Universidad Juárez Autónoma de Tabasco

División Académica de Ciencias Básicas

# Bases de datos como herramienta actuarial

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

**LICENCIADO EN ACTUARÍA**

PRESENTA:

**ANDRÉS IZQUIERDO MORALES**

DIRECTOR DEL TRABAJO:
DR. DAVID JOSAFAT SANTANA COBIAN

CUNDUACÁN, TABASCO 2025.

# Declaración de Autoría y Originalidad

En la ciudad de Cunduacán, el día 18 del mes de febrero del año dos mil veinticinco, el que suscribe Andres Izquierdo Morales alumno del Programa de licenciatura con número de matrícula 172A9015, adscrito a la licenciatura en Actuaría, de la universidad Juárez Autónoma de Tabasco, como autor de la Tesis presentada para la obtención del titulo de licenciado en Actuaría y titulada "Bases de datos como herramienta actuarial" dirigida por el Dr. David Josafat Santana Cobian.

**DECLARO QUE:**

La Tesis es una obra original que no infringe los derechos de propiedad intelectual ni los derechos de propiedad industrial u otros, de acuerdo con el ordenamiento jurídico vigente, en particular, la LEY FEDERAL DEL DERECHO DE AUTOR (Decreto por el que se reforman y adicionan diversas disposiciones de la Ley Federal del Derecho de Autor del 01 de Julio de 2020 regularizando y aclarando y armonizando las disposiciones legales vigentes sobre la materia), en particular, las disposiciones referidas al derecho de cita.

Del mismo modo, asumo frente a la Universidad cualquier responsabilidad que pudiera derivarse de la autoría o falta de originalidad o contenido de la Tesis presentada de conformidad con el ordenamiento jurídico vigente.

Villahermosa, Tabasco a 18 de febrero de 2025.

Nombre y Firma

Andres Izquierdo Morales

UJAT
UNIVERSIDAD JUÁREZ AUTÓNOMA DE TABASCO
"ESTUDIO EN LA DUDA. ACCIÓN EN LA FE"

División Académica de Ciencias Básicas

2025 AÑO DE LA Mujer Indígena

DIRECCIÓN

Cunduacán, Tabasco; a 06 de febrero de 2025.

**C. ANDRÉS IZQUIERDO MORALES**
**PASANTE DE LA LICENCIATURA EN ACTUARÍA**
**PRESENTE**

Por medio del presente, me dirijo a usted para hacer de su conocimiento que proceda a la impresión del trabajo titulado **"BASES DE DATOS COMO HERRAMIENTA ACTUARIAL"**, dirigido por el Dr. David Josafat Santana Cobián, bajo la modalidad de titulación por **TESIS**. La comisión de revisión conformada por el Dr. Abdiel Emilio Cáceres González, Mtro. Candelario Méndez Olan, Mtra. Alejandra Emperatriz Flores Palacios, Dr. David Josafat Santana Cobián y Lic. Landy Grissel Uc Aguilar, liberó el documento en virtud de que reúne los requisitos para el **EXAMEN PROFESIONAL** correspondiente.

Sin otro particular, reciba usted un cordial saludo.

**ATENTAMENTE**

DIVISIÓN ACADÉMICA DE CIENCIAS BÁSICAS

**DRA. HERMICENDA PÉREZ VIDAL**
**DIRECTORA**

C.c.p. Archivo.

DIR´DRA.HPV/kfvg

Km.1 Carretera Cunduacán-Jalpa de Méndez, A.P. 24, C.P. 86690, Cunduacán, Tab., México.
Tel/Fax: (993) 3581500 Ext. 6702,6701 E-Mail: direccion.dacb@ujat.mx

www.ujat.mx

# Carta de Cesión de Derechos

Villahermosa, Tabasco a 18 de febrero de 2025.

Por medio de la presente manifestamos haber colaborado como AUTOR(A) y/o AUTORES(RAS) en la producción, creación y/o realización de la obra denominada "Bases de datos como herramienta actuarial".

Con fundamento en el artículo 83 de la Ley Federal del Derecho de Autor y toda vez que, la creación y/o realización de la obra antes mencionada se realizó bajo la comisión de la Universidad Juárez Autónoma de Tabasco; entendemos y aceptamos el alcance del artículo en mención, de que tenemos el derecho al reconocimiento como autores de la obra, y la Universidad Juárez Autónoma de Tabasco mantendrá en un 100% la titularidad de los derechos patrimoniales por un período de 20 años sobre la obra en la que colaboramos, por lo anterior, cedemos el derecho patrimonial exclusivo en favor de la Universidad.

## COLABORADORES

Andres Izquierdo Morales

Dr. David Josafat Santana Cobian

## TESTIGOS

Claudia Soledad López Pérez

María del Cielo Pérez Sevilla

# Dedicatoria

Quiero agradecer al arquitecto del universo por permitir estar con salud durante el proceso de titulación, de vivir estos momentos de felicidad con mis seres queridos, de igual manera por permitir darme la inteligencia y resiliencia para concluir este proyecto.

A mis padres, Antonio Izquierdo y Deisy Morales, por su amor incondicional y apoyo inconmensurable que tuvieron antes, durante y no dudo que después de este gran logro, ellos, quienes me han enseñado el valor del esfuerzo y la perseverancia.

A mis amigos, por ser parte de cada escrito, por esas noches en las que debatíamos temas acordes, por dar ideas de mejora y aliento en momentos difíciles. Y a mis profesores, por su conocimiento compartido y el buen acompañamiento que me dieron en momentos de duda, gracias por ser esa guía estudiantil que fue fundamental en mi formación académica. Esta tesis es un reflejo de todo lo que he aprendido y de todas las personas que han estado en este caminar.

# LICENCIATURA - TR BASE DE DATOS COMO HERRAMIENTA ACTUARIAL - ANDRÉS IZQUIERDO MORALES

INFORME DE ORIGINALIDAD

**18%**
ÍNDICE DE SIMILITUD

FUENTES PRIMARIAS

| | | |
|---|---|---|
| 1 | pdfcoffee.com<br>Internet | 1132 palabras — 5% |
| 2 | edoc.pub<br>Internet | 596 palabras — 2% |
| 3 | www.coursehero.com<br>Internet | 324 palabras — 1% |
| 4 | doku.pub<br>Internet | 260 palabras — 1% |
| 5 | qdoc.tips<br>Internet | 243 palabras — 1% |
| 6 | idoc.pub<br>Internet | 219 palabras — 1% |
| 7 | dokumen.pub<br>Internet | 211 palabras — 1% |
| 8 | www.scribd.com<br>Internet | 152 palabras — 1% |
| 9 | 1pdf.net<br>Internet | 126 palabras — 1% |

| Alojamiento de la Tesis en el Repositorio Institucional | |
|---|---|
| **Título de Tesis:** | Bases de datos como herramienta actuarial. |
| **Autor de la Tesis** | Andres Izquierdo Morales |
| **ORCID:** | 0009-0004-1981-4511 |
| **Resumen de la Tesis:** | Destaca la importancia de las TIC en el almacenamiento y acceso a los datos, se centra en la relevancia de aprender a gestionar bases de datos en la carrera en Actuaría. Y muestra la necesidad de toma de decisiones en instituciones. |
| **Palabras claves de la Tesis:** | Base de datos relacionales, DBMS, Modelado de datos, Normalización, SQL, Reglas de integridad, Toma de decisiones. |
| **Referencias citadas:** | **Codd, E. F.** (1970). "A Relational Model of Data for Large Shared Data Banks". Communications of the ACM.<br>**Ricardo, C. M.** (2009). "Bases de datos". McGraw Hill.<br>**Silberschatz, A., Korth, H. F., & Sudarshan, S.** (1998). "Fundamentos de bases de datos". McGraw Hill.<br>**Date, C. J.** (2001). "Introducción a los sistemas de bases de datos". Pearson Educación.<br>**Otros...** |

# Índice general

# Índice de figuras

# Índice de cuadros

# Capítulo 1

# Introducción

Las Tecnologías de la Información y Comunicación (TIC) han permitido el almacenamiento y acceso de todo tipo de datos, desde una visita a una página de Facebook hasta el número de transacciones diarias que se hacen en cierta casa de bolsa. Lo anterior, ha conducido a que individuos, colectivos o instituciones científicas se organicen para desarrollar programas de gran utilidad en la administración de datos y de igual manera poder estudiar el comportamiento de estos. De este modo se toma en cuenta que cada dato es almacenado de forma electrónica en un sistema informático organizado y estructurado en conjunto, para así formar una base de datos.

Este trabajo se centra en analizar la importancia de aprender bases de datos en la carrera de Actuaría, asociado a los cambios en el mercado laboral, la alta demanda que se tiene en México para los negocios e instituciones gubernamentales.

Se busca que el lector analice y aprenda conceptos básicos de bases de datos relacionales, código SQL y comprenda cómo impactan en el ámbito actuarial.

En mi experiencia trabajando en el sector bancario al norte del país, pude notar la gran demanda que se tienen a los profesionistas que dominan herramientas de administración y análisis de datos. La institución en la que trabajé usaba técnicas básicas de consulta para encontrar a clientes potenciales y así ofertarles beneficios financieros, por ejemplo: una nueva tasa de interés más atractiva para sus créditos, un tipo de cambio competitivo, contratos *forward* para sus transacciones futuras, entre otros. Además, contaban con una *Fintech*, la cual reduce costos y permite ofrecer mejores promociones a sus clientes con el buen análisis y administración de bases de datos.

