EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO
Instituto Tecnológico de Cd. Cuauhtémoc

INFORME DEL 2DO. PERIODO SABÁTICO:
02/03/2025 al 01/09/2025
2do Periodo:

B.1.2 Elaboración de apuntes de asignatura: Fundamentos de Base de Datos / AEF-1031 del plan de estudio Sistemas Computacionales 2010

PRESENTADO POR: M.A. ENRIQUE GARCÍA GRAJEDA

Cd. Cuauhtémoc Chihuahua, Marzo de 2025

INTRODUCCIÓN

En todos los aspectos de la vida cotidiana, las personas buscamos herramientas que nos ayuden a facilitar nuestras actividades y a hacerlas más llevaderas. Si nos enfocamos en el ámbito organizacional, ya sea que se trate de empresas con fines de lucro o instituciones sin ánimo de lucro, es lógico que tengan que seguir un proceso administrativo que abarque la planeación, la organización, la dirección y el control. En cada una de estas etapas, utilizar herramientas adecuadas es clave para simplificar las tareas y lograr mejores resultados.

Una necesidad que suele ser común en cualquier organización es contar con información confiable y accesible, pues con base en ella se pueden tomar decisiones acertadas que ayuden a alcanzar los objetivos. Al final del día, quienes facilitan este acceso a la información son los conocidos Sistemas de Información, que se convierten en aliados esenciales para el éxito y el buen funcionamiento de cualquier organización.

Las bases de datos son como el corazón de la gestión de información en cualquier empresa o institución. Imagínate que tienes que manejar muchísimos datos todos los días; una base de datos te ayuda a tener todo en orden, como si tuvieras un archivero digital donde cada cosa tiene su lugar y es muy fácil encontrar lo que buscas cuando lo necesitas.

Además, no solo se trata de tener la información guardada, sino también de poder acceder a ella de manera rápida y segura. Gracias a las bases de datos, es posible implementar sistemas que cuiden la confidencialidad de los datos y aseguren que solo las personas autorizadas puedan ver o modificar la información. Así, puedes estar tranquilo de que tus datos están protegidos y en buenas manos.

Otro punto importante es la consistencia. Las bases de datos ayudan a que toda la información esté alineada y actualizada, sin errores ni duplicados, aunque varias personas estén trabajando al mismo tiempo. Esto se logra con reglas y validaciones que mantienen el orden y la calidad de los datos.

Contar con una base de datos bien hecha también te facilita la vida a la hora de tomar decisiones. Puedes generar reportes, hacer análisis y revisar estadísticas de manera sencilla, porque tienes acceso a datos precisos y actualizados. Esto te ayuda a tomar decisiones más acertadas y a planear mejor el futuro de tu organización.

Por si fuera poco, las bases de datos tienen la ventaja de que pueden crecer junto contigo. Si tu negocio se expande, la base de datos puede adaptarse y manejar más información sin que tengas que cambiar todo el sistema. Es flexible y escalable, lo que te permite estar preparado para cualquier cambio.

Esta asignatura aporta al perfil del egresado la capacidad para analizar, diseñar y gestionar sistemas de bases de datos conforme a los requerimientos del entorno para garantizar la integridad, disponibilidad y confidencialidad de la información, así como para desarrollar e implementar sistemas de información para la gestión de

procesos y apoyo en la toma de decisiones, utilizando metodologías basadas en estándares internacionales.

Es importante porque el estudiante adquiere las competencias en el análisis y el diseño de base de datos, que le permitirán desarrollar aplicaciones para sistemas de información robustos que ofrezcan garantía en el manejo de la información. Es conveniente mencionar que hoy en día la información forma parte del capital intangible de las organizaciones y cada vez se demandan sistemas de información que garanticen la integridad y seguridad de la misma.

La asignatura propicia el dominio de modelos de diseño de base de datos basados en reglas de normalización, de integridad y de seguridad.

Esta asignatura requiere como competencia previa que el estudiante comprenda y aplique los conceptos y propiedades de álgebra de conjuntos, relaciones y álgebra booleana adquiridas en matemáticas discretas. Se relaciona con asignaturas posteriores donde se apliquen bases de datos y desarrollen aplicaciones para el tratamiento de información.

ÍNDICE

ÍNDICE DE FIGURAS

TEMA 1 Introducción a las bases de datos

El tema uno proporciona al estudiante el sustento teórico de las bases de datos, como son los objetivos, los diferentes modelos, la clasificación, las áreas de aplicación y arquitecturas que sirven de fundamento para que el estudiante incursione en el área de conocimiento de base de datos. Se recomienda que, en el tema de Arquitectura de la base de datos, se aborden los temas de niveles de abstracción, tipos de usuarios y tipos de lenguajes.

1.1 Conceptos básicos

Imagina que tienes un gran archivero digital donde guardas información importante, como si fuera una colección organizada de papeles, pero todo está almacenado electrónicamente. Eso es una base de datos. Así, puedes guardar mucha información, buscarla fácil, modificarla cuando lo necesites y siempre tenerla a la mano, como si supieras dónde está cada carpeta y cada hoja dentro de tu archivero.

1. Datos e Información

¿Te ha pasado que tienes un montón de datos sueltos como nombres, números o lugares? Por sí solos no dicen mucho, por ejemplo: "Juan", "30", "México". Pero si los juntas y les das sentido, obtienes información útil, como: "Juan es un hombre de 30 años que vive en México". Así, los datos se convierten en información cuando los organizas y les das contexto.

2. Entidad

Piensa en las entidades como los protagonistas de tu historia. Son cosas o personas sobre las que quieres guardar información. Por ejemplo, en la base de datos de una escuela, los protagonistas serían los estudiantes, los profesores y las clases.

3. Atributo

Ahora imagina que cada protagonista tiene sus propias características. A eso se le llama atributo. Por ejemplo, cada estudiante tiene un número de control, nombre, apellido, edad y dirección. Cada atributo es como una columna en una tabla, mostrando una propiedad diferente de la entidad.

4. Relación

Las relaciones son como los vínculos que unen a los protagonistas. Por ejemplo, un profesor puede impartir varias clases, y esa conexión entre "Profesor" y "Clase" es lo que llamamos relación. Así se conecta la información y se evita repetir lo mismo varias veces.

5. Clave

Para no perderte entre tanta información, existen las claves. Son como los identificadores únicos, el número de matrícula de cada estudiante, por ejemplo. La clave primaria es ese identificador que nadie más tiene y no se puede dejar vacío.

Por otro lado, la clave foránea es como una referencia que conecta una tabla con otra, por ejemplo, el número de profesor en la tabla de clases, que señala quién imparte esa materia.

6. Tabla

Finalmente, imagina una tabla como una hoja de cálculo donde tienes filas y columnas. Cada fila es un registro (por ejemplo, los datos de un estudiante) y cada columna es un atributo (como el nombre o la edad). Así es como se organiza toda la información dentro de una base de datos relacional, para que puedas encontrar lo que buscas sin complicaciones.

1.2 Objetivos de las Bases de Datos

Imagina que el sistema gestor de bases de datos (SGBD) es como el cerebro de una empresa, encargado de guardar y organizar toda la información importante. Este sistema no solo es una gran colección de datos conectados entre sí, sino también el conjunto de programas que te ayudan a encontrar y usar esos datos cuando los necesitas. La base de datos, como si fuera una gran libreta, guarda todo lo relevante para el negocio.

La principal misión de un SGBD es que puedas guardar y recuperar información de manera sencilla y rápida, sin que sea un dolor de cabeza. Está pensado para manejar montones de datos, como si fuera una enorme biblioteca donde cada libro está bien ubicado y etiquetado. Además, se encarga de definir cómo se van a acomodar esos datos y de ofrecerte herramientas para modificarlos o consultarlos cuando lo requieras.

Y como en toda buena historia, la seguridad es clave: estos sistemas están diseñados para que la información esté protegida, aunque haya un apagón o alguien trate de entrar sin permiso. Así, puedes confiar en que tus datos estarán seguros y disponibles cuando los necesites.

1.3 Áreas de Aplicación de los Sistemas de Bases de datos

Las bases de datos están presentes en nuestra vida cotidiana, aunque a veces ni nos damos cuenta. Por ejemplo, cuando vas al banco y pides información sobre tus cuentas o tus préstamos, todo eso se gestiona gracias a una base de datos que tiene guardados tus datos y movimientos. Si alguna vez has reservado un vuelo, tu información de reserva y la planificación de los vuelos se manejan con bases de datos que, desde hace décadas, conectan terminales de todo el mundo para que todos estén en sintonía, ¡incluso usando redes telefónicas en sus inicios!

En las universidades, las bases de datos ayudan a registrar todo lo relacionado con los alumnos: sus datos personales, las materias en las que están inscritos y los cursos que toman. Así, nadie se pierde en el mar de información académica. Cuando pagas con tarjeta de crédito, cada compra queda registrada y, al final del

mes, se genera tu estado de cuenta gracias a la información guardada en una base de datos.

Las empresas de telecomunicaciones también las usan para llevar un control de tus llamadas, generar tus facturas mensuales y mantener actualizados los saldos de tus tarjetas de prepago. Además, almacenan todo lo necesario para que las redes funcionen correctamente. En el mundo de las finanzas, las bases de datos almacenan información de grandes empresas, ventas y compras de acciones y bonos, ayudando a que todo se mueva de manera ordenada.

Por si fuera poco, cuando compras algo en una tienda física o en línea, los datos de los clientes, los productos y las compras también se guardan en una base de datos. Como ves, prácticamente todas las empresas dependen de ellas para funcionar correctamente.

En las últimas décadas, las bases de datos han evolucionado junto con la tecnología. Los cajeros automáticos nos permitieron interactuar directamente con ellas, y los sistemas de respuesta por teléfono hicieron posible que cualquier persona pudiera consultar información sin ir a una sucursal. Con la llegada de Internet, el acceso se volvió mucho más sencillo: hoy en día, cuando buscas un libro o una canción en una tienda en línea, realmente estás navegando por una base de datos que te ayuda a encontrar justo lo que quieres.

Niveles de abstracción

- Nivel físico: Imagina este nivel como el sótano de una biblioteca, donde se guardan todos los libros y papeles importantes en estantes y cajas especiales. Aquí es donde se decide exactamente cómo y dónde se almacenan los datos en la computadora, hasta en los últimos detalles. Es el trabajo de los expertos en tecnología asegurarse de que todo esté bien acomodado y listo para usarse, aunque los usuarios jamás vean este proceso tan técnico.
- Nivel lógico: Ahora, subiendo un piso, llegamos a la sala principal de la biblioteca. Aquí todo está más ordenado y fácil de entender: cada libro tiene su lugar y sabemos qué información hay y cómo se conecta con la demás. Los administradores de la base de datos, como los bibliotecarios, deciden qué información vale la pena guardar y cómo organizarla, sin tener que preocuparse por los detalles del sótano.
- Nivel de vistas: Finalmente, este nivel es como las vitrinas o exhibiciones de la biblioteca. Los visitantes sólo ven una parte de toda la colección, la que realmente les interesa o necesitan. Así, cada usuario accede solo a la información que le corresponde, sin enredarse con datos de más. El sistema puede crear varias "vistas" diferentes para que cada quien encuentre justo lo que busca, sin complicaciones.

Por ejemplo, en un lenguaje tipo SQL, se pueden declarar registros como sigue:

create table *cuenta*

(*número-cuenta* **char**(10),

saldo **integer**)

En el nivel físico, puedes imaginar que un registro de cliente, cuenta o empleado es como un archivo guardado en cajones consecutivos dentro de un archivo físico, solo que aquí esas “posiciones” son espacios en la memoria de la computadora, como palabras o bytes. Lo interesante es que, aunque todo esto sucede detrás de bambalinas, los programadores no tienen que preocuparse por esos detalles técnicos, porque el compilador del lenguaje se encarga de manejarlos y esconderlos.

Al pasar al nivel lógico, cada registro de este tipo se define con una especie de “molde” o plantilla, como el ejemplo de código que vimos antes. Aquí, también se establecen las relaciones entre los diferentes tipos de registros. Si eres programador y usas algún lenguaje de programación, este es el nivel en el que normalmente trabajas, ya que te permite pensar en términos de la información y cómo se conecta, sin preocuparte por cómo se guarda exactamente en la máquina.

De manera similar, en el nivel de vistas se crean diferentes “ventanas” o accesos personalizados a la base de datos para distintos usuarios. Así, cada quien ve solo lo que necesita ver y nada más. Estas vistas no solo simplifican el acceso, también sirven como una especie de filtro de seguridad: evitan que los usuarios lleguen a información confidencial o que no les corresponde. Por ejemplo, los cajeros en un banco solo pueden ver los datos de las cuentas de los clientes, pero no tienen acceso a información como sueldos de los empleados. Es una forma práctica de mantener la información segura y ordenada, mostrando a cada persona solo lo que realmente le interesa.

1.4 Modelos de bases de datos

Para mostrar el concepto de un modelo de datos, describimos dos de ellos en este apartado: el modelo entidad relación y el modelo relacional.

1. Modelo entidad-relación

El modelo de datos entidad-relación (E-R) es una manera de organizar y entender la información inspirada en cómo vemos el mundo cotidiano. Imagina que tienes un grupo de cosas importantes, a las que llamamos entidades, y que esas cosas se conectan entre sí de diferentes formas. Por ejemplo, cada persona sería una entidad, igual que una cuenta bancaria.

Para guardar la información de cada entidad en una base de datos, se usan atributos, que son como las características que describen a cada cosa. Por ejemplo, una cuenta de banco puede tener atributos como el número de cuenta y el saldo, mientras que un cliente puede tener nombre, calle y ciudad. Además, hay atributos especiales, como el id-cliente, que sirve para distinguir a cada persona, porque podría haber dos clientes con el mismo nombre y dirección.

Las relaciones en este modelo son como los lazos que unen a las entidades. Por ejemplo, la relación "impositor" conecta a cada cliente con las cuentas que tiene en

el banco. Así, no solo se sabe quién es quién, sino también cómo se relacionan entre sí.

Todo este conjunto de entidades y relaciones puede representarse de manera visual en un diagrama E-R, que te ayuda a ver cómo está organizada la información y cómo se conectan las diferentes partes de la base de datos. Es una forma sencilla de entender y diseñar bases de datos, pensando en las cosas y las relaciones que vemos en la vida real.

• Rectángulos, que representan conjuntos de entidades.

• Elipses, que representan atributos.

• Rombos, que representan relaciones entre conjuntos de entidades.

• Líneas, que unen los atributos con los conjuntos de entidades y los conjuntos de entidades con las relaciones.

El modelo entidad-relación se utiliza habitualmente en el proceso de diseño de bases de datos.

