**EJERCICIO 1.** (3 Puntos) **Modelado.**

Una nueva editorial de artículos científicos desea crear la base de datos en la que almacenar toda la información relativa a su negocio. Por ahorrarse unos pocos euros y el esfuerzo de definir los requisitos, la editorial aprovecha que un trabajador suyo tiene una antigua copia de seguridad de la base de datos de la última editorial en la que trabajó, y nos solicita que repliquemos la misma estructura a partir de dicha copia de seguridad, confiada en que si a la otra editorial le valía, a ellos también.

Los ficheros de la copia de seguridad y un extracto con la cabecera de su contenido son los siguientes:

- **journals.csv**

  ```
  journal_id;journal_name;JIF;JIF_Quartile;editorial_id
  ```

- **affiliations.csv**

  ```
  affiliation_id;affiliation_name;city;country
  ```

- **articles.csv**

  ```
  DOI;title;publication_date;num_citations;journal_id
  ```

- **authors.csv**

  ```
  author_id;author_name;department_id
  ```

- **author_affiliation.csv**

  ```
  author_id;affiliation_id
  ```

- **author_article.csv**

  ```
  DOI;author_id
  ```

- **cites.csv**

  ```
  DOI_1;DOI_2
  ```

- **editorials.csv**

  ```
  editorial_id;name
  ```

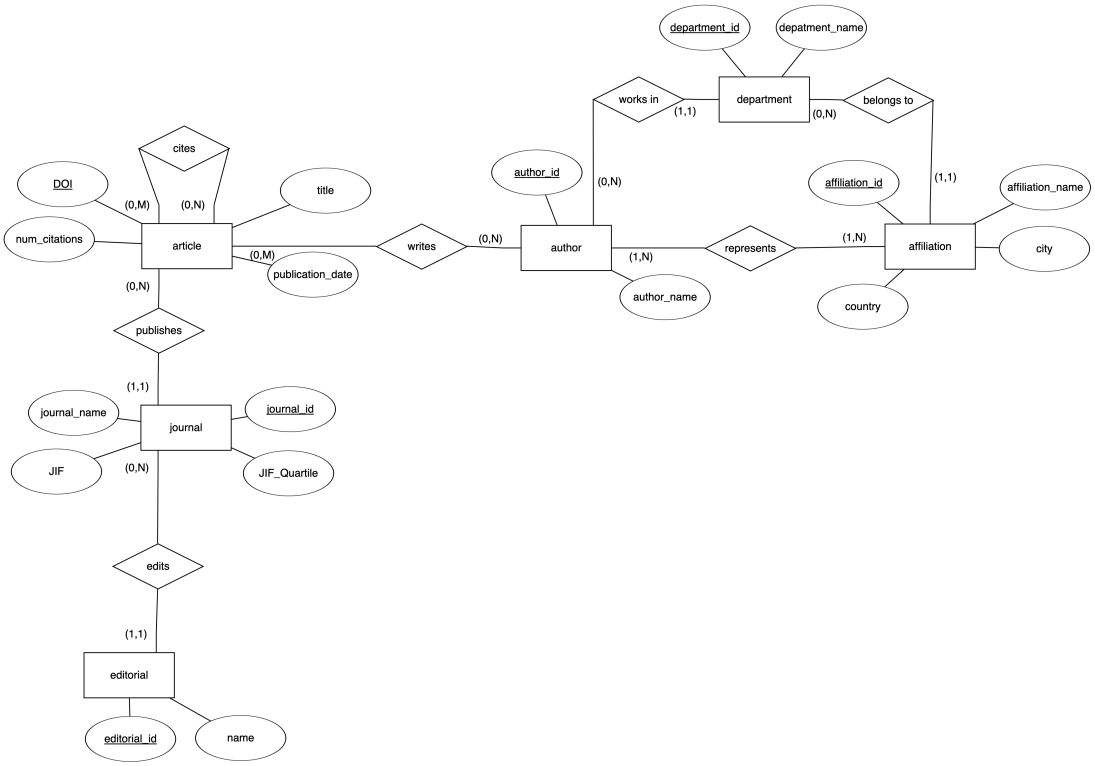
- **department.csv**

  ```
  department_id;department_name,affiliation_id
  ```

Se pide:

- A partir de dichos ficheros, defina el modelo conceptual que permita almacenar toda la información disponible en los ficheros de la copia de seguridad. Será obligatorio utilizar el modelo Entidad-Relación con notación Chen.
- Justificar al menos cuatro cardinales mínimas del modelo propuesto.