Se toma en cuenta la premisa de que los usuarios generalmente no solo requieren una base de datos, sino que requieren formas, reportes y preguntas basadas en sus datos. Si el usuario tiene una gran cantidad de información en forma de tablas es necesario contar con un Sistema Gestor de Bases de Datos (`SGBD`) que es un software para manejar bases de datos. Como actuarios, es muy importante saber manejar esos `SGBD` que son conocimientos muy valorados a la hora de competir en el mercado laboral, la intención de este trabajo es que el actuario conozca lo necesario para que se desarrolle como administrador de datos.

Finalmente, se busca identificar las áreas de oportunidad que uno puede detectar en la administración de bases de datos, poder aplicar reglas de operación efectivas y abstraerlas en una tabla, es una actividad que se desarrolla durante este trabajo para que otros

actuarios puedan desarrollar modelos de datos, administrar y analizar bases de datos como complemento a sus actividades.

## 1.1. Panorama general de la tesis

### 1.1.1. Marco teórico

Los diferentes mecanismos de almacenamiento de bases de datos, están estrictamente ligados a su desarrollo y evolución histórica. Según se ha ido avanzando en la tecnología, las bases de datos han ido mejorando, actualizando e innovando la forma de representar y extraer la información para su análisis.

De esta manera, se representa la evolución que tuvieron las bases de datos; que va desde el almacenamiento de datos en tarjetas perforadoas, hasta los Sistemas de Gestión de Bases de Datos (`SGBD`).

La importancia de los datos es un tema que importa a muchos, de modo que la finalidad de estos es crear información valiosa que resulte benéfica. La distribución de la información es otro tema para tener en cuenta. La necesidad de hacer registros no es nueva para la sociedad actual, Catherine M [2]. pone en contexto que desde el principio de los tiempos del hombre se dio; mencionando lo siguiente: "Los intentos para proporcionar registros permanentes de transacciones se ven en las tablas de arcilla sumerias, en artefactos dejados por los babilónicos, en los jeroglíficos del antiguo Egipto e incluso en las pinturas rupestres" [2].

Si bien el uso del papel para hacer los registros es muy antiguo y su uso fue de gran importancia para almacenar datos, está sujeto a cambios para mejorar el almacenamiento de datos, y un cambio notorio fue el uso de tarjetas perforadoras. El uso de las tarjetas perforadoras para el almacenamiento de datos se introdujo en 1890 por Hollerith [3] con el objetivo de recopilar y almacenar los datos censales de los estadounidenses de aquella época.

Las tarjetas perforadoras funcionaron como un medio de entrada para las computadoras, tanto para programas como para datos, dicha tarjeta contaba con 12 filas y 80 columnas. En cada columna se podía almacenar un carácter, es decir, para representar un 1, se hacía una perforación, el 0 se codificaba sin hacer una perforación[4].

El mecanismo para acceder a la información de la tarjeta perforadora era secuencial, esto dificultaba al momento de hacer consultas, por lo que hacia finales de la década de 1950 surge el disco magnético como nueva forma para almacenar datos, permitiendo el acceso a los datos de manera directa [5].

Con esta tecnología aparecen las bases de datos jerárquicas y en red, que aprovechan la capacidad de acceso directo a la información de los discos magnéticos para estructurar la información en forma de lista enlazadas y árboles de información, dando solución a las consultas *ad hoc* [3].

El punto más importante en las primeras tecnologías de software para manejo de datos fue el desarrollo de `COBOL` y `FORTRAN` en 1960 [6] que fueron lenguajes de programación orientados a los negocios. Cabe destacar que ambos lenguajes no eran sencillos de aprender, por lo que se necesitaba ser un experto para poder manejarlos.

COBOL fue el software utilizado para intentar estandarizar muchos aspectos del procesamiento de datos, esto mediante CODASYL, una organización que reúne a grandes proveedores y usuarios de hardware y software [2].

"El lenguaje de programación BASIC *(Beginner's All-purpose Symbolic Instruction Code)* se crea en el Dartmouth College, en 1964. Fue diseñado con la idea de que todos los estudiantes pudieran trabajar con ordenadores sin necesidad de ser matemáticos o científicos[6]".

CODASYL formó un subgrupo llamado Database Task Group (DBTG) para abordar la cuestión de estandarización para los sistemas de administración de base de datos basados en el modelo en red, mandando una propuesta a *American Nartional Standars Institute* (ANSI), obteniendo una respuesta negativa, siendo este apenas el primer avance que obtenía este subgrupo, siguió intentando hasta que se le atribuyo al modelo CODASYL o el modelo DBTG [2].

Es muy importante recalcar que el modelo CODASYL proporcionó el vocabulario y marco conceptual para discutir los temas de bases de datos, y establecio por primera vez la noción de una arquitectura de bases de datos en capas y terminología común. (Ricardo, 2009)

Para los años setenta se hizo presente el modelo relacional de bases de datos gracias a Edgar Frank Codd marcando un antes y un después en la evolución de las bases de datos. Su influencia fue tan significativa que sus aportes sirvieron de base para que Lawrence Joseph Ellison y otros dos colegas diseñaran uno de los sistemas de gestión de datos más icónicos de todos los tiempos, llamado ahora ORACLE [5].

ORACLE fue la primera compañía en lanzar un producto que utilizaba el inglés basado en SQL. Esto permitió a los usuarios extraer información por sí mismos sin tener que acudir al grupo de sistemas para la obtención de cada informe (George Koch, 1992) [7].

En la década de los ochenta SQL se posicionaba como uno de los primeros Sistemas para la Gestión de Bases de Datos más utilizados según IBM. Ante este escenario de preferencia, el lenguaje SQL se convirtió en el estándar tanto del Instituto Nacional Estadounidense de Estándares (ANSI) como de la Organización Internacional de Normalización (ISO) entre 1986 y 1987 respectivamente [8].

Con la llegada de la década de los noventa, Microsoft Office, lanza su sistema de gestión de bases de datos conocido como Microsoft Access pensado para reunir datos e información desde otras plataformas como Excel.

Con la llegada de los SGBD las empresas aseguradoras utilizaron estos software para gestionar sus bases de datos, tal es el caso de la aseguradora Suiza Zurich que en 1994 inicia un proyecto en la administración de la base de datos de sus clientes que obtuvieron un seguro de vida, esto con la finalidad de dar informes a sus directores en un periodo no muy largo, lo cual implicaba la elaboración de políticas con proyecciones de flujo de efectivo, análisis de ganancias, etc [9]. Registrando así una aplicación de la administración de bases de datos a los seguros de vida.

Hay que tomar en cuenta que existen muchos SGBD pero para nuestro interés en este proyecto se usará MySQL, por la razón de que MySQL ocupa el puesto número 2 en el ranking de SGBD más populares según la página https://db-engines.com/en/ranking

que publicó dicho ranking en noviembre de 2022.

"En México la historia de la profesión Actuarial va de la mano con la creación de las primeras Compañias de seguros en el País."(CONAC, 2021) [10]. Lo anterior no quiere decir que hoy día el Actuario este de tiempo completo en los seguros. Jaime Vázquez Alamilla [11] menciona en su libro "La profesión actuarial"que los nuevos programas de la carrera sustituyen la materia de programación II por manejo de bases de datos, ajustándose al desarrollo tecnológico de la información.

La Universidad Juárez Autónoma de Tabasco (UJAT) a través de la División Académica de Ciencias Básicas (DACB) consideran la opción de crear la licenciatura en Actuaría siendo autorizada en el mes de agosto de 2011; misma que en el 2018 emprende un proceso de reestructuración del plan de estudios entrando en vigor en el ciclo escolar 2021, el cual incluía como materia obligatoria "Manejo de Bases de Datos" para el 3er semestre según CONAC (2021)[9] , este cambio fue realizado gracias a la evaluación que hicieron en 2017 los Comités Interinstitucionales para la Evaluación de la Educación Superior (CIEES) que observó el plan de estudios 2011 y consideró que se tenía una estructura obsoleta que no considera el Sylabus de la Asociación Actuarial Internacional. Aunado a esto, según el plan de estudios 2019 de la UJAT se muestran los resultados de una encuesta sobre "Áreas de conocimiento en las cuales los egresados les hubiera gustado mayor profundidad de enseñanza"mostrando gran interés hacia las bases de datos y SQL(Plan UJAT, 2019).

Es por ello que en este trabajo se busca mostrar al estudiante de la licenciatura en actuaría las herramientas que maneja MySQL como un SGBD para la gestión de bases de datos relacionales.

Mas allá de los aportes antes planteados, la administración de bases de datos es una actividad muy valiosa en el presente y el futuro, conceptos como Artificial Intelligence, Data Mining han logrado que cada día se establezca más interés en la administración de bases de datos.

### 1.1.2. Justificación

La actualidad se ha caracterizado por generar, recopilar y procesar información, se toma en cuenta que en cualquier institución se debe guardar cierta información, tal como clientes, productos, precios, el estado de las finanzas, entre otros.

El crecimiento económico y social ha demandado que muchas instituciones aceleren sus procesos de consultas de datos, esto conlleva a que una vez guardado los datos en una base de datos es necesario tener una buena estructura de los mismos para la implementación de consultas y del mismo modo poder generar reportes de una manera ágil, minimizando el tiempo que esta operación implica.

Según Janet Abbate [4] durante los últimos años tanto el gobierno como organizaciones privadas han destinado gran cantidad de recursos para desarrollar un sistema que administre y gestione datos. La gestión y manipulación de estos es llevada a cabo por los SGBD.

Es por ello que el actuario desarrolla conocimientos analíticos, matemáticos estadísticos y de programación que permiten incursionar al mundo de la gestión de datos. Por gestión de datos Coronel, Morris & Rob se refieren en su libro "Bases de datos" al proceso de crear modelos de datos, mismos que sirven como herramienta de comunicación entre los

usuarios por medio del modelo entidad relación ER y el modelo relacional con la finalidad de hacer consultas a traves de un lenguaje estructurado de consulta (SQL) [12].