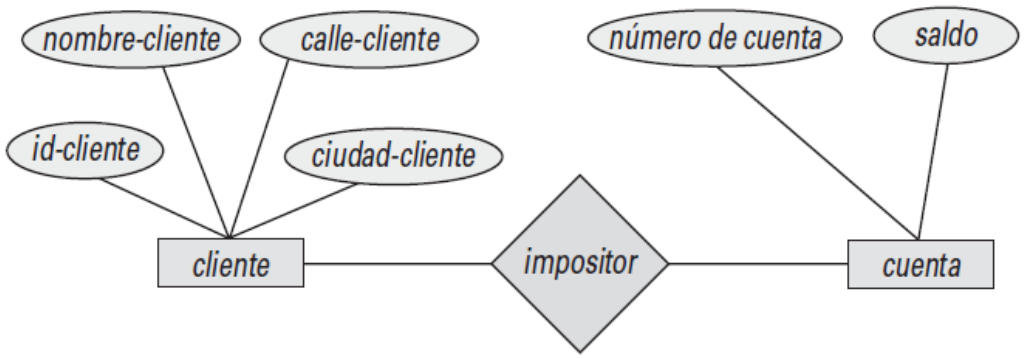

Figura 1 Modelo entidad relación

2. Modelo relacional

En el modelo relacional, la información se organiza en varias tablas, parecidas a hojas de cálculo, donde cada una representa algo distinto y las relaciones entre esas cosas. Cada tabla tiene columnas con nombres únicos, como si fueran etiquetas que nos ayudan a saber qué dato va en cada lugar.

Imagínate una base de datos de un banco: podríamos tener tres tablas principales. La primera es la de clientes, donde aparece, por ejemplo, que el cliente con el ID 19283746 se llama González y vive en la calle Arenal, en La Granja. La segunda tabla es la de cuentas, en la que vemos que la cuenta C-101 tiene un saldo de 500 dólares y la cuenta C-201 tiene 900 dólares.

La tercera tabla nos muestra qué cuentas pertenecen a cada cliente. Por ejemplo, la cuenta C-101 es de González (ID 19283746), mientras que la cuenta C-201 la comparten González y Gómez (ID 01928374), quizá porque tienen un negocio juntos.

Este modelo relacional es el más común y usado en el mundo de las bases de datos; la mayoría de los sistemas actuales funcionan así porque es práctico y muy eficiente para organizar todo tipo de información.

id-cliente	*nombre-cliente*	*calle-cliente*	*ciudad-cliente*
19283746	González	Arenal	La Granja
01928374	Gómez	Carretas	Cerceda
67789901	López	Mayor	Peguerinos
18273609	Abril	Preciados	Valsaín
32112312	Santos	Mayor	Peguerinos
33666999	Rupérez	Ramblas	León
01928374	Gómez	Carretas	Cerceda

(a) La tabla *cliente*

número-cuenta	*saldo*
C-101	500
C-215	700
C-102	400
C-305	350
C-201	900
C-217	750
C-222	700

(b) La tabla *cuenta*

id-cliente	*número-cuenta*
19283746	C-101
19283746	C-201
01928374	C-215
67789901	C-102
18273609	C-305
32112312	C-217
33666999	C-222
01928374	C-201

(b) La tabla *impositor*

Figura 2 Ejemplo de Tablas Relacionales

1.5 Clasificación de Bases de Datos

Las bases de datos se pueden entender y clasificar de muchas formas. Todo depende de cómo se organizan los datos, de si la información cambia mucho o poco, y de cómo están hechas por dentro. La manera más habitual de clasificarlas es según su modelo de datos, es decir, cómo acomodan y estructuran la información.

Tipos de bases de datos

Bases de datos relacionales: Estas son las más conocidas y usadas. Imagina que guardan toda la información en tablas, como si fueran hojas de Excel, con filas y columnas. Además, usan llaves para conectar una tabla con otra. Algunos ejemplos que seguro has escuchado son MySQL, PostgreSQL, Oracle, SQL Server o Sybase. Se usan mucho para transacciones que necesitan ser muy precisas y confiables, como en bancos o sistemas de inventarios.

Bases de datos no relacionales (NoSQL): Este tipo es más flexible y no se basa en tablas. Son ideales para guardar grandes cantidades de datos que no tienen una estructura fija, como documentos, gráficos o pares clave-valor. Ejemplos populares son MongoDB y Cassandra.

Estas bases de datos NoSQL se dividen en varios tipos:

- Clave-Valor: Guardan la información en pares, como si fueran palabras clave y su significado. Son muy rápidas. Ejemplo: Redis.
- Orientadas a documentos: Aquí los datos se almacenan en documentos que pueden tener distintos formatos, como JSON o BSON. Son perfectas para apps donde la información cambia seguido. Ejemplo: MongoDB.
- Orientadas a columnas: Se especializan en organizar la información por columnas, lo que las vuelve muy útiles para analizar grandes volúmenes de datos. Ejemplo: Cassandra.
- Orientadas a grafos: Usan nodos y conexiones para representar datos y relaciones. Son ideales para redes sociales, sistemas de recomendaciones o detectar fraudes. Ejemplo: Neo4j.

Bases de datos jerárquicas: Estas organizan la información como si fuera un árbol genealógico, donde hay registros padres y registros hijos. El detalle es que cada hijo sólo puede tener un padre, lo que las hace algo rígidas y ya no se usan tanto.

Bases de datos de red: Son una evolución del modelo anterior. Aquí, un hijo puede tener varios padres, como si en vez de un árbol tuvieras una red. Esto permite manejar relaciones más complejas, pero también las hace más difíciles de administrar, por eso no son tan populares como las relacionales.

Clasificación según la variabilidad

Bases de datos estáticas: Son como grandes archivos históricos. La información se guarda, casi no se cambia y se consulta muchas veces, por ejemplo, para analizar tendencias o tomar decisiones en empresas. Se usan sobre todo en sistemas de almacenamiento de datos (data warehousing).

Bases de datos dinámicas: Aquí la información está en constante movimiento: se actualiza, se modifica y se borra todo el tiempo. Son la base de la mayoría de los sistemas transaccionales, como los que usan los supermercados para el inventario, las páginas web o las aplicaciones móviles.

Clasificación según la arquitectura

Bases de datos centralizadas: Toda la información está guardada en un solo lugar, como si fuera la bóveda de un banco. Son fáciles de manejar, pero si algo le pasa a ese lugar, todo se cae. No son lo mejor para empresas que manejan mucha información o que están en diferentes lugares.

Bases de datos distribuidas: Aquí los datos están repartidos en varios lugares, pueden ser diferentes servidores o centros de datos que se comunican entre sí. Esto ayuda a que el sistema sea más rápido, seguro y pueda crecer sin problemas, porque el trabajo se reparte. Ejemplo: Cassandra o las bases de datos que están en la nube.

1.6 Arquitectura de base de datos

Se describe la arquitectura de un Sistema Gestor de Bases de Datos y las partes que lo componen.

Un sistema gestor de bases de datos (SGBD) es una colección de datos interrelacionados y un conjunto de programas para acceder a esos datos, según Silberschatz A., Korth H. y Sudarshan S. (2006).

Arquitectura de un Sistema Gestor de Bases de Datos

A continuación, se describe cómo es su **arquitectura**. Podemos ver sus componentes y más adelante se explicarán algunos de ellos.

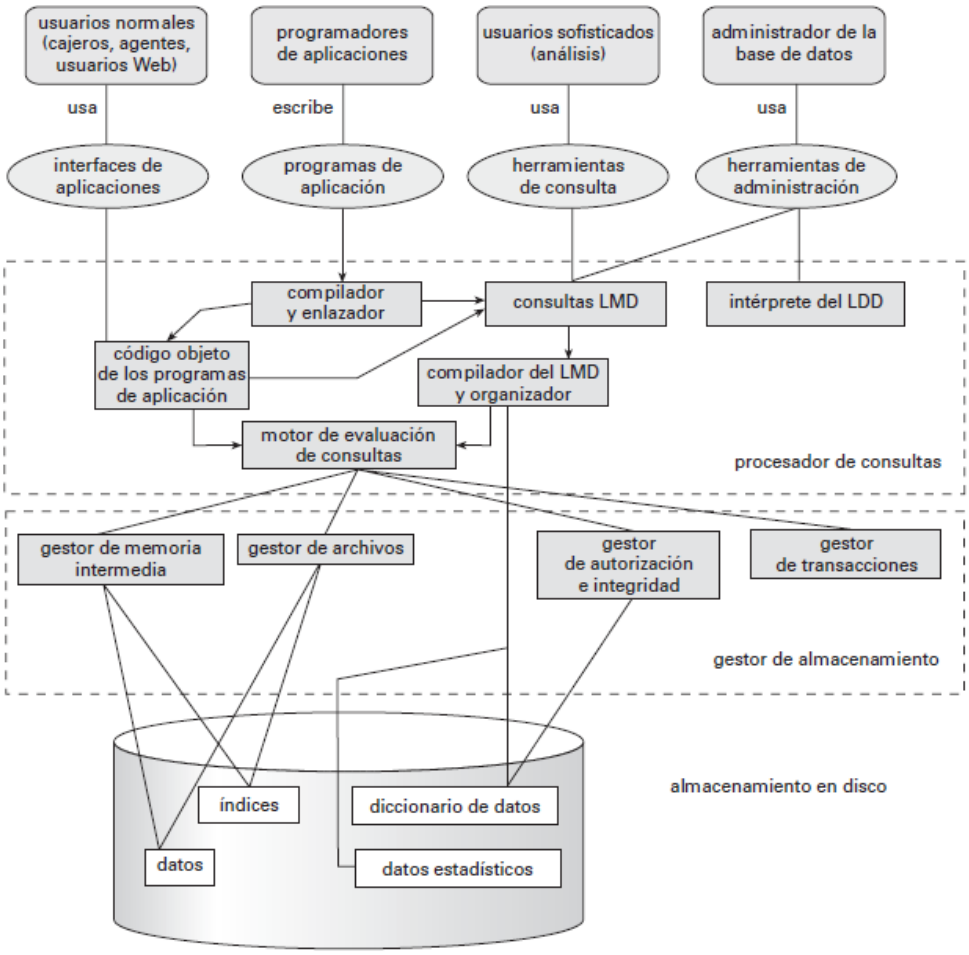

Figura 3 Arquitectura

Un sistema de bases de datos te da las herramientas para crear y modificar la estructura donde se va a guardar la información, así como para hacer consultas y cambios en esa misma información. Por lo general, no necesitas aprender dos lenguajes diferentes para esto: ya que la mayoría, como SQL, te permite definir y manipular los datos usando el mismo conjunto de instrucciones.

Cuando hablamos de manipulación de datos nos referimos a varias acciones básicas que puedes realizar en la base de datos, como:

- Buscar y recuperar información que ya está guardada.
- Agregar datos nuevos.
- Borrar información que ya no necesitas.
- Modificar datos existentes para actualizarlos o corregirlos.

El lenguaje de manipulación de datos, también conocido como LMD, es justo el que te permite hacer todo esto. Es la manera en que los usuarios pueden interactuar con los datos, ya sea para consultar algo específico o para hacer ajustes. De hecho, cuando alguien hace una consulta está pidiendo, a través de una instrucción, que le regrese cierta información. Por eso, muchas veces se usan los términos "lenguaje de consultas" y "lenguaje de manipulación de datos" como si fueran lo mismo, aunque no sea estrictamente correcto.

Por ejemplo, si quieres saber el nombre de un cliente usando su número de identificación, podrías hacer una consulta en SQL que se ve así:

select *cliente.nombre-cliente*

from *cliente*

where *cliente.id-cliente* = ‘19283746’

USUARIOS DE LA BASE DE DATOS

Uno de los propósitos más importantes de un sistema de bases de datos es poder consultar información y guardar datos nuevos de manera sencilla. Las personas que usan una base de datos suelen dividirse en dos grandes grupos: los usuarios, que normalmente acceden para buscar o modificar datos, y los administradores, que se encargan de mantener todo funcionando y seguro.

Existen cuatro tipos principales de usuarios, dependiendo de cómo prefieren relacionarse con el sistema. Por esta razón, se han creado interfaces específicas para cada grupo, buscando que cada quien pueda interactuar de la manera más cómoda y eficiente posible.

• **Usuarios normales.** Estos usuarios no tienen conocimientos técnicos avanzados, así que su manera de interactuar con el sistema es usando programas de aplicación que ya están listos y preparados para ellos. Por ejemplo, imagina a un cajero bancario que necesita mover $50 dólares de la cuenta A a la cuenta B; lo único que tiene que hacer es abrir el programa de “transferir”, ingresar el monto, la cuenta de origen y la cuenta destino, y el sistema se encarga del resto. Así, el cajero sólo sigue los pasos que le pide el programa, sin preocuparse por lo que pasa detrás de la pantalla.

• **Programadores de aplicaciones**. Son personas expertas en informática que se dedican a crear programas para facilitar el trabajo con la base de datos. Quienes programan aplicaciones pueden escoger entre varias herramientas para diseñar las

pantallas y funciones que usan los usuarios, buscando siempre que sean fáciles de entender y utilizar.

• **Los usuarios sofisticados** Estos usuarios no necesitan programas ya hechos para poder usar la base de datos. En vez de eso, escriben directamente sus propias consultas usando el lenguaje especial de consultas de bases de datos. Cada vez que hacen una consulta, esta se manda al procesador de consultas, que se encarga de traducir las instrucciones del lenguaje de manipulación de datos (LMD) para que el gestor de almacenamiento pueda entenderlas y ejecutarlas correctamente.

• **Usuarios especializados.** Son personas expertas que se involucran de lleno con las bases de datos, creando sus propias aplicaciones especializadas para tareas que no se pueden resolver fácilmente usando los métodos tradicionales de procesamiento de datos. Estos usuarios saben exactamente lo que necesitan y diseñan soluciones a la medida, aprovechando al máximo las capacidades de la base de datos.

CUESTIONARIO TEMA 1

1. **¿Cuál es la diferencia principal entre datos e información según el texto?**

   a) Los datos son siempre numéricos, mientras que la información solo es textual.
   b) Los datos son elementos sueltos sin contexto, y la información es el resultado de organizarlos y darles sentido.

   c) La información es solo un subconjunto pequeño de los datos almacenados en una base de datos.

2. **¿Qué representa una "clave foránea" en una base de datos?**

   a) Un identificador único que no puede repetirse ni quedar vacío.
   b) Una referencia que conecta una tabla con otra para establecer relaciones.
   c) Un atributo que describe una característica de una entidad.

3. **En el modelo entidad-relación, ¿qué símbolo representa las relaciones entre entidades?**

   a) Rectángulos
   b) Elipses
   c) Rombos

4. **¿Cuál es la función principal del Sistema Gestor de Bases de Datos (SGBD)?**
   a) Solo almacenar grandes volúmenes de datos sin permitir modificaciones.
   b) Guardar, organizar, proteger y facilitar el acceso a los datos de manera rápida y sencilla.

   c) Crear interfaces gráficas para que los usuarios normales accedan a los datos.

5. **¿Cuál de las siguientes bases de datos NO pertenece al grupo de bases de datos NoSQL?**

    a) MongoDB
    b) Casandra
    c) MySQL

6. **¿Qué caracteriza a una base de datos jerárquica?**

    a) Los registros hijos pueden tener varios padres, formando una red compleja.
    b) La información se organiza en forma de árbol, donde cada hijo tiene un único padre.

    c) Organiza los datos en tablas con filas y columnas interrelacionadas.

7. **¿Qué nivel de abstracción en bases de datos es responsable de decidir cómo y dónde se almacenan básicamente los datos?**

    a) Nivel lógico
    b) Nivel físico
    c) Nivel de vistas

8. **¿Cuál de los siguientes usuarios de bases de datos escribe directamente consultas en lenguaje de manipulación de datos (LMD)?**
    a) Usuarios normales

    b) Usuarios deseados

    c) Administradores

9. **¿Qué característica distingue a las bases de datos distribuidas?**
    a) Toda la información se guarda en un único lugar físico.

    b) Los datos están repartidos en varios servidores o centros de datos que se comunican entre sí.

    c) Solo se usan para bases de datos estáticos con poca variabilidad.