**Solución propuesta:**




Cardinalidades mínimas:

- Un autor debe pertenecer al menos a una afiliación, puesto que para poder publicar es necesario estar vinculado con algún organismo.
- Un autor debe trabajar en al menos un departamento, ya que no está permitido que autores publiquen sin pertenecer a alguno.
- Un departamento debe estar afiliado en al menos una afiliación, ya que un departamento debe pertenecer a una organización, no puede ser independiente.
- Un artículo debe publicarse en al menos un *journal*, puesto que para realizar el acto de publicar es necesario que se publiquen en algún lugar.

**EJERCICIO 2.** (2½ Puntos) **Álgebra relacional y SQL.**

Una editorial académica dispone de una base de datos relacional compuesta por las siguientes **tablas** en las que recoge información sobre sus revistas (`JOURNAL`) y artículos publicados en las mismas.

*NOTA: El subrayado continuo indica PK (clave primaria), mientras que la cursiva indica FK (clave foránea) con referencia a la PK con la que comparte nombre.*

`KEYWORDS(`<u>`keyword_id`</u>`, keyword_name)`

`ARTICLE_KEYWORDS(`<u>*`DOI`*</u>`, `<u>*`keyword_id`*</u>`)`

`ARTICLE(`<u>`DOI`</u>`, title, date_published, `*`journal_id`*`)`

`JOURNAL(`<u>`journal_id`</u>`, journal_name, IF)`

`JOURNAL_TOPIC(`<u>*`journal_id`*</u>`, `<u>*`topic_id`*</u>`)`

`TOPIC(`<u>`topic_id`</u>`, topic_name)`

`AUTHOR(`<u>`author_id`</u>`, name)`

`AUTHOR_ARTICLE(`<u>*`DOI`*</u>`,`<u>*`author_id`*</u>`)`

`CITATIONS(`<u>*`DOI_citing`*</u>`,`<u>*`DOI_cited`*</u>`)`

(a) (½ Punto) Álgebra relacional

i. (½ Punto) Obtener el nombre de los *journals* que solamente han publicado artículos a partir del año 2023 (del 2023 incluido en adelante). Considere para ello que todos los *journals* han publicado al menos un artículo.

**Solución propuesta:**

$$\Pi_{journal_name}([\Pi_{journal_id}(Journal) - \Pi_{journal_id}\left(\sigma_{date_published<2023}(Article)\right)] \bowtie Journal)$$

(b) (2 Puntos) SQL

i. (½ Punto) Obtener el DOI, el título del artículo y número de *keywords* de los artículos que tengan el mayor número de *keywords* de la base de datos.

**Solución propuesta:**

```
SELECT a.DOI, a.title, COUNT(*)
FROM article a
    INNER JOIN article_keywords ak ON a.DOI = ak.DOI
GROUP BY a.DOI, a.title
```

```
HAVING COUNT(*) >= ALL(SELECT COUNT(*)
                       FROM article_keywords
                       GROUP BY DOI);
```

II. (½ Punto) Obtener el nombre de los *journals* que hayan publicado en todos los *topics*.

**Solución propuesta:**

```
SELECT j.journal_name
FROM JOURNAL j
WHERE NOT EXISTS (
  SELECT t.topic_id
  FROM TOPIC t
  WHERE NOT EXISTS (
      SELECT *
      FROM JOURNAL_TOPIC jt
      WHERE jt.journal_id = j.journal_id
        AND jt.topic_id = t.topic_id
  )
);
```

III. (½ Punto) Obtener el título de los artículos que no fueron publicados en *journals* con el *topic* "bases de datos" durante el año 2023 (desde el 01/01/2023 hasta el 31/12/2023).