El rol de un actuario como gestor de datos es realizar consultas estructuradas en un tiempo óptimo y con el menor costo computacional, con la finalidad de analizar y generar informes de los datos que se generan día a día para la obtención de mejores decisiones dentro de una organización o empresa. Las actividades que realizan esas organizaciones en materia de extracción de datos son un pilar fundamental, que posteriormente se utilizará en algunas áreas tradicionales de la carrera como son: Seguros, pensiones, finanzas, estadística, demografía, administración de riesgos, por mencionar algunas.

De acuerdo con Business Higher Education Forum & PwC Professional la importancia del buen manejo en bases de datos juega un papel importante hoy día, esto se debe a que todos los días, en todas las empresas, en todos los niveles de gestión y operaciones, los empleados necesitan extraer detalles de contratos, arrendamientos, formularios de impuestos, encuestas y otros documentos.

### 1.1.3. Hipótesis

El actuario que se involucre en aprender y desarrollarse en bases de datos tiene que elaborar modelos de datos, normalizar tablas y conocer lenguaje SQL al mismo tiempo debe tener en cuenta lo importante que es considerar que cada dato recolectado es parte de un modelo individual y único para la institución que lo va a desarrollar. De modo que no se debe tomar la información resumida de las entidades para aplicarla de manera general, puesto que dicha información es verdadera para el que hace el modelo y para el entorno que lo rodea.

### 1.1.4. Objetivos

**Objetivo general:**

Lograr que la implementación y el desarrollo de las bases de datos sea una actividad donde se involucren a los actuarios en las diferentes ramas en que se desarrolle; que conozcan la manera en que se estructuran las bases de datos relacionales y la implementación de consultas con lenguaje SQL.

**Objetivo específico:**

- Crear un documento que sirva como referencia para que el alumno de la carrera de Actuaría se adentre en la administración de bases de datos.
- Trabajar y conocer las estructuras de bases de datos relacionales para su implementación en MySQL.
- Conocer el lenguaje estructurado de consultas que maneja MySQL.

# Capítulo 2

# Bases de datos

En este primer capítulo se presentan los conceptos básicos que se necesitarán en el resto del trabajo. De la sección 2.1 a 2.3 se dan a conocer las bases de datos, cómo se clasifican y así mismo se muestra la diferencia de los conceptos entre datos e información. En las secciones restantes (2.4 a 2.6) se especifican las bases de datos relacionales, las cuales se estudiaran a lo largo de esta tesis; los sistemas gestores de datos, así como las ventajas y funcionalidades que se tienen. Este capítulo es importante para que el lector se motive y se le facilite la lectura plasmada en el capítulo 3.

La mayoría de la información de este capítulo fue investigada en las siguientes referencias [13], [14], [15], [16].

## 2.1. Introducción

Una base de datos es una colección de datos que puede ser almacenado de diferentes maneras, como archivos planos, relacionales, orientado a objetos, etc. con el propósito de almacenar y recuperar información[13]. En este trabajo se estudiará las bases de datos relacionales (tablas), tales tablas se relacionan con otras tablas para crear un sentido y dar eficiencia al momento de realizar consultas.

Las bases de datos son ampliamente usadas y se pueden aplicar en organizaciones de todos los tamaños, desde pequeñas empresas (microempresas) y agencias gubernamentales, hasta grandes corporaciones internacionales; a continuación se presentan algunas aplicaciones representativas:

- *Universidades:* Control de la matrícula estudiantil, asignaturas, cursos, maestros y administración en general de la institución.
- *Seguros:* Manejo y control de clientes asegurados, características del seguro y almacenar información necesaria para el cálculo de primas.
- *Finanzas:* Almacenar información de grandes empresas, ventas y compras de bienes, documentos financieros, etc.
- *Recursos humanos:* Guardar información sobre los empleados, salarios, impuestos y prestaciones, así como para la generación de nóminas.

- *Redes sociales:* Conocer las reacciones e interacciones que tiene el usuario con la publicidad de su marca y ver el crecimiento de alguna página.

Con los ejemplos antes mencionados se observa que las bases de datos se usan para satisfacer las necesidades de información de muchas organizaciones e individuos en una variedad de áreas.

## 2.2. Datos vs información

En su forma simple un dato es una medida o característica de un objeto de estudio que no ha sido procesado; y con más precisión es un conjunto de caracteres que pueden ser; numéricos, alfanuméricos, booleanos, ect. La palabra no procesado indican que los datos todavía no han sido sido analizados para revelar su significado, por ejemplo; se puede presentar un dato que indique una temperatura de 40° C, pero no sabemos si representa a la de un ser humano o la de un motor.

Por otro lado, la información es el resultado de procesar los datos en su forma pura para dejar ver su significado. Se debe tomar en cuenta que el procesamiento de datos puede ser muy sencillo como por ejemplo, organizarlos y revelar patrones, o tan complejos como hacer pronósticos o sacar inferencias con el uso de modelos estadísticos.

TABLA_GRUPO

| ID_Grupo | Año | Inscritos | Carrera |
|---|---|---|---|
| 1 | 2019 | 19 | Matemáticas |
| 2 | 2020 | 25 | Actuaría |
| 3 | 2021 | 17 | Física |
| 4 | 2022 | 33 | Economía |

TABLA_PROFESORES

| ID_Profesor | Nombre | Domicilio | Categoría |
|---|---|---|---|
| 1 | Andrés Izquierdo | Filósofos, 301. Col. Tec. Monterrey N.L. | Profesor Investigador |
| 2 | Claudia López | C. Buenos Aires, Col. Centro; Paraíso, Tab. | Profesor Investigador |
| 3 | David Ricárdez | C. Gregorio Méndez, Col. La Ceiba, Matamoros, Tam. | Profesor |
| 4 | Carlos Manuels | Av. Ruiz Cortines, Col. Insurgentes, Cárdenas, Tab. | Profesor |
| 5 | Rafael Cerino | C. Facundo, Pob. Nicolas Bravo, Jalpa, Tab. | Investigador |

TABLA_ASIGNATURA

| ID_Asignatura | Asignatura | ID_Profesor | H_Entrada | H_Salida | ID_Grupo |
|---|---|---|---|---|---|
| 1 | Matemáticas I | 2 | 08:00:00 a. m. | 09:00:00 a. m. | 1 |
| 2 | Física I | 3 | 09:00:00 a. m. | 11:00:00 a. m. | 3 |
| 3 | Álgebra Superior | 4 | 08:00:00 a. m. | 10:00:00 a. m. | 1 |
| 4 | Programación | 1 | 10:00:00 a. m. | 12:00:00 p. m. | 2 |
| 5 | Matemáticas Actuariales I | 2 | 11:00:00 a. m. | 01:00:00 p. m. | 2 |
| 6 | Cálculo I | 1 | 08:00:00 a. m. | 11:00:00 a. m. | 1 |
| 7 | Probabilidad I | 1 | 01:00:00 p. m. | 03:00:00 p. m. | 2 |
| 8 | Matemáticas Financieras | 3 | 12:00:00 p. m. | 02:00:00 p. m. | 3 |
| 9 | Microeconomía | 4 | 09:00:00 a. m. | 11:00:00 a. m. | 4 |
| 10 | Teoría del Seguro | 2 | 03:00:00 p. m. | 05:00:00 p. m. | 2 |

Figura 2.1: Base de datos de una escuela

Los datos que se analizan y se convierten en información valiosa para las entidades o empresas están estrechamente relacionadas con el desarrollo económico y social. La investigación, la planificación y la toma de decisiones exigen una información precisa, oportuna, completa, coherente y adaptada a las necesidades específicas de cada usuario y cada circunstancia [14].

Cuadro 2.1: Datos vs Información

| Datos | Información |
|---|---|
| En la figura 2.1 se muestra 3 tablas que guardan los datos de los profesores, asignaturas y grupos. | En la figura 2.1 se puede obtener información sobre las categorías de los profesores, cargas académicas, horarios disponibles para profesores, grupos con mayor demanda, entre otras características que pida el usuario. |

En el cuadro 2.1 se presenta un pequeño ejemplo de cómo diferenciar datos respecto a la información que se puede obtener al momento de procesar los datos. La información mencionada en el cuadro 2.1 se puede realizar haciendo consultas.

## 2.3. Bases de datos relacionales

En 1970, Edgar Frank Codd publicó un artículo que demostró la importancia de la "condición" de las bases de datos [15]. Codd mencionó que los datos deberían relacionarse mediante características naturales, lógicas, inherentes a los datos, es decir, la idea principal de las bases de datos relacionales (BDR), como lo indica su nombre es el uso de relaciones. Las relaciones que se consideren generarán conjuntos de datos que cumplan ciertos parámetros. Estos conjuntos de datos forman grupos de datos conocidos como tuplas o filas de una tabla.

**¿Por qué usar una base de datos relacional?**

Su principal ventaja es manejar un gran volumen de datos de manera eficiente y ordenada, que además de estar relacionados, mantienen una estructura capaz de poder realizar consultas de forma adecuada. Las bases de datos deben poseer cierta información para que tenga características que la hagan entendible y distinguible.

Las herramientas matemáticas que ayudarán en la administración de las BDR son la lógica y la teoría de conjuntos. Una base de datos relacional esta representada mediante un esquema relacional tal como se observa en la figura 2.2, que representa la base datos de una escuela que se presentó en la figura 2.1.

