Objetivos

Reforzar los conceptos básicos de Bases de Datos y adquirir autonomía

en el diseño del esquema de una base de datos relacional.

Practicar la recuperación de información mediante archivos de texto plano.

Jose Manuel Pinillos Rubio

BASES DE DATOS avanzadas

Diseño de una base de datos relacional

ÍNDICE

[ÍNDICE 2](#_Toc138402546)

[INTRODUCCIÓN 4](#_Toc138402547)

[DISEÑO DEL DIAGRAMA ENTIDAD-RELACIÓN 5](#_Toc138402548)

[Captura de requisitos 5](#_Toc138402549)

[Diseño conceptual 5](#_Toc138402550)

[**Entidades** 5](#_Toc138402551)

[**Atributos** 5](#_Toc138402552)

[**Relaciones** 6](#_Toc138402553)

[**Relaciones entre tablas** 6](#_Toc138402554)

[**Cardinalidades** 6](#_Toc138402555)

[**Claves primarias y foráneas** 7](#_Toc138402556)

[Diagrama Entidad-Relación 8](#_Toc138402557)

[Diseño lógico 9](#_Toc138402558)

[Diseño físico 10](#_Toc138402559)

[CONSULTAS 12](#_Toc138402560)

[**1. Relación de departamentos y sus directores** 12](#_Toc138402561)

[**2. Relación de profesores y sus departamentos** 12](#_Toc138402562)

[**3. Relación de los alumnos y sus asignaturas matriculadas** 12](#_Toc138402563)

[**4. Relación de alumno, asignaturas y profesores que las imparten** 12](#_Toc138402564)

[**5. Relación de asignaturas y departamentos que las organizan** 13](#_Toc138402565)

[**6. Relación de profesores y su categoría** 13](#_Toc138402566)

[**7. Relación de alumnos y su id** 13](#_Toc138402567)

[**8. Relación profesores y su id** 13](#_Toc138402568)

[**9. Relación de asignaturas y su id** 13](#_Toc138402569)

[**10. Relación de departamentos y su id** 13](#_Toc138402570)

[TÉCNICAS UTILIZADAS 14](#_Toc138402571)

[**Relación "dirigir" entre "profesor" y "departamento"** 14](#_Toc138402572)

[**Relación "pertenecer" entre "profesor" y "departamento"** 14](#_Toc138402573)

[**Relación "matricularse" entre "alumno" y "asignatura"** 15](#_Toc138402574)

[**Relación "impartir" entre "alumno", "asignatura" y "profesor"** 15](#_Toc138402575)

[**Relación "organizar" entre "asignatura" y "departamento"** 15](#_Toc138402576)

[CONCLUSIONES 16](#_Toc138402577)

**INTRODUCCIÓN**

La gestión de bases de datos en SQL es un componente fundamental en el ámbito de la Ingeniería Informática. En este trabajo, se aborda el diseño y la implementación de un sistema de bases de datos utilizando el enfoque del diagrama Entidad-Relación y su transformación en tablas relacionales. Además, se realizará la recuperación de información mediante archivos de texto plano, en este caso CSV. El objetivo principal será aplicar los conocimientos adquiridos en esta área mediante el diseño de una base de datos relacional, a partir de los datos de precios de combustibles en las diferentes comunidades autónomas de España y su posterior recuperación a nuestra base de datos.

El proceso se ha llevado a cabo siguiendo un orden estructurado. En primer lugar, se descargaron los datos con los que se trabajarían, para después realizar una limpieza de estos y exportarlos en un formato de texto plano.

Posteriormente, se realizó el diseñó el diagrama Entidad-Relación, que permitió representar las entidades, atributos y relaciones presentes del dominio de la base de datos. Se ha puesto especial énfasis en comprender y aplicar las cardinalidades, tanto para establecer las relaciones fuertes como para identificar las relaciones débiles.

A continuación, se procedió a la transformación del diagrama Entidad-Relación en un modelo relacional mediante un script SQL. Este proceso implica la creación de tablas que reflejen adecuadamente la estructura y las restricciones de la base de datos, garantizando la integridad y consistencia de los datos.

Posteriormente, se implementó el código de JDBC para la inserción de los datos correspondientes a a nuestra base de datos. Esta fase fue crucial para poblar la base de datos y permitir la realización de consultas y análisis posteriores.

