                                    WHERE a2.nombre = 'Pedro Santillana'));
-- Solución alternativa:

SELECT a.nombre FROM autor a WHERE NOT EXISTS
  (SELECT n.pid FROM noticia n WHERE n.aid = a.aid
    intersect
    SELECT n2.pid FROM noticia n2 JOIN autor a2 ON n2.aid = a2.aid
      WHERE a2.nombre = 'Pedro Santillana');
```

f. (1,5 puntos) Crea un procedimiento almacenado llamado noticiasAutor que reciba como parámetro el id de un autor (idAutor) y muestre por pantalla sus datos personales (aid, nombre y sección), junto con un listado con las noticias (titular y fecha) que ha publicado ordenadas crecientemente por fecha. En caso de error (el id del autor no existe o no hay noticias para ese autor), deberá mostrarse por pantalla un mensaje de advertencia explicando el tipo de error (el autor no existe o no tiene noticias publicadas). Al finalizar el listado se deberá mostrar la suma de las visitas de todas las noticias del autor.

```
CREATE OR REPLACE PROCEDURE noticiasAutor ((idAutor IN autor.aid%TYPE) AS
  nombre autor.nombre%TYPE;
  seccion autor.seccion%TYPE;

  CURSOR c_noticias is
    SELECT titular, to_char(fecha, ' dd/mm/yy hh24:mi:ss') fechaNoticia, numVisitas
    FROM noticia
    WHERE aid = idAutor
    ORDER BY fecha;

  eVacio EXCEPTION;
  lNoticias c_noticias%ROWTYPE;
  sumVisitas noticia.numVisitas%TYPE;

BEGIN
```

```
 SELECT a.nombre, a.seccion
 INTO nombre, seccion
 FROM autor a
 WHERE aid = idAutor;

 DBMS_OUTPUT.PUT_LINE('-------------------------------------------------');
 DBMS_OUTPUT.PUT_LINE('Id: ' || idAutor);
 DBMS_OUTPUT.PUT_LINE('Nombre: ' || nombre);
 DBMS_OUTPUT.PUT_LINE('Seccion: ' || seccion);
 DBMS_OUTPUT.NEW_LINE();

 sumVisitas:=0;
 OPEN c_noticias;
 FETCH c_noticias INTO lNoticias;
 IF c_noticias%ROWCOUNT = 0 THEN
  RAISE eVacio;
 ELSE
  DBMS_OUTPUT.PUT_LINE('Noticias:');
  WHILE c_noticias%found
  LOOP
   DBMS_OUTPUT.PUT_LINE('    Titular: '|| lNoticias.titular);
   DBMS_OUTPUT.PUT_LINE('    Fecha: '|| lNoticias.fechaNoticia);
   DBMS_OUTPUT.NEW_LINE();
   sumVisitas:= sumVisitas + lNoticias.numVisitas;
   FETCH c_noticias INTO lNoticias;
  END LOOP;
  DBMS_OUTPUT.PUT_LINE('Total Visitas: '|| sumVisitas);
 END IF;
 CLOSE c_noticias;
EXCEPTION
 WHEN no_data_found THEN
  DBMS_OUTPUT.PUT_LINE('-------------------------------------------------');
  DBMS_OUTPUT.PUT_LINE('ERROR: Autor ' || idAutor || ' no encontrado');
 WHEN eVacio THEN
  DBMS_OUTPUT.PUT_LINE('No hay noticias para el autor con id:' || idAutor);
END;
/
```

g. (1.5 puntos) Viendo los logs de accesos a nuestra base de datos hemos visto que el número de visitas por autor es algo muy demandado por lo que hemos decidido añadir una nueva columna a la tabla autor que guarde el número de visitas de todos los artículos escritos por cada autor. Para que nuestra base de datos recoja esta nueva información necesitamos que hagas lo siguiente:

- Modificar la tabla autor para añadir una nueva columna llamada numVisitas.

  ```
  ALTER TABLE autor ADD numVisitas INTEGER;
  ```

- Crear un disparador que, cada vez que en la tabla noticias se inserte una noticia o se actualice el número de visitas de una noticia, se cambie automáticamente el número de visitas del autor de la noticia en la tabla autor.