A la metodología que se usa para diseñar un esquema de bases de datos se llama modelado de datos, la importancia de tener un buen modelado de datos es un pilar fundamental para un administrador de bases de datos, un mal modelado no sería capaz de ser un buen inicio para una administración adecuada en la institución en la que trabaje.

En el modelado de datos se encuentra cimentada toda la gestión de la base de datos, es decir, se encuentra toda la estructura, el cuerpo y la lógica del trabajo. En otras palabras, sin un buen modelado de bases de datos no será posible hacer un buen trabajo.


Figura 2.2: BDR de la figura 2.1

Sin mencionar conceptos avanzados, podemos explicar la relación que existe entre las tres tablas. La tabla profesores esta relacionada con la tabla asignatura, es decir, un profesor puede impartir cero, una o hasta tres asignaturas, pero una asignatura no puede estar relacionada con más de un profesor, esto depende de las políticas de la escuela; lo mismo pasa en la relación que hay entre asignatura y grupo, a cada asignatura le corresponde solo un grupo, pero en un grupo puede haber más de una asignatura.

## 2.4. Sistemas gestores de bases de datos

Se tiene el enunciado de que ningún usuario requiere solo una base de datos. Los usuarios requieren formas, reportes y preguntas basadas en sus datos. Es decir, la cantidad de información que requieren las organizaciones deben ser solicitadas y ejecutadas de forma rápida y efectiva, por lo tanto, es necesario ejecutar las bases de datos en un sistema computarizado capaz de poder administrar e implementar toda consulta requerida por alguna organización. Esto permite introducir el término sistema de administración de datos (DBMS,*database management system*), el cual es una colección de programas que permite a los usuarios crear, mantener y administrar una base de datos.

Algunos sistemas de administración de bases de datos (DBMS) son los siguientes:

- `ORACLE` (1977)
- `SQL Server` (1989)
- `PostgreSQL`(1994)
- `SQLite` (2000)
- `MariaDB` (2009)
- `MySQL` (1995)

El DBMS utilizado para este trabajo es `MySQL`, mismo que servirá como intermediario entre el usuario y la base de datos, es decir, `MySQL` recibe todas las peticiones de aplicación y las traduce en las complejas operaciones requeridas para cumplirlas. Se tiene en cuenta de que se oculta gran parte de la complejidad interna de la base de datos de los programas de aplicación a los usuarios.

¿Qué es y cuál es el significado de `SQL`? Es un lenguaje de programación cuyo significado es

- S → *Structured*,
- Q → *Query*,
- L → *Language*,

es decir, `SQL` significa lenguaje para realizar consultas estructuradas. En capítulos posteriores se introducirá al lenguaje `SQL`.

## 2.5. Funciones de un DBMS

- *Administración de un diccionario de datos:* guarda definiciones de los elementos de datos y sus relaciones (metadatos), es un diccionario de datos.
- *Administración de almacenamiento de datos:* crea y maneja las complejas estructuras requeridas para el almacenamiento de estos datos.
- *Transformación y representación de los datos:* transforma los datos introducidos para apegarse a las estructuras de datos requeridas.
- *Administración de la seguridad:* crea un sistema de seguridad que hace cumplir la seguridad del usuario y medida a los datos.
- *Control de accesos múltiples:* sirve para dar integridad y consistencia.
- *Administración de respaldo y recuperación:* suministra respaldo y recuperación de datos para garantizar la seguridad e integridad.
- *Administración de la integridad de datos:* promueve y hace cumplir las reglas.
- *Lenguaje de acceso a bases de datos:* proporciona acceso a los datos por medio de un lenguaje de consulta `SQL`.
- *Interfaz de comunicación:* acepta la petición de los usuarios finales hechas a través de múltiples y diferentes ambientes de red.

Dada las funciones de un DBMS se tienen ventajas como: mayor seguridad, integridad, acceso, mínima inconsistencia de datos, mejor toma de decisiones y una mayor productividad con el usuario.

# Capítulo 3

# Actuaría y su relación con las bases de datos

En este capítulo se presenta la relación que hay entre la actuaría y el manejo de bases de datos, se muestran algunas aplicaciones en diferentes campos científicos y se hace mención de algunas técnicas para la estructura y el análisis de datos. Lo presentado en este capítulo juega un papel importante para que el lector se motive en profundizar investigaciones en diferentes áreas cómo pueden ser las ciencias actuariales, políticas, entre otras.

La mayoría de la información presentada en este capítulo fue investigada en las siguientes referencias [11], [1], [17], [18], [19], [20], [21] , [22], [23].

## 3.1. Introducción

La carrera de Actuaría se encarga en preparar profesionales para la evaluación, identificación y prevención de riesgos a través de análisis de datos, algoritmos y estadística.

La SOA (Society of Actuaries) menciona que los actuarios son profesionales muy solicitados por el hecho de que ayudan a líderes a tomar decisiones estratégicas y a los consumidores a prepararse para su futuro. Pero, ¿qué es la profesión actuarial? y ¿que relación tiene con las bases de datos?.

La Licenciatura en Actuaría tiene por objeto la formación de profesionales de alto nivel académico en la aplicación de la matemática para evaluar la ocurrencia de eventos catastróficos, como por ejemplo, pago de daños materiales, caídas en la bolas de valores, paridad peso-dolar, mismas que están dentro de las áreas de seguros y finanzas. En este sentido, la carrera de Actuaría es la aplicación de las matemáticas dentro del ámbito de los sistemas de seguros y las finanzas. Un actuario es un profesional que utiliza sus aptitudes matemáticas para definir, analizar y resolver problemas sociales y financieros. El Actuario contribuye a diseñar y administrar planes de seguro de pensiones y evaluar el riesgo financiero en que incurren las empresas, entre otras funciones. Universidad Juárez Autónoma de Tabasco (UJAT) `https://www.ujat.mx/332`

Es un hecho que durante la licenciatura se desarrollen habilidades de programación y del uso de softwares. Esto se ve reflejado en las actualizaciones que ha tenido la carrera con su plan de estudios. Jaime Vázquez [11] menciona que el actuario se ve inmerso en el

mundo de las ciencias computacionales gracias a la formación lógico-estructurada que le brindan las matemáticas al momento de resolver y analizar modelos. Enfatiza mucho la automatización de procesos que se obtienen en tres pasos:

1. Gestión de bases de datos: manejo de SQL
2. Análisis de datos: identificar tendencias, realizar inferencias estadísticas, realizar proyecciones, etc.
3. Desarrollo de algoritmos: Etapa final de automatización.

Por lo tanto, todo estudiante de actuaría debe tomar en cuenta que durante la carrera se necesita desarrollar diferentes habilidades, mismas que en conjunto se combinan para desarrollar modelos.

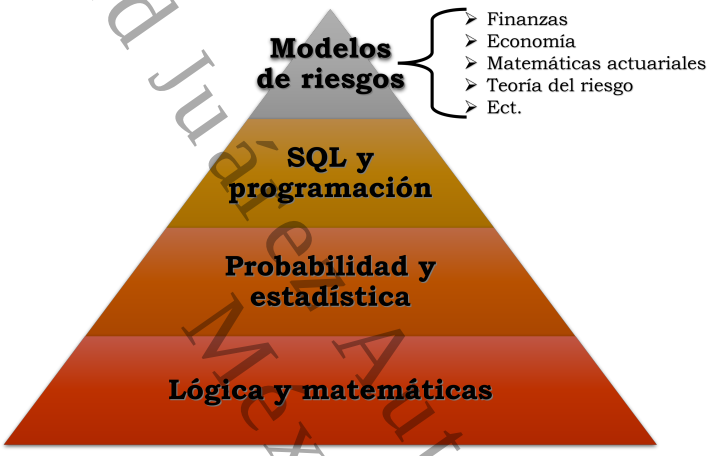



Figura 3.1: Pirámide del conocimiento para todo Actuario

En la figura 3.1 se muestra la importancia escalonada que tiene un actuario para desarrollar modelos, siendo la lógica y matemáticas la base para poder desarrollar habilidades de probabilidad y estadística. Aunque el desarrollo de programación y `SQL` puede ir antes que la probabilidad y estadística, es un hecho que para poder realizar modelos de riesgos se necesita tener habilidad en `SQL` y programación, ligado a esto, también es un hecho que en los cursos finales de probabilidad y estadística ya hace uso de la programación.

Existen diferentes modelos de riesgos que dependen de la disciplina en la que se trabaje. Para desarrollar un modelo en ocasiones se necesita obtener datos para su desarrollo, tal es el caso de un modelo económico que usa series de tiempo, un modelo para tablas de mortalidad o para el caso de las finanzas realizar proyecciones de ventas.

La generación de datos es un recurso muy importante que ha tomado mucho auge en los últimos años, lo cual ha permitido a muchos actuarios ser pioneros en el desarrollo de nuevas iniciativas, tal es el caso de AMIS cuando hace mención de un almacén de datos desarrollado por *Zurich Insurance Group* con el fin de poder realizar reportes regulatorios con facilidad y poder dar soporte a los flujos de efectivo para generar reservas matemáticas.

Aleksey Popelyukhin [1] hace mención a la revolución de la información y enfatiza la rapidez en la que cada vez se generan cientos de datos, mismos que necesitan ser analizados

y resumidos para su comunicación de forma casi inmediata, con la finalidad de tomar decisiones rápidas.

De este modo, un actuario tiene que estar preparado para el manejo y explotación de datos, la relación que existe entre la gestión de datos y la actuaría es muy estrecha por el hecho de que un actuario necesita recopilar y analizar datos para poder realizar pronósticos financieros y evaluaciones de riesgos. Por lo tanto, la gestión de bases de datos es esencial para la profesión actuarial y es una de las herramientas principales para analizar y procesar grandes cantidades de información con la finalidad de tomar decisiones formuladas, informadas y precisas.