Finalmente, se llevaron a cabo diversas consultas sobre la base de datos, lo que permitió aplicar y evaluar la funcionalidad del modelo diseñado. Cada relación presente en el diagrama Entidad-Relación fue acompañada de una descripción breve de la técnica utilizada, destacando su relevancia y aplicabilidad en el contexto del proyecto.

**DISEÑO DEL DIAGRAMA ENTIDAD-RELACIÓN**

En este apartado especificare los pasos seguidos para la realización del diagrama Entidad-Relación.

## **Captura de requisitos**

Lo primero que realizamos fue un correcto análisis de los requisitos mostrados en el enunciado para diseñar una base de datos relacional para la recuperación de datos de combustibles de España.

## **Diseño conceptual**

En segundo lugar, se capturaron todos los elementos necesarios para representar la estructura lógica de la base de datos, mostrando las entidades, sus atributos, las relaciones entre ellas y las restricciones que rigen dichas relaciones.

### **Entidades**

* Provincias → provinces.
* Municipios → municipalities.
* Localidades → localities.
* Operadores → operators.
* Combustibles → fuels.
* Precios → prices.
* Estaciones de servicio → stations.

### **Atributos**

* **provinces**:
  + *pro\_id*
  + name
* **municipalities**:
  + *mun\_id*
  + pro\_id
  + name
* **localities**:
  + *loc\_id*
  + mun\_id
  + name
* **operator**:
  + *op\_id*
  + name
* **fuel**:
  + *fuel\_id*
  + name
* **price**:
  + *fuel\_id*
  + *st\_id*
  + name
* **station**:
  + *st\_id*
  + loc\_id
  + op\_id
  + fuel\_id
  + cp
  + latitude
  + longitude
  + margin
  + address
  + price\_date
  + schedule
  + type

### **Relaciones**

* Gestiona → gestiona
* Tiene → tiene
* Pertenece → pertenece

### **Relaciones entre tablas**

* municipalities → pertenece → provinces
* localities → pertenece → municipalities
* station → pertenece → localities
* operator → gestiona → station
* station → tiene → fuel
* fuel → tiene → prices

### **Claves primarias y foráneas**

* **provinces**:
  + Clave primaria: *pro\_id*
* **municipalities**:
  + Clave primaria: *mun\_id*
  + Clave foránea: pro\_id
* **localities**:
  + Clave primaria: *loc\_id*
  + Clave foránea: mun\_id
* **operator**:
  + Clave primaria: *op\_id*
* **fuel**:
  + Clave primaria: *fuel\_id*
* **price**:
  + Claves primarias: *fuel\_id, st\_id*
* **station**:
  + Clave primaria: *st\_id*
  + Claves foráneas: loc\_id, op\_id, fuel\_id

## **Diagrama Entidad-Relación**

**Explicación de las relaciones**

A continuación, se realizará una breve explicación de las relaciones entre las entidades descritas en el diagrama entidad-relación.

### **Relación "pertenece" entre "municipalities" y "provinces"**

En esta relación, cada municipio pertenece exactamente a una provincia, y a cada provincia le pueden pertenecer varios municipios, lo que implica una relación **1:N**.

El enlace entre estas entidades se hace a través de la clave ***pro\_id***. Se generan las claves foráneas en la entidad **municipalities** que se relaciona con la clave primaria de la entidad **provinces**.

### **Relación "pertenece" entre "localities" y " municipalities "**

En esta relación, cada localidad pertenece exactamente a un municipio, y a cada municipio le pueden pertenecer varias localidades, lo que implica una relación **1:N**.

El enlace entre estas entidades se hace a través de la clave ***mun\_id***. Se generan las claves foráneas en la entidad **localities** que se relaciona con la clave primaria de la entidad **municipalities**.

### **Relación "gestiona" entre "operator" y " station "**

En esta relación, cada operadora puede gestionar varias estaciones de servicio, y cada estación de servicio es dirigida por un único operador, lo que implica una relación **1:N**.

El enlace entre estas entidades se hace a través de la clave ***op\_id***. Se generan las claves foráneas en la entidad **station** que se relaciona con la clave primaria de la entidad **operator**.

### **Relación "tiene" entre "station" y " fuel "**

En esta relación, cada estación de servicio puede tener varios carburantes, y cada carburante puede pertenecer a varias estaciones de servicio, lo que implica una relación **M:N**.