10. **¿Por qué el modelo relacional es el más utilizado en bases de datos actuales?**
    a) Porque no requieren claves para establecer relaciones entre tablas.
    b) organiza Porque la información en tablas con filas y columnas que permiten relaciones claras y eficientes.

    c) Porque solo funciona con datos estáticos sin cambios frecuentes.

Respuestas correctas:

1. b) Los datos son elementos sueltos sin contexto, y la información es el resultado de organizarlos y darles sentido.
2. b) Una referencia que conecta una tabla con otra para establecer relaciones.

3. c) Rombos
4. b) Guardar, organizar, proteger y facilitar el acceso a los datos de manera rápida y sencilla.
5. c) MySQL
6. b) La información se organiza en forma de árbol, donde cada hijo tiene un único padre.
7. b) Nivel físico
8. b) Usuarios sofisticados
9. b) Los datos están repartidos en varios servidores o centros de datos que se comunican entre sí.
10. b) organiza Porque la información en tablas con filas y columnas que permiten relaciones claras y eficientes.

Tema 2: Diseño Conceptual: El Modelo Entidad-Relación (E-R)

En el tema dos se estudia el proceso de diseño conceptual de las bases de datos aplicando el modelo Entidad – Relación (E-R), como una herramienta para modelar los esquemas en una forma consistente y estandarizada. El docente debe promover que el estudiante elija problemas reales y efectúe un análisis de las reglas de negocio antes de elaborar los diagramas E-R.

El proceso se inicia con la fase de análisis de requerimientos, donde el diseñador debe identificar los datos necesarios y las interacciones entre ellos. La clave de esta etapa es la comprensión profunda de las "reglas de negocio", que dictan cómo se debe manejar la información. Un sistema de gestión de bases de datos (SGBD) para una entidad financiera, por ejemplo, debe adherirse estrictamente a las reglas de negocio que rigen las transacciones. Una vez que los requerimientos se han identificado, se utilizan

diagramas de análisis, como el Diagrama Jerárquico Funcional, el Diagrama de Flujo de Datos y, de manera central para el diseño de bases de datos, el

Modelo Entidad-Relación (E-R). Estos diagramas actúan como una herramienta de modelado conceptual que traduce los requerimientos verbales en una representación visual. La transición de los requerimientos del negocio a un modelo conceptual es el paso fundamental que precede al diseño lógico y físico, demostrando que la fase de análisis es la validación del mundo real antes de comenzar con la implementación técnica.

2.1. Conceptos Fundamentales: La Abstracción de la Realidad

El Modelo Entidad-Relación (E-R) es una herramienta de modelado de datos que permite organizar y visualizar la estructura de la información antes de crear la base de datos definitiva.[5] Propuesto por Peter Chen en 1976, este modelo se basa en una percepción del mundo real que lo conceptualiza como una colección de objetos básicos, denominados entidades, y de asociaciones o relaciones entre ellos.

La principal fortaleza del modelo E-R radica en su naturaleza intuitiva y visual, lo que lo convierte en un puente de comunicación esencial entre el diseñador de la base de datos y los usuarios finales o clientes, que a menudo carecen de un profundo conocimiento técnico. Los diagramas E-R, que utilizan una notación gráfica estándar —rectángulos para entidades, rombos para relaciones y elipses para atributos —, permiten que un requerimiento complejo, como "un cliente puede hacer múltiples pedidos", se traduzca visualmente en dos entidades (CLIENTE y PEDIDO) unidas por una relación (HACE). Esta representación compartida ayuda a evitar malentendidos y a validar el diseño conceptual con las partes interesadas antes de invertir recursos en la implementación, lo que se alinea con la visión del TecNM de formar profesionales con capacidad para trabajar en equipos multidisciplinarios.

2.2. Componentes del Modelo E-R: Un Anexo Detallado

2.2.1. Entidades: Los Sustantivos del Modelo

Una entidad representa un objeto, una cosa o un concepto del mundo real que es distinguible de otros objetos, incluso si son del mismo tipo. Por ejemplo, en un sistema de una librería, AUTOR y LIBRO son entidades, ya que se recopila información sobre ellas. La distinción crucial en el modelado E-R es entre entidades fuertes y entidades débiles.

- Entidades Fuertes o Regulares: Son aquellas que tienen existencia por sí mismas. Su identificación es independiente de otras entidades y poseen atributos que las identifican de forma única. Se representan gráficamente con un rectángulo simple. Un ejemplo claro es la entidad DOCTOR en una base de datos de un hospital, cuya existencia no depende de la de un PACIENTE.
- Entidades Débiles: Su existencia o identificación depende de la existencia de otra entidad, a la que se asocian. Se representan con un doble rectángulo. El ejemplo de un AULA y un EDIFICIO ilustra esta dependencia. Un AULA puede tener el mismo número que otra en un edificio diferente, por lo que para identificar un aula de forma única, es necesario conocer también su EDIFICIO. Esta dependencia en identificación tiene una implicación directa en el diseño lógico de la base de datos. La clave primaria de la entidad fuerte (EDIFICIO) se migrará a la tabla de la entidad débil (AULA) para formar parte de su clave primaria, sirviendo como una clave foránea que establece el vínculo y garantiza la unicidad del registro.

2.2.2. Atributos: Las Propiedades de las Entidades

Los atributos son las características o propiedades que describen a una entidad o una relación. En el futuro, estos atributos se convertirán en las columnas de las tablas del modelo relacional. Los atributos se clasifican según su naturaleza, lo que tiene consecuencias directas en el diseño del esquema lógico.

- Simples (Atómicos): Son valores indivisibles. La Primera Forma Normal (1FN) exige que todos los atributos sean atómicos para evitar dificultades en la manipulación y recuperación de datos. Un DNI es un ejemplo de atributo simple.
- Compuestos: Pueden dividirse en subpartes con significado propio. Una Dirección es un atributo compuesto que se puede dividir en Calle, Número, Ciudad, etc.
- Multivaluados: Un atributo puede tener múltiples valores para una sola entidad. Un Teléfono es un ejemplo clásico, ya que un cliente puede tener varios números.

- Derivados: Su valor se calcula a partir de otros atributos o de un proceso. La Edad se deriva de la Fecha de nacimiento. Los atributos derivados no deben almacenarse para evitar redundancia y anomalías de actualización.
- Clave: Un atributo clave, también conocido como identificador, es aquel que identifica de forma única cada ocurrencia de una entidad.

La correcta identificación de los tipos de atributos es fundamental para el proceso de normalización. Un atributo multivaluado o compuesto, si se almacena en una sola columna, viola la 1FN. Para resolverlo, es necesario crear una nueva tabla que contenga el atributo multivaluado y la clave primaria de la entidad original como clave foránea, un paso de refinamiento que demuestra la interdependencia entre el diseño conceptual y el lógico.

A continuación, se presenta una tabla que resume la clasificación de los atributos.

Tabla 1 Clasificación de atributos

Tipo de Atributo	Símbolo de Chen	Descripción	Ejemplo
Simple	Elipse	No puede dividirse en partes más pequeñas.	DNI, Nombre
Compuesto	Elipse conectada a sub-elipses	Se puede descomponer en atributos con significado propio.	Dirección (con sub-atributos Calle, Ciudad, CP)
Multivaluado	Doble elipse	Puede contener múltiples valores para una única entidad.	Teléfono
Derivado	Elipse punteada	Se calcula a partir de otro atributo y no debe almacenarse.	Edad (derivada de Fecha de nacimiento)
Clave	Elipse con nombre subrayado	Identifica de forma única una ocurrencia de una entidad.	ID_Cliente

2.2.3. Relaciones: Los Vínculos entre Entidades

Una relación es la asociación o vinculación entre dos o más entidades. Su función es describir cómo interactúan los objetos del mundo real. Las relaciones se clasifican según su

grado y su cardinalidad.

- Grado: El grado se refiere al número de entidades que participan en una relación.
  - Unaria (grado 1): Una relación consigo misma (ej., EMPLEADO supervisa a EMPLEADO).
  - Binaria (grado 2): Relaciona dos entidades (ej., PROVEEDOR suministra ARTÍCULOS).
  - Ternaria (grado 3): Vincula tres o más entidades al mismo tiempo (ej., un CLIENTE de un BANCO tiene CUENTAS en una SUCURSAL).
- Cardinalidad: La cardinalidad, también conocida como multiplicidad, define la cantidad de veces que las ocurrencias de una entidad se relacionan con las ocurrencias de otra. La cardinalidad máxima es el factor principal para resolver relaciones en el modelo relacional.
  - Uno a Uno (1:1): Cada ocurrencia de una entidad se relaciona con una única ocurrencia de otra. Es poco común, pero se da en casos muy puntuales donde una entidad depende exclusivamente de otra, como en las entidades débiles.
  - Uno a Muchos (1:N): Una ocurrencia de una entidad se relaciona con muchas de la otra, pero cada ocurrencia de la segunda entidad se relaciona con una sola de la primera. Esta es la relación más común y se resuelve en el modelo relacional agregando una clave foránea en la tabla del lado del "muchos".
  - Muchos a Muchos (N:M): Múltiples ocurrencias de una entidad se relacionan con múltiples ocurrencias de otra, y viceversa. Esta cardinalidad es fundamental y se resuelve creando una nueva tabla intermedia que contiene las claves foráneas de ambas entidades, y cuya clave primaria se forma por la combinación de ambas claves foráneas.

La cardinalidad es un factor determinante para el diseñador de bases de datos, ya que un error en su definición puede llevar a un diseño caótico. El análisis de la cardinalidad de una relación N:M, como ESTUDIANTE y CURSO, lleva a la conclusión de que no se puede almacenar la ID_Estudiante en la tabla CURSO o viceversa sin generar redundancia y violar la 1FN. La solución correcta es crear una tabla INSCRIPCION que sirva como tabla de unión, lo que demuestra cómo la lógica del mundo real se traduce en una estructura de base de datos específica y normalizada.

Ejemplos:

Problema 1:

Imagina que tienes una pequeña empresa y necesitas organizar toda la información de tus clientes, artículos y pedidos de una manera sencilla y clara. Hasta ahora, toda esa información está en diferentes documentos, lo cual puede resultar confuso y complicado de manejar. Por eso, lo ideal es contar con una base de datos que te ayude a tener todo en orden.

Por ejemplo, para cada cliente, debes registrar un número único que lo identifique, sus diferentes direcciones de envío, el saldo que tiene, el límite de crédito (que nunca debe superar los 3,000,000 dólares), y el descuento que puede recibir. Así, puedes llevar el control de cuánto puede comprar, dónde enviar sus pedidos y qué beneficios puede aprovechar.

En cuanto a los artículos, es importante saber su número único, las fábricas que los distribuyen y cuántos hay disponibles en cada fábrica. También necesitas guardar una descripción de cada artículo para saber exactamente qué ofreces.

Cuando llega el momento de hacer un pedido, este se divide en dos partes: la cabecera y el cuerpo. En la cabecera, se anotan el número de cliente, la dirección de envío y la fecha y hora del pedido. El cuerpo del pedido incluye varias líneas, y en cada una se indica qué artículo se pidió y en qué cantidad.

Además, hay que tener información de las fábricas, como su número único y el teléfono de contacto, para poder comunicarse en caso de cualquier necesidad. También es útil saber cuántos artículos en total provee cada fábrica, y considerar posibles fábricas alternativas que puedan ofrecer otros productos estratégicos para la empresa.

Recuerda que una dirección debe incluir el número, calle, colonia y ciudad, y que cada fecha debe tener también la hora para llevar un registro preciso de cada pedido y envío.

Solución:

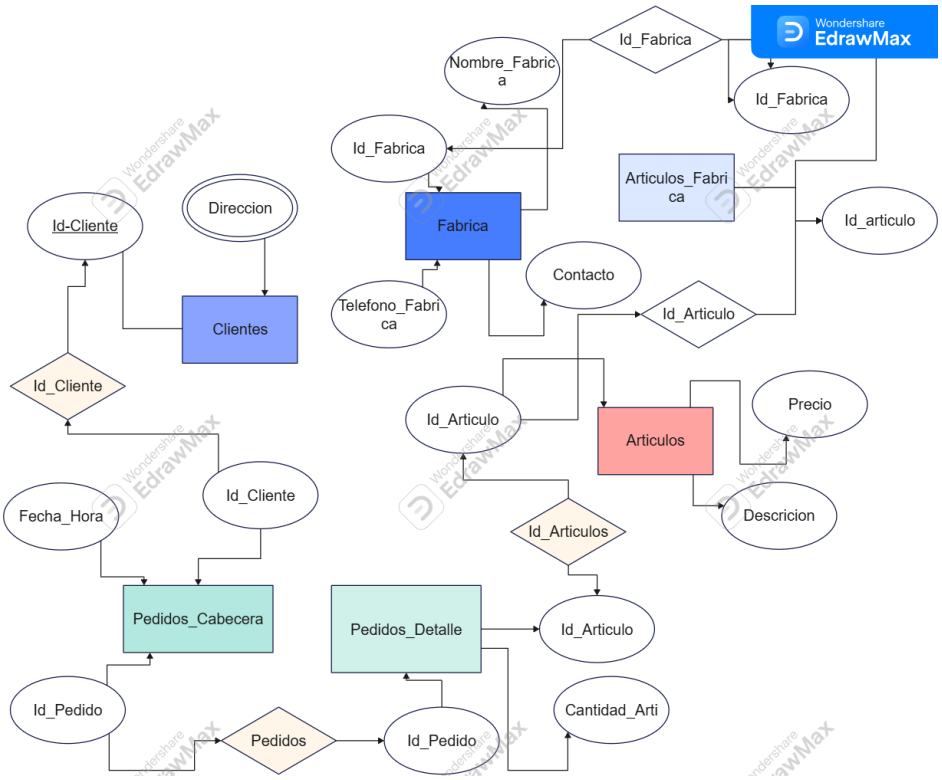

Figura 4 Solución problema 1

Problema 2:

Imagina que te contratan para diseñar una base de datos que ayude a llevar el control de ventas en una empresa. El objetivo es tener toda la información organizada y fácil de consultar, desde los proveedores hasta los clientes, productos y cada venta realizada.

Por ejemplo, para los proveedores, será necesario registrar datos como un identificador único, su nombre, dirección completa (incluyendo calle, número, colonia y ciudad), teléfono y página web. Así, siempre tendrás a la mano la información necesaria para comunicarte o hacer pedidos.

En el caso de los clientes, también se les asigna un id y se guarda su nombre y dirección, pero además puedes registrar varios teléfonos de contacto por si necesitas localizarles fácilmente.

Cada producto tiene un id único, su nombre, el precio actual, la cantidad disponible en stock y el nombre del proveedor que lo distribuye. Para facilitar la organización, los productos se agrupan en categorías, y cada uno pertenece solamente a una categoría específica. Las categorías también llevan su propio id, nombre y una breve descripción para identificar rápidamente de qué se trata.