A.S. Popelyukhin clasifica 3 procesos que todo actuario debe realizar para obtener un buen trabajo independientemente del área en que se especialice.

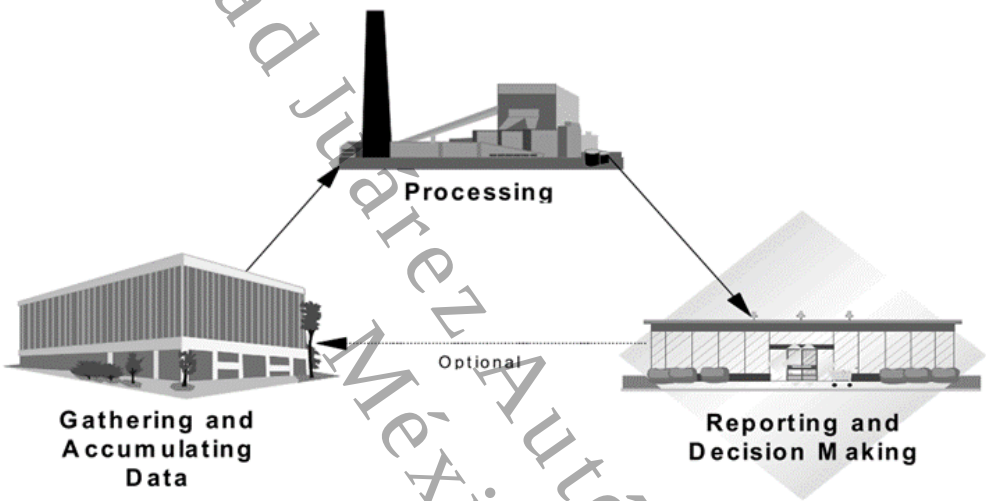



Figura 3.2: Aleksey Popelyukhin y los procesos de un actuario [1]

1. Entrada (gathering and accumulating data).
   Se refiere a la recopilación de datos y construcción del modelo. En este paso el actuario debe ir a los puntos donde se generen de datos, pueden ser datos sobre contrataciones de seguros, reclamaciones, solicitudes de créditos, detalle de operaciones bursátiles, ect.

2. Cálculo (processing).
   Este proceso depende mucho de la área con el que se esté trabajando, por ejemplo, no se puede realizar un mismo análisis y cálculo para datos que tienen que ver con ventas de algún producto contra otros datos que incluyan los rendimientos en gasolina de un lote de carros.

3. Reporte (reporting and decision making).
   Una vez que los datos han sido recopilados, refinados y desarrollados para su cálculo de acuerdo al modelo que se realice, se pasa a la etapa final de comunicarlos a las personas con la finalidad de tomar decisiones para el futuro.

Las herramientas para extraer, validar y analizar datos son componentes esenciales de lo que se conoce como almacén de datos o "data warehouse" (DW).

**¿Qué es un data warehouse?** La definición proporcionada por Bill Inmon [17] en 1992 con su libro *Building the Data Wherehouse* se refiere a una colección de datos orientados a

un tema, integrados, no volátiles e historiados, organizados para ofrecer apoyo a procesos de ayuda a la decisión.

De esta definición, se deriva el hecho de que se trata de un tipo de bases de datos, cuya importancia reside en el apoyo que puede ofrecer a las organizaciones desde el punto de vista estratégico y que, en primera vista no parece muy complejo. Sin embargo, la dificultad comienza al momento de crear el almacén de datos, sin saber, *a priori*, qué datos se necesitan y de qué manera deben organizarse.

Otra característica de un DW es que centraliza y fusiona grandes cantidades de datos de múltiples fuentes. Gracias a sus capacidades analíticas, los actuarios pueden obtener información empresarial valiosa a partir de los datos y mejorar decisiones.

Con el tiempo, se construye un registro histórico de gran valor tanto para los actuarios como para los analistas de negocios y especialistas en el área. Gracias a estas fusiones, un DW puede considerarse "La fuente única de datos" de una organización.

## 3.2. ¿Qué es la administración de datos?

La administración de datos consiste en recopilar, mantener y utilizar datos de manera segura, eficiente y rentable. El objetivo de la administración de datos es ayudar al usuario, que involucra desde las personas físicas hasta las organizaciones privadas y gubernamentales a optimizar el uso de los datos dentro de los límites de las políticas y normativas, para que puedan tomar decisiones y medidas que maximicen el beneficio de sus intereses.

Las actividades para la administración de datos son realizadas por profesionistas que dominan el lenguaje `SQL` de bases de datos y dentro de sus actividades se encuentran:

- Definir el esquema conceptual: Consiste en elaborar el diseño lógico o conceptual de la base de datos. En este punto se define la información que contendrá y como se organizará en el DBMS.
- Definir el esquema interno: Consiste en decidir la forma en que se van a representar los datos en la base de datos almacenada. A este proceso se le conoce como el diseño físico.
- Establecer un enlace con los usuarios: Se define el enlace con los usuarios para asegurar que los datos necesarios estén disponibles para escribir los esquemas necesarios, utilizando el lenguaje de definición de datos (DDL) externo aplicable.

  Otros aspectos de la función de enlace con los usuarios incluyen la asesoría sobre el diseño de aplicaciones; una capacitación técnica; ayuda en la determinación y resolución de problemas; así como otros servicios profesionales similares.
- Definir las restricciones de seguridad y de integridad: Las restricciones de seguridad y de integridad pueden ser vistas como parte del esquema conceptual. El DDL conceptual debe incluir facilidades para especificar dichas restricciones.
- Definir las políticas de vaciado y recarga: Para evitar errores en el sistema o se provoquen inconsistencia de datos, es necesario implementar un esquema apropiado al control y buen manejo de los datos. Es decir, identificar los riesgos e implementar controles para el buen manejo de los datos.

- Supervisar el rendimiento y responder a los requerimientos cambiantes: Se organiza el sistema de tal manera que se obtenga el rendimiento "ideal para la institución" y de hacer los ajustes apropiados, es decir, afinar conforme las necesidades cambien.

La administración de datos es una disciplina que se concentra en la adecuada generación, almacenamiento y recuperación de datos. Dado el papel esencial que desempeñan los datos, no debe sorprender que la administración de datos sea una actividad de máxima importancia para cualquier negocio, agencia gubernamental, organización de servicio o instituciones.

## 3.3. El rol de un actuario en las bases de datos

Cualquier institución que vea a los datos como un recurso fundamental para su crecimiento, le brinda una ventaja sobre las que no lo hacen. Los beneficios de esta atención les permite tener control de sus procesos y registros para así cumplir sus objetivos específicos y que pueden estar asociados a diversos aspectos, por ejemplo: mayor transparencia en su funcionamiento, mejor manejo de inventarios, un aumento en los ingresos de ventas, una disminución en costos, detección de lavado de dinero, aumentos en productividad, entre otros aspectos.

De acuerdo con la información de empleos con mayor auge en el 2023 de LinkedIn [18], donde se muestran los 15 profesionales que han incrementado su participación en los últimos cinco años y las tendencias que están definiendo el futuro del mercado laboral; posiciona en cuarto lugar al especialista en la gestión de datos que tiene como objetivo poner orden al caos de los datos; administrándolos, procesándolos y almacenándolos en sistemas y arquitecturas.

Las actividades que el actuario realiza en las instituciones en materia de gestión, extracción y análisis de datos tienen que ver principalmente con áreas relacionadas a Seguros, Pensiones, Finanzas, Estadística, Demografía, Banca, Administración de riesgos, por mencionar algunos.

A continuación se mencionan algunos ejemplos de la importancia de la administración de bases de datos en diferentes áreas:

**Sistemas de marketing**

Un análisis de marketing implica realizar un trabajo analítico y experto, con base en el conocimiento exhaustivo de clientes, productos, canales de distribución y mercado. Este conocimiento se deriva de la disposición que se tenga de una base de datos para poder recabar lo necesario y predecir variables, con el fin de tomar decisiones certeras.

Tal es el caso de un marketing digital que realiza un análisis de los clicks que dan a su página, las interacciones, las visualizaciones de videos y el alcance de sus publicaciones. Mismo que en conjunto permite al que realiza el estudio, poder clasificar los gustos de los clientes para poder mandarle publicidad o realizar ofertas.

**Seguros**

En cualquier institución de seguros se necesitan actuarios capaces de desarrollar modelos de riesgos y así poder desarrollarlos, cuantificarlos y estimarlos. En la mayoría de los

modelos se necesita extraer cierta información de los clientes para así poder calcular la prima o también extraer información de siniestros para calcular la reserva matemática.

Un ejemplo de aplicación de bases de datos en los seguros pasó en el departamento actuarial de negocios del Grupo Zurich Insurance en el año 1993, cuando el departamento observó que necesitaba un almacén de datos o DW sobre sus productos con el objetivo de poder emitir reportes regulatorios y poder calcular sus reservas. Dicho trabajo fue realizado por actuarios, desarrolladores de sistemas y programadores.

**Finanzas**

La gran cantidad de datos generadas por transacciones financieras, así como la información que generan los mercados financieros, son un elemento muy importante para las instituciones de banca a la hora de realizar sus próximas operaciones en el mercado.

Los cambios de tasas, los valores de acciones, los instrumentos de deuda y la inflación, son algunos de los datos que requiere cualquier institución financiera a la hora de operar. Según Mike Loukides [19] el análisis y la ciencia de datos se ha convertido en una herramienta importante para las finanzas, porque permite el análisis y modelado estadístico de datos financieros para medir el riesgo, la rentabilidad y la eficiencia del mercado.