  ```
  CREATE OR REPLACE TRIGGER updateNumVisitas
   AFTER INSERT OR UPDATE ON noticia
   FOR EACH ROW
   BEGIN
      IF INSERTING THEN
            UPDATE autor
            SET numVisitas = numVisitas + :new.numVisitas
            WHERE aid = :new.aid;
     ELSIF UPDATING THEN
            UPDATE autor
            SET numVisitas = numVisitas - :old.numVisitas + :new.numVisitas
            WHERE aid = :new.aid;
     END IF;
   END;
  ```

Ejercicio 4. (1 punto) Dada la tabla **autor(aid, Nombre, Seccion, numVisitas)** vacía considera la ejecución de la siguiente lista de instrucciones en SQLDeveloper asumiendo que **autocommit= off**

```
   savepoint paso_uno;
   INSERT INTO autor VALUES (101, 'C. Romero', 'Internacional', 200);
-- paso 1 –

   SAVEPOINT paso_dos;
   UPDATE  autor
   SET numVisitas = numVisitas + 100
   WHERE aid = 101;
-- paso 2 –

   ROLLBACK TO SAVEPOINT paso_dos;
-- paso 3 –

   UPDATE  autor
   SET numVisitas = numVisitas + 200
   WHERE aid = 101;
-- paso 4 –

   ROLLBACK;
-- paso 5 –

   INSERT INTO autor VALUES (101, 'C. Romero', 'Internacional', 1000);

   UPDATE   autor
   SET numVisitas = numVisitas + 300
   WHERE aid = 101;
-- paso 6  --

   SAVEPOINT paso_tres;
   COMMIT;
-- paso 7 –

   CREATE TABLE autorEstrella(aid NUMBER(8), nombre varchar(20), numVisitas number(5));

   INSERT INTO autorEstrella VALUES(202,'A. Shepherd', 5000);

   ROLLBACK TO SAVEPOINT paso_tres;
-- paso 8 --

   SELECT * FROM  autorEstrella WHERE aid = 202;

   ROLLBACK;
-- paso 9 --
```

a) Describe qué valor tiene numVisitas para la fila C.Romero exactamente en el momento indicado por cada comentario *-- paso N --* (aunque todavía no sea un valor definitivo).

-- paso 1 → 200
-- paso 2 → 300
-- paso 3 → 200
-- paso 4 → 400
-- paso 5 → No existe la fila 101
-- paso 6 → 1300
-- paso 7 → 1300
-- paso 8 → 1300
-- paso 9 → 1300

b) Indica con qué instrucción empieza cada una de las transacciones de la secuencia.

Transacción 1 → INSERT INTO autor VALUES (101, 'C. Romero', 'Internacional', 200);

Transacción 2 → INSERT INTO autor VALUES (101, 'C. Romero', 'Internacional', 1000);
Transacción 3 → CREATE TABLE autorEstrella(aid NUMBER(8), nombre varchar(20), numVisitas number(5));
Transacción 4 → INSERT INTO autorEstrella VALUES(202,'A. Shepherd', 5000);

c) ¿Se produce algún error? Si lo hay, indica en qué instrucción y por qué.
ROLLBACK TO SAVEPOINT paso_tres; en paso 8 da error ORA-01086, no existe en la transacción ese save_point

d) ¿Qué tablas quedan al final de la ejecución?
autor y autorEstrella (el rollback no deshace las instrucciones DDL)

Solución todo seguido:

```
savepoint paso_uno;  -- crea una transacción
INSERT INTO autor VALUES (101, 'C. Romero', 'Internacional', 200);
   -- paso 1 -- tengo 200 , empieza transacción primera

savepoint paso_dos;
update  autor
set EntradasVendidas = numVisitas + 100
where aid = 101;
-- paso 2
   -- tengo 300

rollback to savepoint paso_dos;
-- paso 3
   -- tengo 200, sigo en transacción primera

update  autor
set numVisitas = numVisitas + 200
where aid = 101;
-- paso 4
   -- tengo 400

rollback;
-- paso 5
   -- elimina insert (101), cierra trans, no trans activa

INSERT INTO autor VALUES (101, 'C. Romero', 'Internacional', 1000);
   --   empieza segunda transacción

update  autor
set numVisitas = numVisitas + 300
where aid = 101;
-- paso 6
   -- tengo 1300

savepoint paso_tres;
commit;
-- paso 7
   -- tengo 1300

CREATE TABLE autorEstrella(aid NUMBER(8), nombre varchar(20), numVisitas number(5));
   --  empieza la tercera transacción, pero termina porque tiene
   --  un commit implícito. Se queda sin transacción activa

INSERT INTO autorEstrella VALUES(202,'A. Shepherd', 5000);
   --  empieza la cuarta transacción
   --     (o tercera si no cuentas la create)

rollback to savepoint paso_tres;
```

```
-- paso 8
   -- da error ORA-01086, no existe la transacción
   -- con ese save_point (y no cambia la transacción)

select * from  autorEstrella  where aid = 202;

rollback;
-- paso 9
   -- tengo 1300,
   -- elimina Enanieves (insert), termina trans
   --  no elimina la tabla autorEstrella por ser DDL
```