Cuando se realiza una venta, se guarda toda la información importante: un identificador de la venta, la fecha en que se hizo, el cliente que compró, el descuento aplicado y el monto final. Además, para cada producto vendido, se registra el precio al momento de la venta, la cantidad que se vendió y el total de esa línea de producto, lo que ayuda a tener un control detallado y preciso de cada transacción.

Así, con una base de datos bien estructurada, es mucho más sencillo y práctico llevar el control de todo lo que sucede en el sistema de ventas.

Solución:

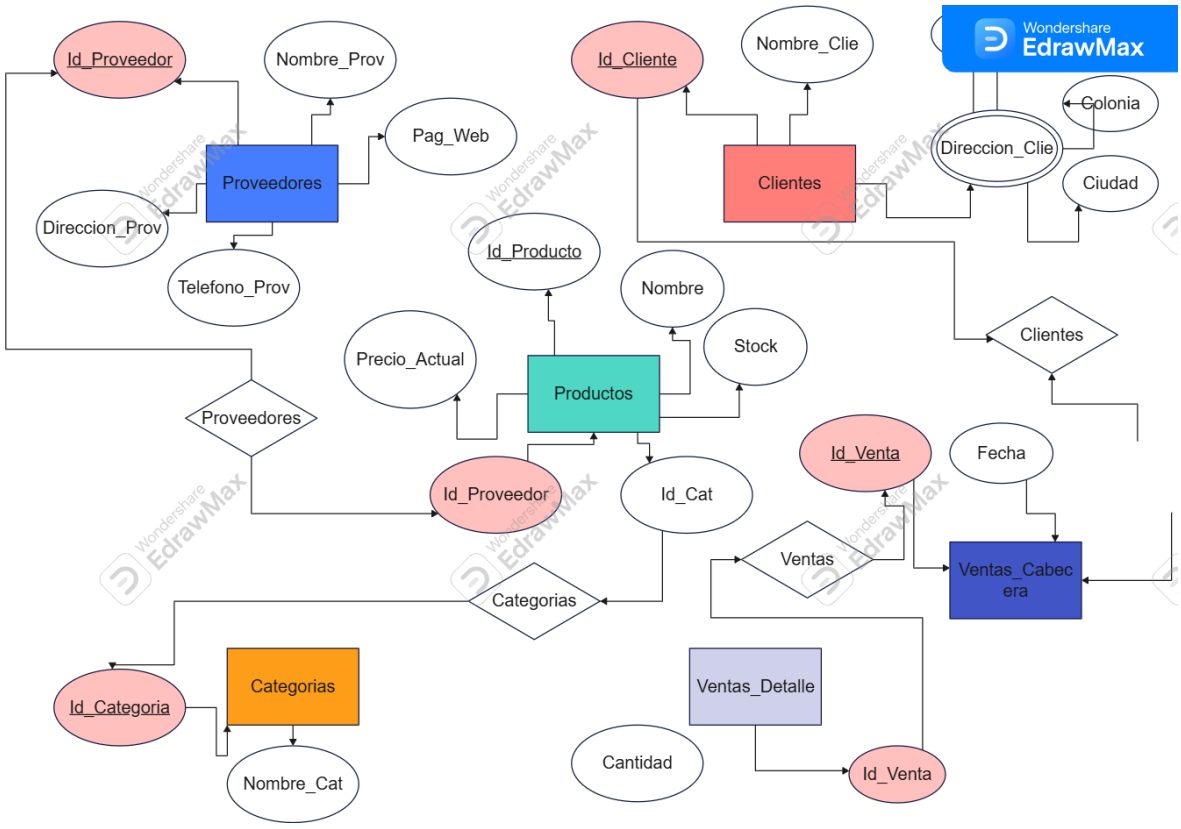

Figura 5 Solución problema 2

CUESTIONARIO TEMA 2

1. **¿Cuál es la fase inicial en el proceso de diseño conceptual según el texto?**

    a) Diseño lógico
    b) Análisis de requisitos
    c) Implementación técnica

2. **¿Qué herramienta es central para el diseño de bases de datos en la fase de análisis?**

    a) Diagrama de Gantt
    b) Diagrama Entidad-Relación (ER)
    c) Diagrama de Pareto

3. **¿Qué representa una entidad en el Modelo Entidad-Relación?**
    a) Una relación entre tablas

    b) Un objeto o concepto del mundo real distinguible

    c) Un atributo compuesto

4. **¿Cómo se representa gráficamente una entidad débil en un diagrama ER?**
    a) Con un rectángulo simple

    b) Con un doble rectángulo

    c) Con un rombo

5. **¿Cuál de los siguientes es un ejemplo de atributo derivado?**
    a) DNI

    b) Edad

    c) Teléfono

6. **¿Qué requisito establece la Primera Forma Normal (1FN) respecto a los atributos?**
    a) Los atributos deben ser multivaluados

    b) Los atributos deben ser atómicos (simples)

    c) Los atributos deben ser derivados

7. **¿Qué tipo de relación tiene un grado ternario?**
    a) Relación entre dos entidades

    b) Relación con una sola entidad

    c) Relación entre tres o más entidades

8. **¿Cómo se resuelve una relación de cardinalidad muchos a muchos (N:M) en el modelo relacional?**

a) Añadiendo una clave foránea en una de las tablas
b) Creando una nueva tabla intermedia con claves foráneas de ambas entidades
c) Eliminando una de las entidades para simplificar

9. **En el ejemplo del aula y edificio, ¿qué sucede con la clave primaria del edificio?**

    a) Se ignora en el diseño de aulas
    b) Se incluye como clave foránea en la entidad débil aula
    c) Se convierte en un atributo derivado

10. **¿Cuál es la función principal de los diagramas ER según el texto?**
    a) Automatizar la creación de bases de datos

    b) Traducir los requerimientos verbales en una representación visual para evitar malentendidos

    c) Ejecutar consultas complejas

Respuestas correctas:

1. b) Análisis de requerimientos
2. b) Diagrama Entidad-Relación (ER)
3. b) Un objeto o concepto del mundo real distinguible
4. b) Con un doble rectángulo
5. b) Edad
6. b) Los atributos deben ser atómicos (simples)
7. c) Relación entre tres o más entidades
8. b) Creando una nueva tabla intermedia con claves foráneas de ambas entidades
9. b) Se incluye como clave foránea en la entidad débil aula
10. b) Traducir los requerimientos verbales en una representación visual para evitar malentendidos

Tema 3 El Modelo Relacional

En este, se conoce y comprende su estructura, elementos que lo conforman y sus reglas de integridad. El docente deberá propiciar que el estudiante identifique la relación que existe entre el modelo E-R y el modelo relacional.

3.1. Transición del Modelo E-R al Modelo Relacional

El modelo E-R es el anteproyecto conceptual de una base de datos. Una vez que este diagrama ha sido validado, el siguiente paso es transformarlo en el modelo relacional. Este proceso de mapeo es una aplicación de reglas formales que garantizan que el diseño lógico sea coherente, sin pérdida de información, y refleje la estructura del mundo real modelado. En esta transición, las entidades se convierten en

tablas (también llamadas relaciones), los atributos simples se transforman en columnas y los atributos clave en claves primarias.

Las relaciones entre entidades se resuelven mediante la implementación de claves foráneas. Un atributo multivaluado o una relación N:M, que violan la 1FN en el modelo conceptual, se resuelven creando tablas separadas en el modelo relacional. Por ejemplo, un atributo

Teléfonos multivaluado en una entidad CLIENTE se convierte en una nueva tabla TELEFONOS, que contendrá el ID_Cliente (como clave foránea) y el Número_Teléfono. De manera similar, una relación N:M entre ESTUDIANTE y CURSO se mapea a tres tablas: ESTUDIANTE, CURSO y una tabla intermedia INSCRIPCION que contiene las claves primarias de las dos primeras como claves foráneas.[14] Este proceso demuestra que el diseño relacional es el resultado directo del análisis conceptual y que las decisiones tomadas en la primera fase tienen un impacto directo en la estructura final de la base de datos.

3.2. Pilares del Modelo Relacional: La Relación, la Tupla y el Atributo

El modelo relacional, propuesto por Edgar F. Codd, estructura la información de manera simple y lógica en tablas. Estas tablas se conocen formalmente como

relaciones y consisten en filas, denominadas tuplas, y columnas, conocidas como atributos.

- Relación (Tabla): Es un conjunto de tuplas. La tabla es la representación visual de una relación.
- Tupla (Fila): Es un registro, o una colección de atributos ordenados que representan una ocurrencia de una entidad. En una base de datos relacional, cada tupla es identificable de manera única por sus atributos.
- Atributo (Columna): Es una propiedad que describe la tupla. Cada atributo tiene un dominio, que es el conjunto de valores permitidos para esa columna. El concepto de dominio es una regla de integridad que va más allá del tipo

de dato, restringiendo los valores a un rango (p. ej., precios entre 10 y 1000) o a un conjunto de valores codificados (p.ej., Pavimento, Grava, Arena para el tipo de superficie). La definición de dominios previene errores de entrada y mejora la consistencia de los datos, contribuyendo a la integridad de la base de datos.

3.3. Las 12 Reglas de Codd: Los Mandamientos del Modelo Relacional

En 1985, Edgar F. Codd publicó un conjunto de 13 reglas (numeradas del 0 al 12) para definir los requisitos de un verdadero SGBD relacional. Aunque muy pocos sistemas modernos cumplen rigurosamente con todas ellas, se considera que un SGBD es "más relacional" cuanto más se adhiere a estos principios. Estas reglas no son una simple lista de verificación, sino que representan la justificación teórica de la superioridad del modelo relacional sobre los modelos de datos jerárquicos y de red.

Algunas de las reglas más importantes son:

- Regla 0 (Regla Fundamental): Un sistema que se proclame como relacional debe gestionar las bases de datos exclusivamente a través de sus capacidades relacionales. Esto evita la subversión del modelo, un problema común en los sistemas heredados que añadían una capa relacional sobre una estructura no relacional preexistente.
- Regla 1 (Regla de la Información): Toda la información debe ser representada explícitamente y de una única manera: mediante valores en tablas.
- Regla 2 (Regla del Acceso Garantizado): Cada valor atómico es lógicamente accesible de manera inequívoca a través de una combinación de nombre de tabla, valor de clave primaria y nombre de columna. Esto subraya la importancia de las claves primarias como identificadores únicos de cada registro.
- Regla 9 (Independencia Lógica de los Datos): La estructura lógica de la base de datos puede modificarse (p. ej., agregando una nueva columna a una tabla) sin que los programas de aplicación existentes se vean afectados. Este principio es una mejora clave con respecto a los modelos jerárquicos y de red, que dependían de punteros rígidos y hacía que el mantenimiento fuera extremadamente difícil. La independencia lógica permite una evolución continua del diseño de la base de datos sin romper el software que la utiliza, un principio de ingeniería de software fundamental que Codd anticipó.
- Regla 11 (Independencia de la Distribución): La distribución de los datos en múltiples ubicaciones debe ser transparente para el usuario final, que debe tener la impresión de que los datos se encuentran en un solo lugar.

Las reglas de Codd no solo definen el modelo relacional, sino que también justifican su solidez, eficiencia y mantenibilidad a largo plazo.

CUESTIONARIO TEMA 3

1. **¿Cuál es el siguiente paso una vez validado el diagrama del modelo ER?**
   a) Crear un modelo jerárquico

   b) Transformarlo en el modelo relacional

   c) Implementar directamente la base de datos físicos

2. **En la transición del modelo ER al relacional, ¿qué representan las entidades?**
   a) Claves primarias

   b) Tablas o relaciones

   c) Atributos multivaluados

3. **¿Cómo se resuelve una relación N:M en el modelo relacional?**

   a) Creando una tabla intermedia que contenga claves foráneas de ambas entidades

   b) Eliminando una de las entidades para evitar la relación

   c) Fusionando las dos entidades en una sola tabla

4. **¿Qué es una tupla en el modelo relacional?**

   a) Una columna que contiene propiedades
   b) Una fila o registro que representa una ocurrencia de entidad
   c) Un conjunto de tablas relacionadas

5. **¿Qué función cumple el dominio de un atributo?**

   a) Definir el nombre de la tabla
   b) Restringir los valores permitidos para esa columna
   c) Establecer la clave primaria de la tabla

6. **Según las reglas de Codd, ¿qué establece la Regla 0 (Regla Fundamental)?**

   a) El sistema debe gestionar bases de datos con capas no relacionales
   b) El sistema debe gestionar bases de datos exclusivamente a través de capacidades relacionales
   c) El sistema debe permitir el acceso mediante punteros rígidos

7. **¿Qué implica la Regla 2 (Regla del Acceso Garantizado) de Codd?**

   a) Que cada valor atómico es accesible lógicamente mediante tabla, clave primaria y columna.
   b) Que solo se pueden acceder a los valores mediante consultas complejas
   c) Que las claves primarias no son necesarias para la identificación única

8. ¿Cuál es el beneficio principal de la Regla 9 (Independencia Lógica de los Datos)?

a) Permite que la estructura lógica se modifique sin afectar los programas existentes
b) Obliga a modificar todos los programas cuando se cambia la estructura de la base de datos
c) Restringe la adición de nuevas columnas para evitar errores

9. ¿Qué describe la Regla 11 (Independencia de la Distribución) de Codd?

a) Que los datos deben estar almacenados en una única ubicación física
b) Que la distribución de datos en Múltiples ubicaciones debe ser transparente para el usuario final
c) Que los usuarios deben conocer la ubicación física de los datos para acceder a ellos

10. ¿Cuál es el principal propósito del proceso de mapeo del modelo ER al modelo relacional?

a) Simplificar el diseño sin preocuparse por la pérdida de información
b) Garantizar coherencia, sin pérdida de información, y reflejar la estructura del mundo real
c) Implementar atributos multivaluados directamente en una única tabla

Respuestas correctas:

1. b) Transformarlo en el modelo relacional
2. b) Tablas o relaciones
3. a) Creando una tabla intermedia que contenga claves foráneas de ambas entidades
4. b) Una fila o registro que representa una ocurrencia de entidad
5. b) Restringir los valores permitidos para esa columna
6. b) El sistema debe gestionar bases de datos exclusivamente a través de capacidades relacionales
7. a) Que cada valor atómico es accesible lógicamente mediante tabla, clave primaria y columna
8. a) Permite que la estructura lógica se modifique sin afectar los programas existentes
9. b) Que la distribución de datos en Múltiples ubicaciones debe ser transparente para el usuario final.
10. b) Garantizar coherencia, sin pérdida de información, y reflejar la estructura del mundo real

Tema 4 Normalización de bases de datos.

Aquí, se estudian las formas normales de base de datos que garantizan la integridad de la base de datos y evitan la redundancia de información, contando con la posibilidad de ahondar en otras formas normales como la cuarta y quinta. Se recomienda que el docente proponga ejemplos de entidades para aplicar las reglas de normalización y demostrar claramente la diferencia o diferencias de entidades no normalizadas y normalizadas.

4.1. Conceptos básicos

Las bases de datos relacionales se distinguen por su capacidad de garantizar las Propiedades ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad). Estas propiedades son un conjunto de garantías que aseguran que las transacciones de la base de datos se procesen de manera confiable, lo cual es vital para aplicaciones que requieren una alta precisión, como los sistemas de transacciones financieras.