**Economía**

En el ámbito económico se generan muchos datos que sirven tanto para dependencias gubernamentales como para empresas privadas que requieren un análisis de mercado, estimaciones de parámetros económicos, entre otros aspectos.

Para las dependencia gubernamentales las bases de datos pueden contener información sobre variables macroeconómicas, como el PIB, la tasa de desempleo, la inflación, el comercio internacional y otros indicadores económicos que son fundamentales para la evaluación económica de un país.

Por otro lado, para instituciones privadas, las bases de datos pueden contener información de variables microeconómicas, tal es el caso cuando se aplica función de costos, de producción, función de demanda y otros tipos de análisis que son importantes para llevar el control de una empresa.

En resumen, las bases de datos son una herramienta fundamental para la evaluación y el análisis de la economía, su uso es indispensable porque permite obtener proyecciones de crecimiento y mejores toma de decisiones efectivas en la economía.

**Salud**

Para las ciencias de la salud, las bases de datos permiten recopilar, almacenar, organizar y acceder a grandes cantidades de información relacionada con la salud y la medicina, como estudios clínicos, registros médicos, estadísticas epidemiológicas, entre otros.

Con la gestión de las bases de datos, los actuarios pueden desarrollar sus habilidades al encontrar información actualizada sobre tratamientos, terapias, medicamentos, condiciones médicas y avances en la investigación médica. Además, estas bases pueden utilizarse para la gestión de recursos, epidemiología, planeación, evaluación, investigación y para la toma de decisiones clínicas y administrativas en el ámbito sanitario.

Tal fue el caso en la pandemia COVID-19, donde se informaban los casos diarios, al mismo

tiempo que mostraban indicadores como incidencia, prevalencia y mortalidad. También se realizaban gráficas que mostraban ciertas tendencias que ayudaban a los epidemiólogos a estimar los casos. Es por eso que las bases de datos tienen importancia en el sector salud, ayudando al personal a tomar mejores decisiones y mejorar la atención médica para los pacientes.

## 3.4. Breve descripción del análisis de datos en la actuaría

Una vez desarrollado habilidades para la gestión de bases de datos, es necesario complementar esta habilidad con técnicas de análisis de datos.

Javier García [21] menciona que gran parte del proceso de análisis de datos se consume en las transformaciones que tienen que aplicarse a los datos para que puedan ser procesados y presentados de forma inteligible y clara. Por lo tanto, la gestión y el análisis de datos son procesos clave para el éxito de una buena toma de decisiones.

### 3.4.1. Estructura básica de datos

Es importante poder identificar la manera en que se introducen los datos, siendo el formato tabular el más común para realizar el análisis. Esta organización de los datos es útil para los algoritmos de clasificación, agrupamiento, extracción de reglas, regresiones, redes de neuronas artificiales, donde la entrada de datos requiere de un vector n-dimensional de atributos, homogéneo para todos los registros.

Las estructuras básicas de datos más utilizadas son:

- Vectores: conjunto de elementos del mismo tipo, son de una dimensión y pueden contener solo un tipo de dato. Los datos en un vector pueden estar en forma de fila o columna.
- Matrices: es una extensión natural de vectores, es decir, n-dimensional, es aplicable cuando se quiere tener más de una fila o columna de datos, para este caso cada columna o fila es independiente del tipo de dato que se obtenga.
- Tablas: una tabla contiene filas y columnas que guardan información de la vida real.
- Listas: una lista de datos es una estructura de datos residente en la memoria que se llena con un conjunto de nombres extraídos de una fuente externa, como por ejemplo un archivo sin formato. La lista es utilizado cuando el conjunto de datos es heterogéneo.
- Hoja de datos (*dataframe*): Se utiliza cuando el conjunto de datos son de tipo heterogéneos por columnas, pero todos los elementos de una misma columna tienen el mismo tipo. Es la estructura más utilizada para organizar los datos.
- Factores: son utilizados para el modelado estadístico, normalmente se emplean para variables de tipo nominal o categóricas. Internamente se tratan como un vector de enteros que referencia al conjunto de factores variables.

### 3.4.2. Técnicas estadísticas para el análisis de datos

La estadística ayuda a extraer conocimientos de los datos mediante el uso de métodos y técnicas que son implementadas en sistemas computacionales. Las bases de datos proporcionan una forma de almacenar y administrar grandes cantidades de datos. Combinar la estadística con bases de datos permite un análisis amplio de los datos.

Sheldon Ross [22] define a la estadística como el arte de aprender a partir de los datos y se ocupa de la recolección, agrupación, presentación, análisis e interpretación de datos para obtener conclusiones.

La mayor parte de las decisiones se toman con la asistencia de información reunida acerca del mercado, el entorno económico y financiero, la fuerza laboral, la competencia y otros factores. Esta información suele llegar en forma de datos que se suelen recolectar y analizar con ayuda de la estadística, misma que puede estar dividida en dos categorías: estadística descriptiva y estadística inferencial.

**Estadística descriptiva**

Mario Rendón-Macías [23] menciona que la estadística descriptiva es la rama de la estadística que formula recomendaciones sobre cómo resumir la información en cuadros o tablas, gráficas o figuras.

En esta rama normalmente se estudian variables cuantitativas (métricas) y variables cualitativas (categóricas).

Las primeras se definen por la existencia de una unidad de medición, que pueden ser contables (unidades enteras) medible o ponderada por algún atributo.Asimismo, pueden ser clasificadas como continuas si aceptan fracciones, o discretas si solo consideran unidades enteras.

Las variables cualitativas se caracterizan por clasificar a los individuos o fenómenos solo con relación a sus atributos. Pueden ser nominales cuando los atributos usados son únicos para una condición (excluyentes mutuamente) y solo existen posibilidades conocidas (exhaustivas). Si el atributo solo acepta dos condiciones, las variables reciben el nombre de nominales dicotómicas, pero si hay más posibilidades se les denomina nominales politómicas.

Los estimadores que se usan mayormente para la estadística descriptiva son:

- Medidas de tendencia central.
  - Promedio.
  - Mediana.
  - Moda.
- Medidas de dispersión.
  - Rango de variación.
  - Varianza.
  - Desviación estandar.
  - Coeficiente de variación.

**Inferencia estadística**

Para Carlos Quintana Ruiz [20] la inferencia estadística busca obtener conclusiones y hacer generalizaciones válidas a la población bajo estudio, a partir de la información generada por una muestra seleccionada al azar de tal población.

En muchos casos el problema de la inferencia consiste en estimar un parámetro de la población o verificar un supuesto establecido acerca de alguna característica de la población y tomar una decisión al respecto.

Los parámetros y algunas pruebas que aplican para la inferencia estadística son:

- Modelos de distribución de probabilidad.
- Estimación de parámetros por mínimos cuadrados.
- Análisis de regresión.
- Pruebas de hipótesis.
- Análisis de la varianza (ANOVA)
- Series de tiempo.

La estadística juega un papel complementario en la administración y análisis de datos. Este trabajo solo se enfoca en presentar la primera etapa que se lleva a cabo a la hora de crear modelos, esta etapa se llama administración de las bases de datos. Sin embargo, es necesario que el estudiante comprenda el camino que queda por recorrer al momento de aprender a gestionar datos.

# Capítulo 4

# Modelado de datos

En este capítulo se definen conceptos y procesos clave para el modelo de datos; se da a conocer los niveles de abstracción que tienen los datos, las reglas de integridad, hasta los tipos de relaciones que existen en las bases de datos así como el proceso de normalización. Éste capítulo es la parte medular del trabajo, la razón es porque con el conocimiento presentado se pueden elaborar nuevos proyectos que involucren la administración y el análisis de datos.

El diseño de una base de datos se concentra en la forma en que un administrador de bases de datos quiera analizar, gestionar o guardar ciertos datos que serán tomados del mundo real. Por lo tanto, se tiene que tener un buen diseño para tener éxito en los resultados.

Un modelo de datos es un diseño elaborado que permite la representación de estructuras de datos reales, es decir, es una abstracción de un objeto o hecho real tomado de la naturaleza que es representado en datos. La función principal de un modelo es ayudar a que el usuario entienda las complejidades del ambiente real. El modelo de datos utiliza conceptos lógicos como objetos, sus propiedades y sus relaciones.

Las referencias que fueron consultadas y que son citadas en éste capítulo son las siguientes: [14], [12], [16].

## 4.1. Abstracción de los datos

Los DBMS necesitan tener una descripción o definición de la DB. Esta descripción recibe el nombre de **esquema de la DB**.

Al igual que el cerebro humano entiende las fórmulas matemáticas para entender un suceso real mostrado por un sistema de ecuaciones; el DBMS necesita un modelo, llámese esquema, que permita entender el manejo la DB.

Todas las bases de datos tienen estructuras de datos complejas que, de hecho, no son útiles para un usuario final. Por lo tanto, estos detalles irrelevantes internos están ocultos para el usuario, lo que simplifica el acceso a los datos y también aumenta la seguridad de los datos. Esto representa la abstracción de datos.

A principios de los años 70, el Standards Planning and Requirements Committee (`SPARC`) del American National Standards Institute (`ANSI`) definió una estructura para modelar

datos con base en niveles de abstracción de datos[12]. Dichos niveles son:

- *Nivel físico (nivel interno):* Es el nivel más bajo de abstracción, describe *¿Cómo?* se almacenan realmente los datos. Se representa por un esquema interno que contendrá la descripción de la organización física de la DB; caminos de acceso (índices, hashing, apuntadores, etc), codificación de los datos, gestión del espacio, etc. En palabras simples, es la representación de la base de datos de como es "vista" por el DBMS.
- *Nivel lógico (nivel conceptual):* El siguiente nivel de abstracción describe *¿Qué?* datos se almacenan en la DB y qué relaciones existen entre esos datos. Se representa por un esquema conceptual el cual se definen las entidades, atributos y reglas de integridad.
- *Nivel de vistas (nivel externo):* El nivel más alto de abstracción describe solo parte de la base de datos completa y se representa por un esquema externo que muestra al usuario final solo aquellos atributos y aquellas entidades que interesan.