El enlace entre estas entidades se hace a través de la clave ***fuel\_id***. Se generan las claves foráneas en la entidad **station** que se relaciona con la clave primaria de la entidad **fuel**.

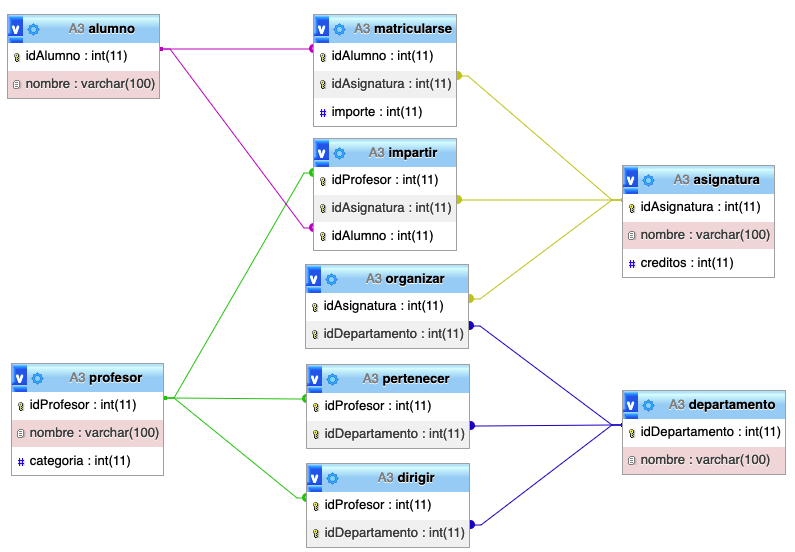
### **Relación "tiene" entre "fuel" y " price "**

En esta relación, cada combustible puede tener varios precios, y cada precio puede ser el mismo para varios combustibles, lo que implica una relación **M:N**.

El enlace entre estas entidades se hace a través de las clave ***fuel\_id***. Se generan las claves foráneas en la entidad **price** que se relaciona con la clave primaria de la entidad **fuel**.

## **Diseño lógico**

En este apartado muestro la transformación del diagrama entidad-relación en un esquema lógico, en el que se representa las tablas, la estructuración de datos y el modelado de restricciones disponibles.



## **Diseño físico**

Una vez definido el esquema lógico, en este apartado especificare los pasos seguidos para la creación de la base de datos, la creación de las tablas y la inserción de los datos a nuestra BD.

1. **Creación de la base de datos**: Para crear una base de datos, se utiliza el comando **CREATE DATABASE** en SQL. Por ejemplo, para crear nuestra base de datos llamada "*actividad*\_3", debemos escribir:

CREATE DATABASE actividad\_3;

1. **Creación de tablas**: Para crear una tabla, se utiliza el comando **CREATE TABLE** en SQL. El siguiente es un ejemplo de cómo crear la tabla llamada "*alumno*" con dos campos: *idAlumno*, *nombre*.

CREATE TABLE alumno (

  idAlumno int(11) NOT NULL,

  nombre varchar(100) NOT NULL

);

1. **Creación de relaciones entre tablas**: Si necesitamos crear relaciones entre tablas, debemos agregar campos de clave externa. En el siguiente ejemplo, relacionamos la tabla “*matricularse*” con la tabla “*alumno*”.

ALTER TABLE matricularse

  ADD CONSTRAINT matricularse\_ibfk1 FOREIGN KEY (idAlumno)

REFERENCES alumno (idAlumno) ON DELETE CASCADE ON UPDATE CASCADE;

1. **Inserción de datos**: Para insertar datos en una tabla, se utiliza el comando **INSERT INTO**. El siguiente es un ejemplo de cómo insertar los nuevos registros en la tabla "*alumno*".

INSERT INTO alumno (idAlumno, nombre) VALUES

(1, 'JuanMR'),

(2, 'LuisP'),

(3, 'Patricial'),

(4, 'ErikaJ'),

(5, 'JoanV'),

(6, 'JovannyV'),

(7, 'CarolD'),

(8, 'MarioVl'),

(9, 'IsabelP'),

(10, 'FelipeB'),

(11, 'JesusJ'),

(12, 'IvannaA');

Al introducir los datos en una tabla debemos tener en cuenta varios factores:

* Hay tablas que deberán contener datos antes que otras para que no se violen algunas de las restricciones (como las de clave externa).
* El orden en que escribimos los atributos debe ser el mismo en el que están dispuestos en las tablas. Si queremos introducirlo en distinto orden, deberemos indicarlo expresamente en la consulta
* Debemos tener en cuenta las restricciones de cada uno de nuestros atributos.