- Atomicidad: Asegura que una transacción se completa en su totalidad o no se realiza en absoluto. Todos los cambios se aplican o se deshacen por completo.
- Consistencia: Define las reglas para mantener la integridad de los datos después de una transacción. Una transacción debe llevar la base de datos de un estado válido a otro.
- Aislamiento: Mantiene los efectos de las transacciones invisibles para las demás transacciones, como si cada una se ejecutara de forma individual.
- Durabilidad: Garantiza que los cambios de datos de una transacción confirmada son permanentes y persisten incluso en caso de fallos del sistema.

El cumplimiento estricto de ACID es un diferenciador clave del modelo relacional frente a las bases de datos NoSQL. Mientras que las bases de datos relacionales sacrifican algo de escalabilidad y flexibilidad para garantizar la integridad absoluta de las transacciones, las NoSQL priorizan la escalabilidad horizontal y la flexibilidad, utilizando modelos de consistencia más relajados. La elección entre un modelo y otro, por lo tanto, no es una cuestión de superioridad, sino de adecuación a los requerimientos de la aplicación. Un ingeniero de sistemas debe evaluar los requisitos de integridad del negocio para determinar cuándo el cumplimiento de ACID es indispensable (ej., un sistema bancario) y cuándo se puede relajar (ej., un sistema de redes sociales).

4.2. Primera forma normal.

La normalización es un proceso sistemático para simplificar el diseño de una base de datos. Consiste en la aplicación de una serie de reglas para organizar las relaciones, cuyo propósito es evitar la redundancia de datos y las anomalías que pueden ocurrir al insertar, modificar o eliminar información.

4.2.1. Las Formas Normales de Codd (1FN, 2FN, 3FN)

Las formas normales de Codd son un conjunto de reglas acumulativas, donde cada forma normal superior incluye las reglas de las anteriores.

- Primera Forma Normal (1FN): La regla fundamental establece que todos los atributos deben ser atómicos o indivisibles, y no debe haber grupos de valores repetidos dentro de una tabla. La violación de esta regla, por ejemplo, al almacenar múltiples valores de Teléfono en una sola columna, hace que la tabla sea ineficiente y propensa a errores.
- Segunda Forma Normal (2FN): Una tabla debe estar en 1FN y no contener dependencias funcionales parciales. Esto significa que todos los atributos no clave deben depender de la clave primaria completa, y no solo de una parte de ella. Esta regla es especialmente relevante en tablas con claves primarias compuestas.
- Tercera Forma Normal (3FN): Una tabla debe estar en 2FN y no tener dependencias transitivas. Esto implica que ningún atributo no clave debe depender de otro atributo no clave. La 3FN resuelve anomalías de actualización y garantiza que la información se almacene en su lugar más apropiado.

4.2.2. La Forma Normal de Boyce-Codd (BCNF)

La Forma Normal de Boyce-Codd (BCNF) es una versión más estricta de la 3FN, desarrollada para abordar ciertos tipos de anomalías que persisten en la 3FN, especialmente en relaciones con múltiples claves candidatas superpuestas.

La regla de la BCNF es sencilla: para cada dependencia funcional no trivial $X \rightarrow Y$, X debe ser una superclave. La diferencia con la 3FN radica en una excepción que esta última permite: en 3FN, la dependencia $X \rightarrow Y$ es válida si Y es un atributo primo (es parte de alguna clave candidata), incluso si X no es una superclave. La BCNF elimina esta excepción, exigiendo que cualquier determinante sea una superclave, lo que elimina cualquier redundancia residual basada en dependencias funcionales.

La normalización es un proceso de refinamiento que busca erradicar la redundancia. La 3FN es suficiente para la mayoría de las relaciones, pero en casos complejos, la BCNF proporciona una garantía adicional de integridad. Comprender la diferencia entre ambas demuestra un conocimiento profundo del modelo relacional y la capacidad de diseñar esquemas de bases de datos robustos y libres de anomalías.

A continuación, se presenta una tabla que resume las formas normales más relevantes.

Forma Normal	Regla	Problema que Resuelve

1FN	Todos los atributos son atómicos y no hay grupos repetidos.	Anomalías al insertar, modificar y eliminar datos en atributos multivaluados y compuestos.
2FN	1FN + Todos los atributos no clave dependen de la clave primaria completa.	Anomalías al modificar datos que dependen de solo una parte de la clave primaria.
3FN	2FN + No hay dependencias transitivas.	Anomalías al modificar atributos no clave que dependen de otros atributos no clave.
BCNF	Cada determinante es una superclave.	Anomalías en casos con múltiples claves candidatas superpuestas, donde la 3FN no es suficiente.

4.3. El Dilema de la Denormalización

La normalización, si bien es una práctica esencial para la integridad de los datos, puede tener un costo en el rendimiento de las consultas. Al normalizar, se crean más tablas, lo que requiere un mayor número de operaciones de JOIN para recuperar datos completos. Para algunas aplicaciones, especialmente aquellas con grandes volúmenes de datos que priorizan las lecturas rápidas sobre la consistencia transaccional absoluta, este aumento en las uniones puede ser inaceptable.

La denormalización es la práctica intencional de introducir redundancia en el diseño de una base de datos para mejorar el rendimiento de las consultas. Este es un acto de equilibrio entre la integridad de los datos y el rendimiento del sistema. La decisión de desnormalizar no debe tomarse a la ligera, sino que debe estar justificada por un análisis detallado de los requerimientos de la aplicación. Mientras que la normalización es la mejor práctica para garantizar la coherencia a largo plazo, la denormalización puede ser una solución viable para escenarios específicos donde la velocidad es el factor más crítico, como en el análisis de big data en tiempo real. Un ingeniero en sistemas debe sopesar estos factores para diseñar una arquitectura de datos óptima que satisfaga las necesidades del negocio.

CUESTIONARIO TEMA 4

1. **¿Cuál es la propiedad ACID que asegura que una transacción se completa por completo o no se realiza en absoluto?**

    a) Consistencia
    b) Atomicidad
    c) Durabilidad

2. **¿Qué propiedad ACID mantiene los efectos de una transacción invisible para otras transacciones?**

    a) Aislamiento
    b) Atomicidad
    c) Consistencia

3. **¿Cuál es la principal diferencia entre bases de datos relacionales y NoSQL respecto a ACID?**

    a) Las bases de datos NoSQL garantizan ACID estrictamente.
    b) Las bases de datos relacionales sacrifican escalabilidad para garantizar ACID.
    c) Las bases de datos relacionales no cumplen con ACID.

4. **¿Qué forma normal exige que todos los atributos sean atómicos y sin grupos repetidos?**

    a) Segunda Forma Normal (2FN)

    b) Primera Forma Normal (1FN)

    c) Tercera Forma Normal (3FN)

5. **¿Qué problema resuelve la Segunda Forma Normal (2FN)?**

    a) Anomalías causadas por dependencias funcionales parciales.

    b) Anomalías causadas por atributos no atómicos.

    c) Anomalías causadas por dependencias transitivas.

6. **¿Cuál es la condición que debe cumplir una tabla para estar en Tercera Forma Normal (3FN)?**

    a) No tener atributos no clave.
    b) No tener dependencias transitivas entre atributos no clave.
    c) Todos los atributos deben depender de una parte de la clave primaria.

7. **¿Qué diferencia principal existe entre la 3FN y la Forma Normal de Boyce-Codd (BCNF)?**

    a) BCNF permite dependencias funcionales parciales.
    b) BCNF exige que todos los determinantes sean superclaves, mientras que la 3FN permite excepciones.
    c) La 3FN es más estricta que BCNF.

8. **¿Cuál es el principal objetivo del proceso de normalización?**
   a) Mejorar la velocidad de las consultas.

   b) Evitar la redundancia y las anomalías en las operaciones con datos.
   c) Incrementar la cantidad de tablas en la base de datos para mayor complejidad.

9. **¿Qué es la desnormalización en bases de datos?**
   a) El proceso de aplicar todas las formas normales para eliminar redundancias.
   b) Introducir redundancia intencionalmente para mejorar el rendimiento de las consultas.

   c) Eliminar datos duplicados para garantizar la integridad.

10. **¿En qué tipo de aplicaciones puede ser más recomendable la desnormalización?**
    a) En sistemas bancarios que requieran alta integridad.

    b) En análisis de big data en tiempo real donde la velocidad es crítica.
    c) En bases de datos sin requisitos de rendimiento.

Respuestas correctas:

1. b) Atomicidad
2. a) Aislamiento
3. b) Las bases de datos relacionales sacrifican escalabilidad para garantizar ACID.
4. b) Primera Forma Normal (1FN)
5. a) Anomalías causadas por dependencias funcionales parciales.
6. b) No tener dependencias transitivas entre atributos no clave.
7. b) BCNF exige que todos los determinantes sean superclaves, mientras que la 3FN permite excepciones.
8. b) Evitar la redundancia y las anomalías en las operaciones con datos.
9. b) Introducir redundancia intencionalmente para mejorar el rendimiento de las consultas.
10. b) En análisis de big data en tiempo real donde la velocidad es crítica.

Tema 5 Álgebra relacional.

Se conoce y comprende el uso y aplicación del álgebra relacional como lenguaje de consulta formal a base de datos, los operadores básicos y los operadores del álgebra relacional extendida. Se sugiere que el docente realice planteamientos de consulta a base de datos.

Los lenguajes de consulta relacionales son la interfaz para manipular y recuperar datos del modelo relacional. Se dividen en dos categorías principales, que representan dos filosofías de consulta distintas: la procedural y la declarativa.

5.1. Álgebra Relacional: El Lenguaje Procedural

El álgebra relacional funciona como una especie de manual de instrucciones para hacer consultas en bases de datos. Imagínalo como un conjunto de herramientas que puedes usar para seleccionar, combinar o modificar la información que tienes guardada. Por ejemplo, puedes elegir solo los datos que cumplen con cierta condición (selección), mostrar solo algunas columnas específicas (proyección), juntar datos de diferentes tablas (unión), quitar elementos que no te interesan (diferencia de conjuntos), combinar todos los datos posibles entre dos tablas (producto cartesiano) y cambiar el nombre de las columnas o tablas para que sean más fáciles de entender (renombramiento). Además de estas herramientas principales, hay otras opciones como la intersección de conjuntos, la reunión natural, la división y la asignación, que se pueden construir usando las operaciones básicas. Todas estas funciones te ayudan a organizar y recuperar la información de manera eficiente y clara, dependiendo de lo que necesites hacer con tus datos.

Operaciones fundamentales

Las operaciones de selección, proyección y renombramiento se llaman unarias porque trabajan únicamente sobre una sola tabla o relación. Por otro lado, existen otras tres operaciones que involucran dos tablas al mismo tiempo, por lo que se les llama operaciones binarias.

La operación selección

Imagina que tienes una lista muy larga de préstamos y quieres quedarte solo con aquellos que cumplen cierta condición, como que pertenezcan a una sucursal específica. Para esto sirve la selección. Utiliza la letra griega sigma minúscula (σ) y en el subíndice se pone la condición que deben cumplir los registros. Después, entre paréntesis, se escribe la tabla sobre la que se va a buscar. Por ejemplo, si solo te interesan los préstamos gestionados en la sucursal "Navacerrada", la consulta se escribe así: σ nombre-sucursal = «Navacerrada» (préstamo). Así, el sistema te muestra únicamente los préstamos de esa sucursal, filtrando los demás.

número-préstamo	*nombre-sucursal*	*importe*
P-11	Collado Mediano	900
P-14	Centro	1.500
P-15	Navacerrada	1.500
P-16	Navacerrada	1.300
P-17	Centro	1.000
P-23	Moralzarzal	2.000
P-93	Becerril	500

Figura 6

Si la relación préstamo es como se muestra en la Figura 6, la relación que resulta de la consulta anterior es como se muestra en la Figura 7.

número-préstamo	*nombre-sucursal*	*importe*
P-15	Navacerrada	1.500
P-16	Navacerrada	1.300

Figura 7

Supón que quieres encontrar todos los registros de préstamos en los que el importe prestado sea mayor a $1,200 dólares. Para hacer esto, la consulta sería así: σimporte > 1200 (préstamo). Es como pedirle al sistema que te muestre solamente los préstamos que superan esa cantidad.

En este tipo de búsquedas, puedes usar comparaciones con símbolos como =, ≠, o ≥. Además, si necesitas que se cumplan varias condiciones a la vez, puedes combinarlas usando “y” (∧) o “o” (∨). Por ejemplo, si quieres ver únicamente los préstamos de más de $1,200 euros que hayan sido concedidos por la sucursal de Navacerrada, la consulta sería: σnombre-sucursal = «Navacerrada» ∧ importe > 1200 (préstamo). Así filtramos la información justo como la necesitamos.

Por otro lado, imagina que quieres hacer una lista sencilla con solo el número de préstamo y el importe, sin incluir el nombre de la sucursal. Para esto sirve la operación de proyección. Esta herramienta te ayuda a crear una tabla que solo muestre los datos que te interesan, eliminando los repetidos para que la información sea más clara. La proyección se indica con la letra pi mayúscula (Π) y, en el subíndice, pones los atributos que quieres ver en el resultado. Por ejemplo, para obtener la lista mencionada, la consulta sería: Πnúmero-préstamo, importe (préstamo). El resultado es una tabla ordenada con los préstamos y sus importes, tal como se muestra en la Figura 8.

número-préstamo	*importe*
P-11	900
P-14	1.500
P-15	1.500
P-16	1.300
P-17	1.000
P-23	2.000
P-93	500

Figura 8 Composición de operaciones relacionales

Es importante el hecho de que el resultado de una operación relacional sea también una relación. Considérese la consulta más compleja «Encontrar los clientes que viven en Peguerinos». Hay que escribir:

Πnombre-cliente (σciudad-cliente = «Peguerinos» (cliente))

- **La operación unión**

Imagina que quieres saber quiénes son los clientes del banco que tienen una cuenta, un préstamo o incluso ambos. Es importante saber que en la relación llamada “cliente” no aparece esta información, ya que no todos los clientes necesariamente tienen una cuenta o un préstamo con el banco. Para responder a esta pregunta, necesitas revisar dos listas diferentes: una que muestra quiénes tienen cuentas, que se llama “impositor” (como ves en la Figura 9), y otra que muestra quiénes han solicitado préstamos, llamada “prestatario” (como se ve en la Figura 10). Por ejemplo, ya sabemos cómo buscar los nombres de los clientes que han pedido préstamos en el banco:

nombre cliente	*número cuenta*
Abril	C-102
Gómez	C-101
González	C-201
González	C-217
López	C-222
Rupérez	C-215
Santos	C-305

Figura 9

nombre cliente	*número préstamo*
Fernández	P-16
Gómez	P-93
Gómez	P-15
López	P-14
Pérez	P-17
Santos	P-11
Sotoca	P-23
Valdivieso	P-17

Figura 10

Πnombre-cliente (prestatario)

También se conoce la manera de averiguar el nombre de los clientes con cuenta en el banco:

Πnombre-cliente (impositor)

Para responder a esta pregunta, es necesario juntar ambos grupos; en otras palabras, necesitamos todos los nombres de clientes que aparecen en cualquiera de las dos listas, o incluso en ambas. Esto se logra utilizando la operación de unión, que, al igual que en la teoría de conjuntos, se representa con el símbolo ∪. Así que, la expresión que necesitamos usar es la siguiente:

Πnombre-cliente (prestatario) ∪ Πnombre-cliente (impositor)

El resultado de esta consulta se muestra en la Figura 5.6. Si te fijas, aparecen diez filas, aunque en realidad hay siete personas diferentes que han solicitado préstamos y seis que tienen cuentas. Esto puede parecer extraño, pero la razón es que algunos clientes, como Gómez, Santos y López, aparecen en ambas listas porque son tanto prestatarios como impositores. Recuerda que, al tratarse de conjuntos, los datos duplicados se eliminan automáticamente y sólo se cuenta cada cliente una vez.

nombre-cliente
Abril
Fernández
Gómez
González
López
Pérez
Rupérez
Santos
Sotoca
Valdivieso

Figura 11

- **La operación diferencia de conjuntos**

La operación de diferencia de conjuntos, identificada con el símbolo –, nos ayuda a encontrar únicamente las filas que están en una lista, pero no aparecen en la otra. En términos sencillos, si tienes dos grupos de datos y quieres saber qué elementos están solo en el primero y no en el segundo, esta operación es la que necesitas. Por ejemplo, si deseas saber qué clientes del banco tienen una cuenta abierta, pero no han solicitado ningún préstamo, solo debes escribir la siguiente expresión:

Πnombre-cliente (impositor) – Πnombre-cliente (prestatario)

La relación resultante de esta consulta aparece en la Figura 12

nombre-cliente
Abril
González
Rupérez

Figura 12

- **La operación producto cartesiano**

La operación de producto cartesiano, representada por el símbolo ×, sirve para combinar datos de dos tablas distintas. Por ejemplo, si juntas la tabla de “prestatarios” con la de “préstamos”, el resultado será una nueva tabla que muestra todas las combinaciones posibles entre ambas. Es como si tomaras cada persona que ha solicitado un préstamo y la emparejaras con cada préstamo existente, creando una lista gigante de pares posibles.