**Ejemplo:** La consultoría AIM realiza estudios estadísticos, administra riesgos y da soporte financiero a instituciones privadas y gubernamentales. Se quiere implementar un esquema para la base de datos de AIM. A continuación se tiene un resumen de los tres niveles de abstracción para la consultoría AIM.

Cuadro 4.1: Ejemplo: niveles de abstracción

| **Nivel** | **Ejemplo** |
|---|---|
| Nivel físico | El Director General de la consultoría AIM decide comprar hardware para poder almacenar los datos de su consultoría. De modo que se puedan acceder a todos los datos de los clientes sin procesar ni modificar. |
| Nivel lógico | La consultoría AIM necesita dar forma y estructura a los datos almacenados. Por lo tanto, se definen entidades como; Clientes, Servicios, Empleados, etc. Al igual que atributos, relaciones, ocurrencia, reglas de integridad, etc. |
| Nivel vistas | El cliente pide un historial de sus finanzas, por ejemplo; utilidades en cierto trimestre, meses con más ventas, cantidad de presupuesto, etc. |

De este modo se concluye que el nivel físico es la capa donde los datos sin procesar se almacenan en formato de archivo en discos duros físicos o en virtuales; el nivel lógico es la capa donde los datos sin procesar del nivel físico, toman una estructura y se organizan de forma adecuada, como puede ser un formato tabular y por último se tiene el nivel vista, donde los usuarios finales obtienen información basada en sus consultas [12].

## 4.2. Modelo de los datos

El **modelo de los datos:** es un conjunto de herramientas conceptuales diseñadas para representar los datos, sus relaciones, su significado y las restricciones que garantizan su consistencia. Para ilustrar el concepto de un modelo de datos, describiremos dos modelos de datos en este trabajo: **modelo entidad-relación** y **modelo relacional**.

### 4.2.1. Modelo entidad relación (E-R)

Se desarrolló para facilitar el diseño de bases de datos permitiendo la especificación de un esquema de la empresa que representa la estructura lógica completa de una base de datos [14]. El modelo de datos E-R es uno de los diferentes modelos de datos semánticos; el aspecto semántico del modelo viene definido con varios conceptos que definen una tabla. Por ejemplo:

Las **entidades** se identifican como representaciones de personas, lugares, eventos, objetos o conceptos de los cuales se recopilan datos.Cada entidad tiene ciertos **atributos**, que son características o propiedades que permiten describir la entidad hasta el punto que desee el usuario. Para que la base de datos tenga sentido, ésta tiene que ser capaz de registrar las **ocurrencias** de algún evento que sea de interés para el usuario.

| TABLA_VENTAS | | | | | | |
|---|---|---|---|---|---|---|
| ID_Venta | ID_Vendedor | Código_Producto | Producto | Cantidad | Fecha | Sucursal |
| 1 | 4 | 2re2 | Gorra | 1 | 01/05/2023 | 1 |
| 2 | 1 | 4er8 | Pantalón | 2 | 06/05/2023 | 3 |
| 3 | 3 | 8df5 | Camisa | 1 | 12/05/2023 | 2 |
| 4 | 2 | 5bn9 | Camisa Polo | 3 | 28/05/2023 | 3 |
| 5 | 4 | 2cx7 | Short | 5 | 01/06/2023 | 1 |

Figura 4.1: Registro de ventas

En el ejemplo mostrado en la figura 4.1, la entidad es VENTAS, el conjunto de atributos es: ID_Venta, ID_Vendedor, Código_Producto, Producto, Fecha, Sucursal. Para la ocurrencia, se muestran 5 ocurrencias, es decir, 5 ventas realizadas.

Algunas entidades pueden tener **relaciones** o asociaciones con otras entidades, por ejemplo, se puede dar el caso que la entidad VENTAS, esté relacionada con otra entidad que se llame VENDEDOR, donde se registren los datos de los vendedores que hay en una institución. Otro concepto importante son las **restricciones**, que limitan hasta cierto punto qué tan tolerable serán las ocurrencias para que sean registradas, por ejemplo, en la entidad VENTAS, en el atributo Sucursal no puede haber más de 40 registros que estén en la misma sucursal y estén ligadas al producto Pantalón.

La estructura lógica general de una base de datos se puede expresar gráficamente mediante un diagrama E-R, que consta de las siguientes componentes básicos:

- *Rectángulos:* representan conjuntos de entidades.
- *Elipses:* representan atributos.
- *Rombos:* representan relaciones entre conjuntos de entidades.
- *Líneas:* unen los atributos con los conjuntos de entidades y los conjuntos de entidades con las relaciones.

**Ejemplo:** Una aseguradora decide organizar una pequeña base de datos considerando dos entidades; CLIENTES y SEGUROS. El administrador de datos decide hacer un diagrama E-R como el de la figura 4.2 para mostrarlo a sus jefes. En el diagrama se muestra que hay una relación llamada CONTRATA entre las entidades, la cual nos puede ayudar para realizar diferentes tipos de consultas como puede ser; clientes maestros que adquieren

un seguro de vida, arquitectos que obtienen un seguro de RC-Profesional, médicos que adquieren un seguro de vida dotal-mixto, entre otros.

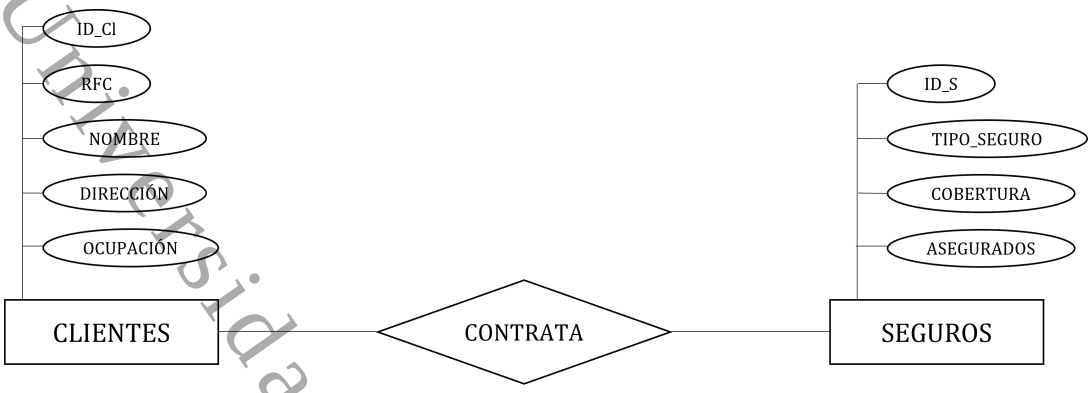



Figura 4.2: Diagrama E-R

### 4.2.2. Modelo relacional

El modelo relacional emplea un conjunto de tablas para representar tanto los datos como las relaciones entre ellos. Cada tabla consta de múltiples columnas, las cuales poseen nombres únicos. Este modelo es un ejemplo de un enfoque basado en registros.

Según Surdarshan [14] el modelo relacional se encuentra en un nivel de abstracción inferior al modelo de datos E-R. Los diseños de bases de datos a menudo se realizan en el modelo E-R y después se reducen al modelo relacional.

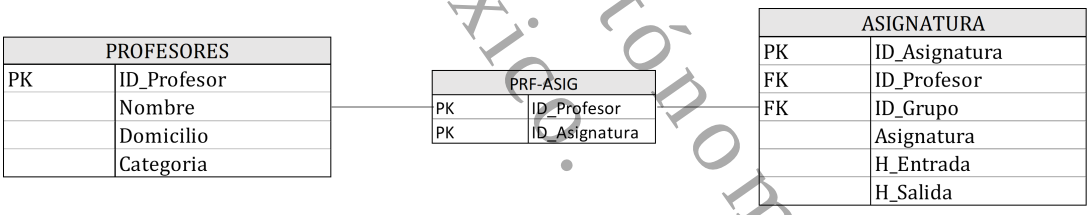



Figura 4.3: Diagrama R

La terminología difiere cuando se plantea el modelo relacional. Así, en ocasiones los renglones u ocurrencias se mencionan como **tuplas**, las columnas o atributos se indican como **campos** y a las tablas o entidades se les llama **archivos**. No obstante, mientras el lector entienda que el modelo en si representa una construcción lógica mas que física, se pueden considerar como antes se definió.

En el diagrama relacional de la figura 4.3 se muestran 2 entidades (PROFESORES, ASIGNATURA) y una una tabla más entre ambos llamada PRF-ASIG que muestra la relación que hay entre las entidades, más adelante se mostrará que cuando existe relación se pueden realizar consultas.

## 4.3. Conceptos de diseño

Para realizar un buen modelo de DB es necesario conocer la terminología y su significado, de modo que estudiar los conceptos y entenderlos será crucial para el desarrollo e

implementación del modelo.

### 4.3.1. Llaves

En el modelado de datos, las llaves o etiquetas únicas son importantes porque se usan para asegurar que cada renglón de la tabla sea identificable de manera única. También se usa para establecer relaciones entre las entidades que conforman la base de datos. Por lo tanto, un apropiado entendimiento del concepto y uso de las llaves en el modelo relacional es muy importante. Una llave esta formada por uno o más atributos que determinan a otros atributos.