**CONSULTAS**

### **1. Relación de departamentos y sus directores**

SELECT departamento.nombre as departamento, profesor.nombre as profesor

FROM profesor, departamento, dirigir

WHERE departamento.idDepartamento = dirigir.idDepartamento AND

profesor.idProfesor = dirigir.idProfesor;

### **2. Relación de profesores y sus departamentos**

SELECT departamento.nombre as departamento, profesor.nombre as profesor

FROM profesor, departamento, pertenecer

WHERE departamento.idDepartamento = pertenecer.idDepartamento AND

profesor.idProfesor = pertenecer.idProfesor;

### **3. Relación de los alumnos y sus asignaturas matriculadas**

SELECT alumno.nombre as alumno, asignatura.nombre as asignatura, asignatura.creditos, matricularse.importe

FROM alumno, asignatura, matricularse

WHERE alumno.idAlumno = matricularse.idAlumno AND

asignatura.idAsignatura = matricularse.idAsignatura;

### **4. Relación de alumno, asignaturas y profesores que las imparten**

SELECT profesor.nombre as profesor, alumno.nombre as alumno, asignatura.nombre as asignatura

FROM profesor, alumno, asignatura, impartir

WHERE alumno.idAlumno = impartir.idAlumno AND

profesor.idProfesor = impartir.idProfesor AND

asignatura.idAsignatura = impartir.idAsignatura

ORDER BY profesor.nombre ASC;

### **5. Relación de asignaturas y departamentos que las organizan**

SELECT asignatura.nombre as asignatura, departamento.nombre as departamento

FROM asignatura, organizar, departamento

WHERE asignatura.idAsignatura = organizar.idAsignatura AND

departamento.idDepartamento = organizar.idDepartamento

ORDER BY asignatura.idAsignatura ASC;

### **6. Relación de profesores y su categoría**

SELECT nombre, categoria

FROM profesor;

### **7. Relación de alumnos y su id**

SELECT idAlumno, nombre

FROM alumno;

### **8. Relación profesores y su id**

SELECT profesor.idProfesor, profesor.nombre

FROM profesor;

### **9. Relación de asignaturas y su id**

SELECT idAsignatura, nombre

FROM asignatura;

### **10. Relación de departamentos y su id**

SELECT idDepartamento, nombre

FROM departamento;

**CONCLUSIONES**

El diseño de una base de datos, desde la creación del diagrama entidad-relación hasta la elección de las tablas adecuadas, puede ser un proceso desafiante debido a varias dificultades.

En primer lugar, comprender completamente los requisitos del sistema y las necesidades de los usuarios es esencial para diseñar una base de datos efectiva. Si los requisitos no están claros o son contradictorios, puede ser difícil identificar las entidades y las relaciones entre ellas.

Además, la complejidad de los datos puede dificultar el modelado en el diagrama entidad-relación, aunque en nuestro caso, esto no ha sido un problema al tratarse de datos sencillos.

La normalización también es un desafío en el diseño de la base de datos, eliminar la redundancia de datos y garantizar la integridad de la base de datos es esencial. Sin embargo, aplicar correctamente las reglas de normalización puede ser complicado, especialmente al tratar con relaciones complejas o tablas con muchos atributos.

El diseño de relaciones entre las entidades es otro aspecto difícil. Determinar el tipo de relación (uno a uno, uno a muchos, muchos a muchos) y su cardinalidad puede ser un desafío, ya que implica comprender en profundidad las interacciones y dependencias entre las entidades.

Además, es importante anticiparse a los cambios futuros en los requisitos del sistema. El diseño de la base de datos debe ser lo suficientemente flexible y adaptable para acomodar los cambios sin tener que rediseñar todo el sistema.

Finalmente, nos damos cuenta de que el diseño de una base de datos desde el diagrama entidad-relación hasta la elección de las tablas correctas implica superar varias dificultades y requiere de experiencia y de un enfoque meticuloso para garantizar un diseño eficiente.