Ahora bien, puede suceder que tanto la tabla de prestatarios como la de préstamos tengan columnas con el mismo nombre, como “número de préstamo”. Para evitar

confusiones, se acostumbra agregar el nombre de la tabla al principio del nombre de la columna. Por ejemplo: “prestatario.número-préstamo” y “préstamo.número-préstamo”. De esta manera, es fácil saber de dónde viene cada dato. Para los campos que sólo aparecen en una de las tablas, generalmente se omite el prefijo, ya que no hay riesgo de mezclar información. Así, el resultado final de la unión podría tener columnas como: nombre-cliente, prestatario.número-préstamo, nombre-sucursal, préstamo.número-préstamo e importe.

número-préstamo	*nombre-sucursal*	*importe*
P-11	Collado Mediano	900
P-14	Centro	1.500
P-15	Navacerrada	1.500
P-16	Navacerrada	1.300
P-17	Centro	1.000
P-23	Moralzarzal	2.000
P-93	Becerril	500

nombre cliente	*número préstamo*
Fernández	P-16
Gómez	P-93
Gómez	P-15
López	P-14
Pérez	P-17
Santos	P-11
Sotoca	P-23
Valdivieso	P-17

Figura 13

Ya que sabes cómo funciona el esquema de la relación r = prestatario × préstamo, es momento de entender cómo se generan las filas que aparecen en r. Básicamente, por cada combinación posible entre una persona que solicita un préstamo y cada préstamo existente, se crea una fila nueva en r. Es como si tomaras a cada cliente y lo emparejaras con todos los préstamos, uno por uno. Por eso, la relación r puede ser bastante grande, tal como se muestra en la Figura 13, donde solo se presenta una parte de todas esas combinaciones.

nombre-cliente	*prestatario.número-préstamo*	*préstamo.número-préstamo*	*nombre-sucursal*	*importe*
Santos	P-17	P-11	Collado Mediano	900
Santos	P-17	P-14	Centro	1.500
Santos	P-17	P-15	Navacerrada	1.500
Santos	P-17	P-16	Navacerrada	1.300
Santos	P-17	P-17	Centro	1.000
Santos	P-17	P-23	Moralzarzal	2.000
Santos	P-17	P-93	Becerril	500
Gómez	P-23	P-11	Collado Mediano	900
Gómez	P-23	P-14	Centro	1.500
Gómez	P-23	P-15	Navacerrada	1.500
Gómez	P-23	P-16	Navacerrada	1.300
Gómez	P-23	P-17	Centro	1.000
Gómez	P-23	P-23	Moralzarzal	2.000
Gómez	P-23	P-93	Becerril	500
López	P-15	P-11	Collado Mediano	900
López	P-15	P-14	Centro	1.500
López	P-15	P-15	Navacerrada	1.500
López	P-15	P-16	Navacerrada	1.300
López	P-15	P-17	Centro	1.000
López	P-15	P-23	Moralzarzal	2.000
López	P-15	P-93	Becerril	500
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...
Valdivieso	P-17	P-11	Collado Mediano	900
Valdivieso	P-17	P-14	Centro	1.500
Valdivieso	P-17	P-15	Navacerrada	1.500
Valdivieso	P-17	P-16	Navacerrada	1.300
Valdivieso	P-17	P-17	Centro	1.000
Valdivieso	P-17	P-23	Moralzarzal	2.000
Valdivieso	P-17	P-93	Becerril	500
Fernández	P-16	P-11	Collado Mediano	900
Fernández	P-16	P-14	Centro	1.500
Fernández	P-16	P-15	Navacerrada	1.500
Fernández	P-16	P-16	Navacerrada	1.300
Fernández	P-16	P-17	Centro	1.000
Fernández	P-16	P-23	Moralzarzal	2.000
Fernández	P-16	P-93	Becerril	500

Figura 14

Para obtener únicamente los nombres de los clientes, lo que se hace es una proyección, que básicamente significa que nos quedamos solo con la columna que nos interesa. En este caso, escribimos:

Πnombre-cliente (σprestatario.número-préstamo = préstamo.número-préstamo (σnombre-sucursal = «Navacerrada» (prestatario × préstamo)))

El resultado de esta consulta aparece en la Figura 14 y responde exactamente a lo que se preguntó.

Imagina que te interesa saber quiénes son los clientes con un préstamo aprobado en la sucursal de Navacerrada. Para esto necesitas información tanto de la tabla de préstamos como de la de prestatarios. Si escribes:

σnombre-sucursal = «Navacerrada» (prestatario × préstamo)

El resultado será la tabla que puedes ver en la Figura 15. Ahora, aunque la relación sólo incluye datos de la sucursal de Navacerrada, puede que la columna de nombre del cliente muestre personas que realmente no tienen un préstamo aprobado en esa sucursal. Esto sucede porque el producto cartesiano combina todas las posibilidades entre ambas tablas, así que aparecen emparejamientos que no necesariamente corresponden a préstamos realmente otorgados en esa sucursal.

nombre-cliente	*prestatario.número-préstamo*	*préstamo.número-préstamo*	*nombre-sucursal*	*importe*
Santos	P-17	P-15	Navacerrada	1.500
Santos	P-17	P-16	Navacerrada	1.300
Gómez	P-23	P-15	Navacerrada	1.500
Gómez	P-23	P-16	Navacerrada	1.300
López	P-15	P-15	Navacerrada	1.500
López	P-15	P-16	Navacerrada	1.300
Sotoca	P-14	P-15	Navacerrada	1.500
Sotoca	P-14	P-16	Navacerrada	1.300
Pérez	P-93	P-15	Navacerrada	1.500
Pérez	P-93	P-16	Navacerrada	1.300
Gómez	P-11	P-15	Navacerrada	1.500
Gómez	P-11	P-16	Navacerrada	1.300
Valdivieso	P-17	P-15	Navacerrada	1.500
Valdivieso	P-17	P-16	Navacerrada	1.300
Fernández	P-16	P-15	Navacerrada	1.500
Fernández	P-16	P-16	Navacerrada	1.300

Figura 15

El producto cartesiano en bases de datos funciona como juntar todas las combinaciones posibles entre las filas de dos tablas, por ejemplo, «prestatario» y «préstamo». Así, si queremos saber qué clientes tienen un préstamo aprobado específicamente en la sucursal de Navacerrada, necesitamos filtrar primero solo los datos de esa sucursal y luego asegurarnos de que el número de préstamo en ambas tablas coincida. Esto lo logramos con la siguiente expresión:

σprestatario.número-préstamo = préstamo.número-préstamo (σnombre-sucursal = «Navacerrada» (prestatario × préstamo))

Esta operación nos da únicamente las filas donde el cliente realmente tiene un préstamo en Navacerrada. Si lo que nos interesa es solo saber el nombre del cliente, aplicamos una proyección para quedarnos con esa columna:

Πnombre-cliente (σprestatario.número-préstamo = préstamo.número-préstamo (σnombre-sucursal = «Navacerrada» (prestatario × préstamo)))

La respuesta que obtenemos es justo la lista de clientes con préstamos aprobados en Navacerrada, tal como aparece en la Figura 16. En pocas palabras, esta consulta ayuda a filtrar y mostrar exactamente los nombres de quienes sí tienen préstamos en esa sucursal en vez de mostrar todas las combinaciones posibles.

nombre-cliente
Férnandez
López

Figura 16

Otras operaciones del algebra relacional.

- **La operación intersección de conjuntos**

La primera operación extra que vamos a ver del álgebra relacional es la intersección de conjuntos (∩). Imagina que quieres saber cuáles clientes del banco tienen tanto un préstamo aprobado como una cuenta abierta. Para obtener esa información, usamos la intersección de conjuntos, que nos ayuda a encontrar a las personas que aparecen en ambas listas. En términos prácticos, la consulta quedaría así:

Πnombre-cliente (prestatario) ∩ Πnombre-cliente (impositor).

Esto significa que buscamos los nombres de los clientes que están registrados como prestatarios y también como impositor, mostrando solo aquellos que cumplen con ambas condiciones.

La relación resultante de esta consulta aparece en la Figura 17

nombre-cliente
Gómez
Pérez
Santos

Figura 17

- **La operación reunión natural**

La reunión natural es una herramienta que nos ayuda a juntar información de dos tablas diferentes de manera inteligente. Imagina que quieres saber los nombres de

los clientes que han recibido un préstamo en el banco y, además, conocer cuánto les prestaron. En vez de hacerlo por pasos y complicarte, la reunión natural permite unir ambas tablas de una sola vez, pero solo cuando tienen datos en común, como el número de préstamo. Así, solo se combinan los registros que coinciden en ese número de préstamo y se eliminan los datos repetidos, para que la información sea clara y ordenada.

Por ejemplo, piensa en dos listas: una con los clientes y otra con los préstamos. La reunión natural busca los casos donde el número de préstamo aparece en ambas, junta los datos relevantes (nombre del cliente, sucursal, número de préstamo e importe) y, al final, nos da una tabla sencilla con la información que realmente importa. Así se facilita mucho saber quién recibió qué préstamo y cuánto fue.

Después de aplicar este proceso, te queda una lista como la que se muestra en la Figura 18, donde puedes ver fácilmente los datos de los clientes y sus préstamos.

nombre-cliente	*número-préstamo*	*importe*
Fernández	P-16	1.300
Gómez	P-23	2.000
Gómez	P-11	900
López	P-15	1.500
Pérez	P-93	500
Santos	P-17	1.000
Sotoca	P-14	1.500
Valdivieso	P-17	1.000

Figura 18

- **La operación división**

La operación división, representada con el símbolo ÷, es muy útil cuando queremos hacer consultas que implican la idea de «para todos». Por ejemplo, imagina que queremos encontrar a todos los clientes que tienen una cuenta abierta en cada una de las sucursales que se encuentran en Arganzuela. Para lograr esto, primero necesitamos identificar todas las sucursales ubicadas en Arganzuela usando una expresión específica.

número-cuenta	*nombre-sucursal*	*saldo*
C-101	Centro	500
C-215	Becerril	700
C-102	Navacerrada	400
C-305	Collado Mediano	350
C-201	Galapagar	900
C-222	Moralzarzal	700
C-217	Galapagar	750

Figura 19

nombre de la sucursal	*ciudad de la sucursal*	*activos*
Galapagar	Arganzuela	7.500
Centro	Arganzuela	9.000.000
Becerril	Aluche	2.000
Segovia	Cerceda	3.700.000
Navacerrada	Aluche	1.700.000
Navas de la Asunción	Alcalá de Henares	1.500
Moralzarzal	La Granja	2.500
Collado Mediano	Aluche	8.000.000

Figura 20

nombre cliente	*número cuenta*
Abril	C-102
Gómez	C-101
González	C-201
González	C-217
López	C-222
Rupérez	C-215
Santos	C-305

Figura 21

r1 = Πnombre-sucursal (σciudad-sucursal = «Arganzuela» (sucursal)) La relación resultante de esta expresión aparece en la Figura 21

nombre-sucursal
Centro
Galapagar

Figura 22

Para saber en qué sucursal tiene cuenta cada cliente, lo que hacemos es buscar todos los pares de nombres: el del cliente y el de la sucursal. Es decir, revisamos la lista de cuentas y anotamos junto a cada cliente la sucursal donde tiene abierta su cuenta. Así, nos queda una tabla, como la que aparece en la Figura 22, donde es fácil ver de manera clara quién tiene cuentas en qué sucursales.

nombre-cliente	*nombre-sucursal*
Abril	Collado Mediano
Gómez	Becerril
González	Centro
González	Galapagar
López	Navacerrada
Rupérez	Moralzarzal
Santos	Galapagar
Valdivieso	Navacerrada

Figura 23

Lo que sigue es identificar a los clientes que aparecen en la lista r2 y que tienen cuentas en todas las sucursales que forman parte de r1. Para esto, utilizamos la operación de división, que nos ayuda a encontrar justo a esos clientes. La consulta se escribe así: Πnombre-cliente, nombre-sucursal (impositor cuenta) ÷ Πnombre-sucursal (σciudad-sucursal = «Arganzuela» (sucursal)). El resultado de esta operación es una tabla con los nombres de los clientes que cumplen el requisito; por ejemplo, la tabla incluiría a González.

5.2 Algebra relacional extendida.

- **Proyección generalizada**

nombre-cliente	*límite*	*saldo-crédito*
Gómez	2.000	400
López	1.500	1.500
Pérez	2.000	1.750
Santos	6.000	700

Figura 24

nombre-cliente	*crédito-disponible*
Gómez	1.600
López	0
Pérez	250
Santos	5.300

Figura 25

- **Funciones de agregación**

Las funciones de agregación nos ayudan a obtener información resumida a partir de varios datos. Por ejemplo, imagina que tienes una lista de números y quieres saber cuánto suman todos juntos; para eso está la función sum, que junta todos los valores y te da el total. Si aplicamos sum a la colección {1, 1, 3, 4, 4, 11}, el resultado sería 24.