**Solución propuesta:**

```
SELECT a.title
FROM article a
WHERE a.date_published >= '2023-01-01'
AND a.date_published <= '2023-12-31'
AND a.journal_id NOT IN (
    SELECT jt.journal_id
    FROM journal_topic jt
        INNER JOIN topic t ON jt.topic_id = t.topic_id
    WHERE t.topic_name = 'bases de datos');
```

IV. (½ Punto) Obtener el título de los artículos que no contienen las keywords "machine learning" ni "deep learning".

**Solución propuesta:**

```
SELECT a.title
FROM article a
INNER JOIN article_keywords ak ON a.DOI = ak.DOI
INNER JOIN keywords k ON ak.keyword_id = k.keyword_id
WHERE a.DOI NOT IN (
  SELECT ak.DOI
```

```
  FROM article_keywords ak
    INNER JOIN keywords k ON ak.keywords_id = k.
       keywords_id
  WHERE k.keyword_name = 'machine learning')
AND a.DOI NOT IN (
  SELECT ak.DOI
  FROM article_keywords ak
    INNER JOIN keywords k ON ak.keywords_id = k.
       keywords_id
  WHERE k.keyword_name = 'deep learning');
```

**EJERCICIO 3.** (2½ Puntos) **Procedimientos, triggers y funciones.**

La editorial descrita en el ejercicio 2 quiere mantener unos estándares de calidad científica y para ello, entre otras cosas, ha definido una política de concienciación sobre las autocitas excesivas (autores en cuyos artículos referencian a otros artículos del mismo autor). Esta política consiste en detectar si un autor tiene un 20 % o más de autocitas sobre el total de citas del artículo que se ha publicado, y en ese caso, avisar a dichos autores. La primera acción realizada ha sido añadir una nueva columna llamada *self_ citations* a la tabla `AUTHOR_ARTICLE` para registrarlo.

(a) (½ Punto) Función: programe una función `get_self_citations` en SQL que reciba el identificador de un artículo (`DOI`) y un identificador de un autor (`author_id`) y devuelva un número entero con el número de veces que el autor se ha citado a sí mismo dentro de ese artículo.

**Solución propuesta:**

```
DELIMITER &&
CREATE FUNCTION get_self_citations(id_art INT, id_au INT)
RETURNS INT
DETERMINISTIC
BEGIN
    DECLARE autocitas INT DEFAULT 0;
    SELECT COUNT(*) INTO autocitas
    FROM AUTHOR_ARTICLE
        JOIN CITATIONS
            ON AUTHOR_ARTICLE.DOI = CITATIONS.DOI_cited
    WHERE CITATIONS.DOI_citing = id_art
      AND AUTHOR_ARTICLE.author_id = id_au;
    RETURN (autocitas);
END&&
DELIMITER ;
```

(b) (1 Punto) Procedmiento: programe un procedimiento en SQL sin parámetros de entrada ni de salida, que utilice un cursor para recorrer las tuplas de la tabla `AUTHOR_ARTICLE`, calcule el número de autocitas del autor en su artículo (utilizando la función anterior), y por último, rellene la nueva columna `self_citations` de la tabla que se creó tal y como se especificó en el enunciado de este ejercicio. NOTA: el cursor solo debe recorrer aquellas tuplas SIN la información de las autocitas (valor NULL).

**Solución propuesta:**

```
DELIMITER &&
CREATE PROCEDURE fill_self_citations()
BEGIN
    DECLARE done INT DEFAULT false;
    DECLARE id_art, id_au, selfcit INT;
    DECLARE cur1 CURSOR FOR SELECT *
                            FROM AUTHOR_ARTICLE
```

```
                              WHERE self_citations IS NULL;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = true;
    OPEN cur1;
    loop1: LOOP
        FETCH cur1 INTO id_art, id_au, selfcit;
        IF done THEN
            LEAVE loop1;
        END IF;
        UPDATE AUTHOR_ARTICLE
        SET self_citations = get_self_citations(id_art, id_au
           )
        WHERE DOI=id_art AND author_id=id_au;
  END LOOP;
    CLOSE cur1;
END &&
DELIMITER ;
```

(c) (1 Punto) Trigger: programe un trigger en el que cada vez que se actualice la tabla `AUTHOR_ARTICLE` con la información de las autocitas, se calcule el porcentaje de autocitas sobre el total de citas que tiene ese artículo (`num_citations`) y en caso de superar el porcentaje del 20 %, impida la inserción y devuelva un ERROR con un mensaje donde aparezca el `author_id` del autor que viole esta política (puede para ello usar el método `concat()`).