Otra función útil es avg, que sirve para calcular el promedio; si la usamos con los mismos números, el resultado sería 4. Si lo que necesitas es saber cuántos elementos hay en tu colección, puedes usar count, que en este caso te diría que hay 6 números. También existen min y max, que te muestran el número más pequeño y el más grande de la lista. Con el ejemplo anterior, min sería 1 y max sería 11.

nombre-empleado	*nombre-sucursal*	*sueldo*
González	Centro	1.500
Díaz	Centro	1.300
Jiménez	Centro	2.500
Catalán	Leganés	1.600
Cana	Leganés	1.500
Cascallar	Navacerrada	5.300
Fernández	Navacerrada	1.500
Ribera	Navacerrada	1.300

Figura 26

Imagina que queremos saber cuánto se paga en sueldos, pero no de todo el banco, sino por cada sucursal y solo a los empleados que trabajan por horas. Para lograr esto, es necesario agrupar la información de los empleados por el nombre de la

sucursal donde trabajan y luego sumar los sueldos de cada grupo por separado. En términos de álgebra relacional, esto se hace usando el operador de agregación G de la siguiente manera: nombre-sucursal Gsum(sueldo) (trabajo-por-horas).

Lo que significa esta expresión es que primero se divide la lista de empleados por horas según la sucursal a la que pertenecen. Después, dentro de cada grupo (cada sucursal), se suman los sueldos de todos los empleados de ese grupo. Al final, obtienes una tabla donde aparece el nombre de cada sucursal junto con la suma total de los sueldos que se pagan a los empleados por horas en esa sucursal, como puedes ver en la Figura 27.

nombre-sucursal	*suma de sueldos*
Centro	5.300
Leganés	3.100
Navacerrada	8.100

Figura 27

- **Reunión externa**

La operación de reunión externa es como una versión mejorada de la reunión tradicional, pensada para casos en los que falta información. Imagina que tienes dos listas: una con los datos personales de empleados que trabajan a tiempo completo (nombre, calle y ciudad), y otra con detalles sobre su empleo (nombre, sucursal y sueldo). Supón que quieres juntar todo en una sola tabla, para que cada empleado aparezca con su dirección, la sucursal donde trabaja y cuánto gana.

Normalmente, podrías usar la operación de reunión natural para unir estas dos listas, relacionando los registros por el nombre del empleado. Este método te ayuda a combinar toda la información relevante en una sola tabla, facilitando la consulta de datos completos de cada trabajador a tiempo completo.

nombre-empleado	*calle*	*ciudad*
Segura	Tebeo	La Loma
Domínguez	Viaducto	Villaconejos
Gómez	Bailén	Alcorcón
Valdivieso	Fuencarral	Móstoles

nombre-empleado	*nombre-sucursal*	*sueldo*
Segura	Majadahonda	1.500
Domínguez	Majadahonda	1.300
Barea	Fuenlabrada	5.300
Valdivieso	Fuenlabrada	1.500

Figura 28

empleado ⋈ trabajo-a-tiempo-completo

En la Figura 28 puedes ver el resultado de juntar las dos tablas. Sin embargo, nota que se perdió cierta información importante: por ejemplo, ya no aparecen la calle y la ciudad donde vive Gómez, porque en la tabla de trabajo-a-tiempo-completo no hay ningún registro que lo incluya. De forma similar, tampoco se muestran el nombre de la sucursal ni el sueldo de Barea, ya que la tabla de empleados no tiene ningún dato de ella. Así, al hacer esta unión, hay detalles que se quedan fuera si no hay coincidencias entre las listas.

nombre-empleado	*calle*	*ciudad*	*nombre-sucursal*	*sueldo*
Segura	Tebeo	La Loma	Majadahonda	1.500
Domínguez	Viaducto	Villaconejos	Majadahonda	1.300
Valdivieso	Fuencarral	Móstoles	Fuenlabrada	1.500

Figura 29

Para que no se pierda información importante al unir dos listas, se pueden usar tres tipos diferentes de reunión externa. Por ejemplo, la reunión externa por la izquierda funciona así: toma todos los registros de la primera lista (la de la izquierda) que no tengan coincidencias en la segunda lista (la de la derecha), les pone “nulo” en los lugares donde falta información de la derecha y los agrega al resultado final junto con las coincidencias habituales. En la Figura 30 se muestra un caso: la fila (Gómez, Bailén, Alcorcón, nulo, nulo) aparece porque Gómez no tiene datos en la otra tabla, pero gracias a esta operación, no se pierde su información. Así, todos los datos de la primera lista se mantienen en el resultado, aunque falten detalles de la segunda.

nombre-empleado	*calle*	*ciudad*	*nombre-sucursal*	*sueldo*
Segura	Tebeo	La Loma	Majadahonda	1.500
Domínguez	Viaducto	Villaconejos	Majadahonda	1.300
Valdivieso	Fuencarral	Móstoles	Fuenlabrada	1.500
Gómez	Bailén	Alcorcón	*nulo*	*nulo*

Figura 30

La reunión externa por la derecha funciona de manera parecida a la reunión por la izquierda, pero al revés. Por ejemplo, imagina que tienes una lista con información adicional de algunos empleados, como la sucursal donde trabajan y su sueldo, pero esta lista no incluye a todos los empleados que aparecen en la otra tabla. Cuando usas la reunión externa por la derecha, te aseguras de que todos los registros de esta segunda lista (la de la derecha) estén presentes en el resultado, aunque no tengan pareja en la primera. Si hay datos que no coinciden, los espacios que faltan se llenan con “nulo” para que nada se pierda. Por ejemplo, en la Figura 31 aparece la fila (Barea, nulo, nulo, Fuenlabrada, 5,300), lo que significa que Barea tiene información en la tabla de la derecha, pero no en la de la izquierda, así que sus

datos personales aparecen como “nulo”. Así, toda la información de la segunda lista se incluye en el resultado, sin importar si tiene coincidencia en la primera.

nombre-empleado	*calle*	*ciudad*	*nombre-sucursal*	*sueldo*
Segura	Tebeo	La Loma	Majadahonda	1.500
Domínguez	Viaducto	Villaconejos	Majadahonda	1.300
Valdivieso	Fuencarral	Móstoles	Fuenlabrada	1.500
Barea	*nulo*	*nulo*	Fuenlabrada	5.300

- Figura 31

Figura 31

La reunión externa completa funciona como una combinación de las dos operaciones anteriores: por la izquierda y por la derecha. Esto significa que, al juntar dos listas, se asegura de que ninguna información se pierda. Si algún registro de la primera lista no tiene pareja en la segunda, se mantiene en el resultado y los datos faltantes se rellenan con “nulo”. Lo mismo pasa con los registros de la segunda lista que no tienen coincidencia en la primera: también se agregan y se rellenan los espacios con “nulo” donde sea necesario. Así, el resultado incluye todos los datos disponibles de ambas listas, aunque falten coincidencias. En la Figura 32 puedes ver cómo queda el resultado cuando se aplica una reunión externa completa.

nombre-empleado	*calle*	*ciudad*	*nombre-sucursal*	*sueldo*
Segura	Tebeo	La Loma	Majadahonda	1.500
Domínguez	Viaducto	Villaconejos	Majadahonda	1.300
Valdivieso	Fuencarral	Móstoles	Fuenlabrada	1.500
Gómez	Bailén	Alcorcón	*nulo*	*nulo*
Barea	*nulo*	*nulo*	Fuenlabrada	5.300

Figura 32

Operaciones para modificar la base de datos

Hasta ahora, nos hemos enfocado en cómo obtener información de la base de datos, pero también es importante saber cómo cambiarla. En este apartado vamos a ver cómo se pueden agregar, eliminar o modificar datos dentro de la base.

Para hacer estos cambios, se utiliza una operación llamada asignación. Cuando queremos que estos cambios se apliquen directamente sobre las tablas reales de la base de datos, usamos el símbolo ←.

Borrar datos:

Las instrucciones para borrar datos son muy parecidas a las consultas normales. La diferencia es que, en vez de mostrarte los datos seleccionados, se eliminan de la base de datos las filas que cumplen con la condición que indiques. Es importante saber que solo puedes borrar filas completas, no partes específicas de ellas. En el álgebra relacional, el borrado se representa así:

r ← r – E donde r es una relación y E es una consulta del álgebra relacional.

He aquí varios ejemplos de solicitudes de borrado del álgebra relacional:

> Borrar todas las cuentas de Gómez. impositor ← impositor – σnombre-cliente = «Gómez» (impositor) - Borrar todos los préstamos con importes entre 0 y 50.

préstamo ← préstamo – σimporte ≥ 0 and importe ≤ 50 (préstamo)

Inserción de datos

Cuando quieres agregar información nueva a una tabla de la base de datos, tienes que decidir qué datos vas a meter. Esto puede ser tan simple como añadir una sola fila, o tan complejo como insertar el resultado de una consulta que genera varias filas. Lo importante es que cada dato que pongas debe coincidir con el tipo de dato que espera cada columna, y que la cantidad de datos debe ser la correcta para la tabla.

En el lenguaje del álgebra relacional, insertar datos se ve como r ← r ∪ E, donde "r" es la tabla y "E" lo que vas a añadir. Si solo quieres meter una fila, "E" es nada más esa fila escrita como una constante.

Por ejemplo, digamos que quieres registrar que el cliente Gómez tiene $1,200.00 MXN en la cuenta C-973, que está en la sucursal de Navacerrada. Así lo harías:

- Para la tabla cuentas, escribirías: cuenta ← cuenta ∪ {(C-973, "Navacerrada", 1200)}
- Para la tabla de impositores, pondrías: impositor ← impositor ∪ {("Gómez", C-973)}

Así de sencillo es agregar datos nuevos a una base de datos usando álgebra relacional, siempre y cuando respetes las reglas de los tipos de datos y el número de columnas.

Actualización

En ocasiones, es posible que solo quieras cambiar uno de los valores dentro de una fila (tupla) de una tabla, sin necesidad de modificar todos los demás datos de esa misma fila. Para lograr esto, se puede usar lo que en álgebra relacional se llama la “proyección generalizada”. Esta operación se escribe así:

r ← ΠF1, F2, …, Fn (r)

Para que te quede más claro, imagina que tienes que actualizar los saldos de todas las cuentas bancarias porque se va a pagar un 5% de intereses. En este caso, lo

que harías sería aumentar el saldo de cada cuenta en ese porcentaje. Usando la notación del álgebra relacional, la actualización quedaría así:

cuenta ← Πnombre-sucursal, número-cuenta, saldo, saldo * 1.05 (cuenta)

De esta forma, todos los saldos reflejarán el pago de intereses sin necesidad de modificar las demás columnas.

CUESTIONARIO TEMA 5

1. **¿Cuál de las siguientes operaciones del álgebra relacional es una operación unaria?**

   a) Unión

   b) Selección

   c) Producto cartesiano

2. **¿Qué símbolo se utiliza para indicar la operación de selección en álgebra relacional?**

   a) Π (pi mayúscula)

   b) σ (sigma minúscula)

   c) × (producto cartesiano)

3. **¿Qué operación relacional se usa para obtener las tuplas que están en una relación, pero no en otra?**

   a) Intersección
   b) Diferencia de conjuntos
   c) Unión

4. **En el producto cartesiano de dos relaciones, ¿cómo se evita la ambigüedad cuando ambos esquemas tienen atributos con el mismo nombre?**
   a) Se eliminan los atributos duplicados automáticamente

   b) Se renombra el atributo en la relación de la derecha con un prefijo
   c) Se añade el nombre de la relación como prefijo al atributo

5. **¿Cuál es la diferencia principal entre la reunión natural y el producto cartesiano seguido de selección en álgebra relacional?**

   a) La reunión natural elimina atributos duplicados y fuerza igualdad en atributos comunes, mientras que el producto cartesiano no
   b) La reunión natural solo trabaja con una relación, el producto cartesiano con dos
   c) El producto cartesiano realiza la unión de las relaciones, la reunión natural no

6. **La operación división (÷) en álgebra relacional se usa para:**
   a) Encontrar todos los clientes que tienen una cuenta en todas las sucursales de una ciudad específica

   b) Calcular la suma de valores en un atributo numérico

   c) Encontrar la intersección entre dos relaciones

7. **¿Qué función de agregación devuelve el número de elementos en un conjunto?**
   a) suma

b) recuento

c) promedio

8. **¿Cuál es la característica principal de la reunión externa por la izquierda?**
   a) Incluye todas las tuplas de la relación derecha, rellenando con nulos las faltantes de la izquierda

   b) Incluye todas las tuplas de la relación izquierda, rellenando con nulos las faltantes de la derecha

   c) Combina las tuplas comunes sin añadir ninguna tupla adicional

9. **¿Cómo se expresa en álgebra relacional la eliminación de todas las cuentas de un cliente llamado "Gómez"?**

   a) impositor ← impositor – σnombre-cliente = «Gómez» (impositor)
   b) impositor ← impositor ∪ σnombre-cliente = «Gómez» (impositor)
   c) impositor ← impositor × σnombre-cliente = «Gómez» (impositor)

10. **Para aumentar todos los saldos de una relación cuenta en un 5%, ¿qué operación es correcta?**

    a) cuenta ← Πnombre-sucursal, número-cuenta, saldo * 1.05 (cuenta)
    b) cuenta ← Πnombre-sucursal, número-cuenta, saldo, saldo * 1.05 (cuenta)
    c) cuenta ← cuenta – σsaldo > 0 (cuenta)

Respuestas correctas:

1. b) Selección
2. b) σ (sigma minúscula)
3. b) Diferencia de conjuntos
4. c) Se añade el nombre de la relación como prefijo al atributo
5. a) La reunión natural elimina atributos duplicados y fuerza igualdad en atributos comunes, mientras que el producto cartesiano no
6. a) Encontrar todos los clientes que tienen una cuenta en todas las sucursales de una ciudad específica
7. b) contar
8. b) Incluye todas las tuplas de la relación izquierda, rellenando con nulos las faltantes de la derecha
9. a) impositor ← impositor – σnombre-cliente = «Gómez» (impositor)
10. b) cuenta ← Πnombre-sucursal, número-cuenta, saldo, saldo * 1.05 (cuenta)

Tema 6: Implementación Práctica: Creación de la Base de Datos

Aquí se contempla aplicar los comandos básicos del Lenguaje de Definición y de Manipulación de Datos, haciendo uso de las herramientas del Sistema Gestor de Base de Datos, entre los cuales se deben considerar la creación de base de datos, creación de tablas y definición de llaves primarias y foráneas, la manipulación y consulta de la base de datos por medio de las operaciones de inserción, eliminación, modificación y consulta de datos. Es importante que el profesor aborde este tema a nivel básico, ya que en la asignatura de Taller de Base de Datos se dará profundidad en la definición, manipulación y control de la base de datos.

6.1. Herramientas de Modelado y SGBD Populares

El diseño de una base de datos no termina hasta que se lleva a la práctica, y hoy en día ese paso resulta mucho más sencillo gracias a herramientas modernas. Por ejemplo, existen aplicaciones como MySQL Workbench que permiten a los ingenieros crear diagramas visuales —es decir, plasmar gráficamente cómo será la base de datos— y, con solo unos clics, generan automáticamente el código SQL necesario para ponerla en marcha. Esto ayuda a que el salto del papel a la realidad sea directo y minimiza los errores que podrían surgir si el proceso fuera manual.

En la carrera de ingeniería en sistemas, es fundamental saber cómo implementar bases de datos usando sistemas populares como MySQL y PostgreSQL. Estos programas no solo se utilizan ampliamente en empresas, sino que también son herramientas básicas en el entorno académico.

SQL (Structured Query Language) es el lenguaje estándar para interactuar con bases de datos relacionales. Se utiliza para **gestionar, consultar y manipular** los datos. Aunque existen variaciones entre diferentes sistemas de gestión de bases de datos (SGBD) como SQL Server, MySQL y Oracle, las instrucciones básicas son las mismas.

Instrucciones DDL (Data Definition Language)

Se utilizan para definir, modificar y eliminar la estructura de los objetos de la base de datos.

- **CREATE**: Se usa para crear nuevos objetos, como bases de datos, tablas, vistas, etc.

SQL

```
CREATE TABLE Empleados (
    EmpleadoID INT PRIMARY KEY,
    Nombre VARCHAR(50) NOT NULL,
    FechaContratacion DATE
```

);

- **ALTER**: Permite modificar la estructura de un objeto existente.

SQL

ALTER TABLE Empleados

ADD Salario DECIMAL(10, 2);

- **DROP**: Elimina un objeto de la base de datos.

SQL

DROP TABLE Empleados;

- **TRUNCATE**: Elimina todas las filas de una tabla, pero mantiene su estructura. Es más rápido que DELETE para borrar todos los registros.

SQL

TRUNCATE TABLE Empleados;

Instrucciones DML (Data Manipulation Language)

Se utilizan para manipular los datos dentro de las tablas.

- **INSERT**: Agrega nuevas filas de datos a una tabla.

SQL

INSERT INTO Empleados (EmpleadoID, Nombre, FechaContratacion)

VALUES (1, 'Juan Pérez', '2023-01-15');

- **UPDATE**: Modifica datos existentes en una o más filas.

SQL

UPDATE Empleados

SET Salario = 60000.00

WHERE EmpleadoID = 1;

- **DELETE**: Elimina filas de una tabla.

SQL

DELETE FROM Empleados

WHERE EmpleadoID = 1;

Instrucciones DQL (Data Query Language)

Se utilizan para consultar y recuperar datos de la base de datos.

- **SELECT**: Recupera datos de una o varias tablas. Es la instrucción más utilizada.

SQL

-- Selecciona todas las columnas y filas de la tabla

SELECT * FROM Empleados;

-- Selecciona columnas específicas

SELECT Nombre, Salario FROM Empleados;

-- Selecciona datos con una condición

SELECT Nombre FROM Empleados WHERE Salario > 50000;

Instrucciones DCL (Data Control Language)

Se utilizan para controlar los permisos y el acceso a los datos.

- **GRANT**: Otorga permisos a un usuario o rol.

SQL

GRANT SELECT, INSERT ON Empleados TO Usuario1;

- **REVOKE**: Revoca un permiso previamente otorgado.

SQL

REVOKE DELETE ON Empleados FROM Usuario1;

- **DENY**: Niega explícitamente un permiso a un usuario, incluso si lo hereda de un rol.

SQL

DENY DELETE ON Empleados TO Usuario1;

6.2. Creación de Tablas y Restricciones con SQL

La implementación de un diseño relacional se realiza principalmente con el Lenguaje de Definición de Datos (LDD) de SQL, a través de comandos como CREATE TABLE.[18] La sintaxis de este comando permite definir la estructura de la tabla, incluyendo sus columnas, sus tipos de datos y sus restricciones. Las restricciones son cruciales para asegurar la integridad de los datos.

- Clave Primaria (PRIMARY KEY): Se utiliza para identificar de forma única cada registro en una tabla. Un ejemplo de su sintaxis es: CREATE TABLE productos (codigo INT NOT NULL AUTO_INCREMENT PRIMARY KEY,...).

- Clave Foránea (FOREIGN KEY): Se usa para establecer una relación entre dos tablas, enlazando los datos de una con los de otra. Por ejemplo, en una relación uno a muchos entre CLIENTES y PEDIDOS, la tabla pedidos contendría una clave foránea id_cliente que referencia a la clave primaria de la tabla clientes.

Un aspecto importante de la implementación de claves foráneas es la definición de las reglas de integridad referencial, que dictan el comportamiento de la base de datos cuando se modifican o eliminan registros en la tabla referenciada. Las opciones comunes incluyen ON DELETE RESTRICT, que previene la eliminación de un registro si está siendo referenciado, y ON DELETE CASCADE, que elimina automáticamente los registros relacionados. Al elegir una de estas opciones, el ingeniero de bases de datos está traduciendo una regla de negocio del mundo real (p. ej., "un pedido no puede existir sin un cliente") a un comportamiento automatizado del SGBD, lo que garantiza la coherencia física del modelo.

El modelo relacional ha sido el paradigma dominante en la gestión de datos durante décadas. Sin embargo, el auge de las tecnologías de la información, el big data y la web ha dado lugar al surgimiento de bases de datos no relacionales (NoSQL), que ofrecen una arquitectura flexible y una escalabilidad horizontal superior. La elección entre un modelo relacional y uno NoSQL no es una dicotomía simple, sino una decisión de arquitectura estratégica que debe basarse en los requerimientos específicos de la aplicación.

- Modelo Relacional: Es ideal para escenarios donde la integridad y la estructura de los datos son fundamentales.[38] Sus principales fortalezas radican en la garantía de las propiedades ACID, lo que lo hace indispensable para aplicaciones de misión crítica como sistemas de transacciones financieras o de gestión de inventario. Además, su soporte para el lenguaje SQL permite la realización de consultas complejas y operaciones de unión de manera eficiente.
- Modelo NoSQL: Emplea una arquitectura que no se basa en el esquema rígido de tablas, filas y columnas. Es especialmente valioso para entornos con datos masivos, complejos o que cambian rápidamente. Sus ventajas incluyen la flexibilidad del esquema, la capacidad de manejar tipos de datos diversos (como JSON o XML) y una escalabilidad horizontal inherente, lo que lo hace idóneo para bases de datos de redes sociales, análisis de big data en tiempo real y aplicaciones de IoT.

La coexistencia del modelo relacional y los modelos NoSQL ha creado un ecosistema de datos diverso. Un profesional con una visión estratégica no debe casarse con un solo paradigma, sino ser capaz de evaluar y seleccionar la tecnología más apropiada para el problema a resolver. Si un proyecto requiere una integridad transaccional rigurosa, el modelo relacional sigue siendo la mejor opción. Si, por el contrario, el requisito principal es la ingesta masiva de datos no estructurados y una escalabilidad dinámica, una base de datos NoSQL es más adecuada.[38] Esta capacidad de tomar decisiones arquitectónicas informadas es lo que distingue a un ingeniero en sistemas con visión empresarial y global.

Bibliografía

1. Bases de Datos - Facultad de Ingeniería - UNAM https://www.ingenieria.unam.mx/programas_academicos/licenciatura/Computacion/07/bases_de_datos.pdf
2. Modelo Entidad-Relación: qué es, cómo se hace y ejemplos - iLERNA, https://www.ilerna.es/blog/modelo-entidad-relacion-base-datos
3. Modelo entidad-relación - Wikipedia, la enciclopedia libre, https://es.wikipedia.org/wiki/Modelo_entidad-relaci%C3%B3n
4. Notación Modelo Entidad Relación (MER), https://ejuarez.bitbucket.io/daw/lectura2_Mer/lectura2.html
5. Crear un diagrama con notación de base de datos de Chen - Soporte técnico de Microsoft, https://support.microsoft.com/es-es/topic/crear-un-diagrama-con-notaci%C3%B3n-de-base-de-datos-de-chen-75d28eff-2509-4faf-8cd9-3eda5fb4327b
6. Unidad 3 Entidades Fuertes y Débiles - Quizlet, https://quizlet.com/es/868454900/unidad-3-entidades-fuertes-y-debiles-flash-cards/
7. 3.1.- Tipos: fuertes y débiles. | DAM_BD03_Contenido - S'Arreplec, https://sarreplec.caib.es/pluginfile.php/9827/mod_resource/content/2/31_tipos_fuertes_y_dbiles.html
8. Diagrama entidad relación: ¿Qué es para qué sirve? Con ejemplos - Miro, https://miro.com/es/diagrama/que-es-diagrama-entidad-relacion/
9. 2.1.7. Entidades debiles - Dataprix, https://www.dataprix.com/es/bases-de-datos-master-de-software-libre-de-la-uoc/217-entidades-debiles
10. Modelo de Entidad - Relación - EVELIA, https://www.evelia.unrc.edu.ar/evelia/archivos/idAula5086375488/materiales/teoricos/Teorico_2_Entidad-Relacion_.pdf
11. Todo sobre Cardinalidad en Base de Datos + Ejemplos, 2025, https://informaticosinlimites.com/base-de-datos/cardinalidad/
12. Guía Completa de Atributos en Base de Datos! + Ejemplos, 23, 2025, https://informaticosinlimites.com/base-de-datos/atributos/
13. Normalización de Bases de Datos - UNAM, https://programas.cuaed.unam.mx/repositorio/moodle/pluginfile.php/872/mod_resource/content/7/Contenido/index.html
14. 4.1.- Tipos de atributos. | DAM_BD03_Contenido - S'Arreplec, https://sarreplec.caib.es/pluginfile.php/9827/mod_resource/content/2/41_tipos_de_atributos.html

15. Cómo crear una tabla en MySQL | Andrés Ledo, https://andresledo.es/mysql/crear-tablas/

16. 1. Modelo ENTIDAD - RELACIÓN (entidades, relaciones, atributos y primary key o llave primaria) - YouTube, https://www.youtube.com/watch?v=Apj02at-Cic&pp=0gcJCfwAo7VqN5tD

17. análisis y diseño - UNS, acceso: http://cs.uns.edu.ar/~td/ayds2012/downloads/Clases/ADS_10_2012-%20Modelos%20de%20Datos%20IV.pdf

18. Grado, https://danielbenvenuto.com/EDUCACION/EBD-I/grado.htm

19. 5.1.- Grado de una relación. | DAM_BD03_Contenido - S'Arreplec, https://sarreplec.caib.es/pluginfile.php/9827/mod_resource/content/2/51_grado_de_una_relacin.html

20. Muchos a muchos, uno a muchos, muchos a uno : r/SQL - Reddit, https://www.reddit.com/r/SQL/comments/1e14nqr/many_to_many_one_to_many_many_to_one/?tl=es-419

21. ¿Qué es una base de datos relacional (RDBMS)? | Google Cloud, https://cloud.google.com/learn/what-is-a-relational-database?hl=es-419

22. es.wikipedia.org, acceso: https://es.wikipedia.org/wiki/C%C3%A1lculo_relacional_basado_en_tuplas#:~:text=Base%20de%20datos%20relacional,-Debido%20a%20que&text=Una%20tupla%20o%20registro%20es,visual%20aceptada%20de%20una%20relaci%C3%B3n.

23. Tupla - AppMaster, https://appmaster.io/es/glossary/tupla-es

24. Introducción a los dominios de atributo—ArcGIS Pro | Documentación, https://pro.arcgis.com/es/pro-app/latest/help/data/geodatabases/overview/an-overview-of-attribute-domains.htm

25. pro.arcgis.com, https://pro.arcgis.com/es/pro-app/latest/help/data/geodatabases/overview/an-overview-of-attribute-domains.htm#:~:text=Los%20dominios%20de%20atributo%20son,tabla%20o%20clase%20de%20entidad.

26. Un recorrido rápido por los dominios de atributo - ArcMap Resources for ArcGIS Desktop, https://desktop.arcgis.com/es/arcmap/latest/manage-data/geodatabases/an-overview-of-attribute-domains.htm

27. 12 reglas de Codd - Wikipedia, la enciclopedia libre, 2025, https://es.wikipedia.org/wiki/12_reglas_de_Codd

28. Codd's 12 rules - Wikipedia, https://en.wikipedia.org/wiki/Codd%27s_12_rules

29. 12 Reglas de Codd | PDF | Modelo relacional | Bases de datos - Scribd, https://es.scribd.com/document/38686971/12-reglas-de-Codd
30. MODELOS DE ESTRUCTURACIÓN, https://www.uv.es/ceaces/base/tratnoes/modelos.htm
31. 05-Reglas de Codd. - Base de Datos - El Profe Alegría, 2025, https://elprofealegria.com/05-reglas-de-codd/
32. Modelo Jerarquico y Red | PDF | Bases de datos | Ingeniería Informática - Scribd, acceso: https://es.scribd.com/document/442051534/modelo-jerarquico-y-red
33. Bases de Datos Relacionales y SQL - Introducción 2025, https://aprenderbigdata.com/bases-de-datos-relacionales/
34. ¿Qué es un sistema de administración de bases de datos ..., https://azure.microsoft.com/es-es/resources/cloud-computing-dictionary/what-is-a-relational-database
35. ¿Cuál es la diferencia? Bases de datos relacionales y no relacionales - insightsoftwarehttps://insightsoftware.com/es/blog/whats-the-difference-relational-vs-non-relational-databases/
36. Bases de datos SQL vs NoSQL: Diferencias clave e ideas prácticas – DataCamp, https://www.datacamp.com/es/blog/sql-vs-nosql-databases
37. Normalización de Base de Datos (1NF, 2FN, 3FN, BCFN, 4FN) | SQL SERVER 2020, https://www.youtube.com/watch?v=bTPQfOfhIQs
38. La normalización de una base de datos: qué es, reglas, formas ..., https://ebac.mx/blog/normalizacion-de-bases-de-datos
39. Boyce–Codd normal form - Wikipedia, https://en.wikipedia.org/wiki/Boyce%E2%80%93Codd_normal_form
40. Difference between 3NF and BCNF in simple terms (must be able to explain to an 8-year old) - Stack Overflow, https://stackoverflow.com/questions/8437957/difference-between-3nf-and-bcnf-in-simple-terms-must-be-able-to-explain-to-an-8
41. Boyce-Codd Normal Form (BCNF) - GeeksforGeeks, 2025, https://www.geeksforgeeks.org/dbms/boyce-codd-normal-form-bcnf/
42. Forma normal de Boyce-Codd - Wikipedia, la enciclopedia libre, https://es.wikipedia.org/wiki/Forma_normal_de_Boyce-Codd
43. Normalización - FING, https://www.fing.edu.uy/tecnoinf/paysandu/cursos/2do/bd1/material/bd1-7.pdf

44. Teórico 1 - Algebra Relacional - Bases de Datos 2, https://www.fing.edu.uy/tecnoinf/maldonado/cursos/bd2/materiales/teo/bd2-teorico01.pdf
45. Cálculo Relacional de Tuplas (CRT) - UNS, http://cs.uns.edu.ar/~gis/ebd/Archivos/Clases/EBD%20-%20Clase%2007%202004%20BN.pdf
46. ÁLGEBRA RELACIONAL, https://www.uv.mx/personal/ermeneses/files/2020/09/Clase9-OperacionesBDRelacionales-I_respuestas.pdf
47. Uso de MySql WorkBench para el diseño de modelo relacional - YouTube, https://www.youtube.com/watch?v=9wPl7Sv6ojk
48. Llaves primarias/foráneas [PostgreSQL] : r/SQL - Reddit, https://www.reddit.com/r/SQL/comments/10qvtn6/primary_foreign_keys_postgresql/?tl=es-419
49. Guía para aprender a utilizar MySQL - IONOS, https://www.ionos.com/es-us/digitalguide/servidores/know-how/guia-para-aprender-a-utilizar-mysql/
50. Documentation: 7.1: CREATE TABLE - PostgreSQL, https://www.postgresql.org/docs/7.1/sql-createtable.html