**Solución propuesta:**

```
DELIMITER &&
CREATE TRIGGER check_self_citations
BEFORE UPDATE ON AUTHOR_ARTICLE
FOR EACH ROW
BEGIN
  DECLARE n_cit INT DEFAULT 0;
  DECLARE error_msg VARCHAR(250);

  SELECT COUNT(*) INTO n_cit
  FROM CITATIONS
  WHERE DOI_citing = NEW.DOI;

  IF (NEW.self_citations IS NOT NULL) THEN
    IF (NEW.self_citations/n_cit >= 0.2) THEN
      SET error_msg = concat('The author ', NEW.author_id,
       ' has exceeded the self-citations ratio in article ',
          NEW.DOI);
      SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = error_msg;
    END IF;
  END IF;
END &&
DELIMITER ;
```

**EJERCICIO 4.** (1 Punto) **Control de acceso.**

Cree una vista en SQL que permita conocer el número de artículos publicados por cada `journal` del modelo de base de datos del ejercicio 2. Las columnas de la vista deberán ser el nombre del journal (`journal_name`) y el número total de artículos publicados. A continuación, cree un usuario denominado '`consultor`' y otorguele únicamente permisos de consulta sobre esa vista.

**Solución propuesta:**

```
CREATE VIEW num_articles AS
SELECT journal_name, COUNT(*)
FROM article NATURAL JOIN journal
GROUP BY journal_id, journal_name;

CREATE USER 'consultor' IDENTIFIED BY '1234';

GRANT SELECT ON num_articles TO 'consultor';
```

**EJERCICIO 5.** (1 Punto) **Programación contra bases de datos.**

Realiza el etiquetado de clases y sus atributos para que mediante el ORM Hibernate de Java se pueda realizar la conexión con la base de datos satisfactoriamente cumpliendo con el **modelo relacional mostrado en el ejercicio 2**. Ten en cuenta que las claves numéricas deberán generarse automáticamente y que los atributos de nombre no pueden ser nulos.

**No repliques el código**. Incluye las anotaciones en el código Java que se muestra a continuación.

*NOTA: Con el fin de simplificar la solución, algunas relaciones presentes en el modelo relacional del ejercicio 2 han sido intencionadamente excluidas en el código de este ejercicio. Utiliza únicamente las clases y atributos proporcionados.*

```
public class Journal {

    private Long id;

    private String name;

    private Float impactFactor;

    private Set<Article> articles;

    private Set<Topic> topics;
}

public class Article {

    private String doi;

    private String title;

    private Date datePublished;

    private Journal journal;
```

```
}

public class Topic {

    private Long id;

    private String name;

    private Set<Journal> journals;
}
```

**Solución propuesta:**

```
@Entity
@Table(name = "JOURNAL")
public class Journal {
    @Id
    @GeneratedValue
    @Column(name = "journal_id")
    private Long id;

    @Column(name = "journal_name", nullable = false)
    private String name;

    @Column(name = "impactFactor")
    private Float impactFactor;

    @OneToMany(mappedBy = "journal")
    private Set<Article> articles;

    @ManyToMany
    @JoinTable(name = "JOURNAL_TOPIC",
                joinColumns = @JoinColumn(name = "journal_id"))
    private Set<Topic> topics;
}

@Entity
@Table(name = "ARTICLE")
public class Article {
```

```
    @Id
    @Column(name = "DOI", nullable = false)
    private String doi;

    @Column(name = "title", nullable = false)
    private String title;

    @Column(name = "date_published")
    @Temporal(TemporalType.DATE)
    private Date datePublished;

    @ManyToOne
    @JoinColumn(name = "journal_id", nullable = false)
    private Journal journal;
}

@Entity
@Table(name = "TOPIC")
public class Topic {
    @Id
    @GeneratedValue
    @Column(name = "topic_id")
    private Long id;

    @Column(name = "topic_name", nullable = false)
    private String name;

    @ManyToMany(mappedBy = "topics")
    private Set<Journal> journals;
}
```