En la figura 4.4 se muestra un ejemplo de llave para la columna ID_Asignatura, donde se observa que para cada asignatura le corresponde un número que identifica la fila de manera única, es decir, dicho número no se repetirá jamas en la tabla. Otra observación que resaltar es que hay dos columnas que tienen la función de relacionar la tabla con otra, tal es el caso de ID_Profesor y ID_Grupo. Mas adelante se verán a detalle estos casos.

| TABLA_ASIGNATURA | | | | | |
|---|---|---|---|---|---|
| ID_Asignatura | Asignatura | ID_Profesor | H_Entrada | H_Salida | ID_Grupo |
| 1 | Matemáticas I | 2 | 08:00:00 a. m. | 09:00:00 a. m. | 1 |
| 2 | Física I | 3 | 09:00:00 a. m. | 11:00:00 a. m. | 3 |
| 3 | Álgebra Superior | 4 | 08:00:00 a. m. | 10:00:00 a. m. | 1 |
| 4 | Programación | 1 | 10:00:00 a. m. | 12:00:00 p. m. | 2 |
| 5 | Matemáticas Actuariales I | 2 | 11:00:00 a. m. | 01:00:00 p. m. | 2 |
| 6 | Cálculo I | 1 | 08:00:00 a. m. | 11:00:00 a. m. | 1 |
| 7 | Probabilidad I | 1 | 01:00:00 p. m. | 03:00:00 p. m. | 2 |
| 8 | Matemáticas Financieras | 3 | 12:00:00 p. m. | 02:00:00 p. m. | 3 |
| 9 | Microeconomía | 4 | 09:00:00 a. m. | 11:00:00 a. m. | 4 |
| 10 | Teoría del Seguro | 2 | 03:00:00 p. m. | 05:00:00 p. m. | 2 |

Figura 4.4: Tabla Asignaturas

La función de una llave está basada en un concepto conocido como **determinación**. Por ejemplo, la entidad ID_Asignatura puede buscar (determinar) el nombre de la asignatura, Grupo o Profesor, es decir, en el contexto de una base de datos, el enunciado "A determina B" indica que si se conoce el valor del atributo A, se puede buscar (determinar) el valor del atributo B.

El principio de determinación se utiliza en la definición del concepto central de una base de datos relacional conocida como **dependencia funcional**, que se define como:

**El atributo B es funcionalmente dependiente del atributo A si cada valor de la columna A determina un y solo un valor de la columna B.**

En la figura 4.4, es apropiado decir que ID_Profesor es funcionalmente dependiente de ID_Asignatura. Por otra parte se tiene que ID_Asignatura no es funcionalmente dependiente de ID_Profesor porque el 1, 2, 3 y 4 tienen varios valores que representan las diferentes asignaturas que dan los 4 profesores.

La definición de dependencia funcional se pude generalizar para cubrir el caso en el que determinan valores de atributos que represente más de una tabla. La dependencia funcional puede definirse de este modo:

**El atributo A determina el atributo B (esto es, B es funcionalmente dependiente de A) si todos los renglones de la tabla concuerdan en valor para el**

**atributo A, también concuerdan en valor para el atributo B.**

Recuerde que podría ser necesario más de un solo atributo para definir dependencia funcional; esto quiere decir que una llave puede estar compuesta con más de un atributo. Esa llave de atributos múltiples se le conoce como **llave compuesta.**

Cualquier atributo que sea parte de una llave se conoce como **atributo llave**.

Dada la posible existencia de una llave compuesta, la noción de dependencia funcional se puede refinar más si se especifica una **dependencia funcional completa.**

**Si el atributo (B) es funcionalmente dependiente de una llave compuesta (A), pero no de cualquier subconjunto de esa llave compuesta, el atributo (B) es total y funcionalmente dependiente de (A).**

Dentro de la amplia clasificación de llaves, se pueden definir varias especializadas. En el cuadro 4.2 se muestran las llaves para una base de datos relacional.

Cuadro 4.2: Llaves de base de datos relacional

| Tipo de clave | Definición |
|---|---|
| **Superllave** | Un atributo (o combinación de atributos) que de modo único identifica cada renglón en una tabla. |
| **Llave candidata** | Una superllave mínima (irreductible). Una superllave que no contiene un subconjunto de atributos que no sea en sí una superllave. |
| **Llave primaria** | Una clave candidata seleccionada para identificar de modo único todos los valores de atributo en cualquier renglón. No puede contener entradas nulas. |
| **Llave secundaria** | Un atributo (o combinación de atributos) que se usa estrictamente para fines de recuperación de datos. |
| **Llave foránea** | Un atributo (o combinación de atributos) de una tabla cuyos valores deben corresponderse con la llave primaria en otra tabla o ser nulos. |

A continuación se muestra un ejemplo de cómo podemos clasificar llaves dada la tabla de registros Ventas para la sucursal 5 de una tienda de productos de belleza.

| TABLA_VENTAS-SUC-5 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| ID_Cliente | RFC | Nombre | Apellido | CP | Producto | Cantidad | ID_Vendedor | Telefono |
| 1 | IGMA281299HCTNRK03 | Claudia | Lopéz | 64700 | Secadora prof. De cabello | 1 | 5945 | 9334568715 |
| 2 | IAMD010197SHTNRN02 | Soledad | Villegaz | 86600 | Alaciadora prof. De cabello | 2 | 6319 | 9936482378 |
| 3 | INER071193HAKNTS87 | Deisy | Morales | 86615 | Sillon para lavar cabello | 1 | 5945 | 8129634856 |
| 4 | YBRE190691BTXQPO47 | Ericka | Facundo | 86615 | Masajeador | 4 | 3409 | 7715987265 |
| 5 | IYAP230599YCNARN05 | Claudia | Pérez | 86700 | Exfoliante prof. | 8 | 6319 | 5592367801 |

Figura 4.5: Tabla Ventas Suc-5

Cuadro 4.3: Ejemplos de llaves

| **Tipo de clave** | **Campos correspondientes** |
|---|---|
| **Superllave** | {RFC, ID_Cliente, Apellido}, {RFC, Nombre}, {RFC, Nombre, Teléfono} |
| **Llave candidata** | {RFC}, {ID_Cliente}, {RFC, ID_Cliente} |
| **Llave primaria** | {ID_Cliente}, {RFC} |
| **Llave secundaria** | {Nombre},{Apellido},{Teléfono}, {Nombre, Apellido}, {Nombre, Apellido, Teléfono} |
| **Llave foránea** | {ID_Vendedor} |

Hay que tener en cuenta que cualquier atributo que sea parte de una llave se conoce como atributo llave. Por lo tanto, una superllave es cualquier llave que de manera única identifique a cada renglón. Por ejemplo, en la figura 4.5 podemos realizar una superllave compuesta por {RFC, Nombre, Teléfono} donde {RFC} es una llave primaria y {Nombre, Teléfono} son atributos llaves, que en conjunto conforman una superllave.

Una llave candidata se puede describir como una superllave sin atributos llave innecesarios, es decir, una superllave mínima. De este modo nótese que {RFC, ID, Apellido} es una superllave, pero no una llave candidata porque {RFC, ID_Cliente}, en si, es una llave candidata.

Por otro lado está la llave primaria, que es un atributo (o combinación de atributos) que de manera única identifique a cualquier renglón dado, por lo tanto, cada valor de la llave primaria debe ser único para asegurar que cada renglón sea identificado de manera única por la llave primaria. En ese caso, se dice que la llave exhibe integridad de entidad, es decir, en una llave primaria no se permite un valor nulo. Para el ejemplo anterior, una llave primaria sería {RFC} o {ID_Cliente} o la combinación de ambas {RFC, ID_Cliente}

Un nulo o `NULL` no es ningún valor, puede indicar que no existe valor para colocar en ese atributo. Los nulos nunca pueden ser parte de una llave primaria y también deben evitarse hasta donde sea posible en otros atributos. Por ejemplo, el atributo {Teléfono} no puede ser llave primaria por dos razones: a) se puede dar el caso que dos personas

registren el mismo teléfono y b) pueden haber personas que no tengan número telefónico, creando así valores nulos.

Para el siguiente caso, una llave secundaria se define como aquella que se usa estrictamente para fines de recuperación de datos. Supongamos que viene un cliente a consultar su compra y no recuerda su RFC ni su ID asignado, entonces se puede consultar su compra a través de la llave secundaria, en la cual el cliente da los siguientes datos: Nombre,Apellido, Teléfono y CP. Aunque la llave secundaria no es una manera segura de consultar los datos, suele ser una opción en caso de presentar una situación donde no se recuerden las demás llaves.

Por ultimo, se tiene la llave foránea, la cual es un atributo cuyos valores corresponden con los valores de una llave primaria de una tabla relacionada. Por ejemplo para el caso de la figura 4.5 el atributo {ID_Vendedor} corresponde a valores asignados a otra tabla donde venga información de cada vendedor.

### 4.3.2. Tipos de datos

La llave primaria y llave secundaria añaden estructura a una tabla garantizando que la base de datos sea accesible y estén correctamente relacionadas con otras tablas.

Cada atributo de una tabla tiene un tipo de dato dependiendo qué es lo que se quiera guardar para cada ocurrencia, por lo tanto, es importante conocer con qué tipo de datos estaremos trabajando. Existen una gran variedad de tipos de datos que se pueden presentar, pero, en este trabajo se manejarán los más importantes, debido a su uso y aplicación.

1. **Numéricos:** La principal característica de este tipo de dato es que solo acepta valores numéricos, por lo tanto, se pueden realizar operaciones matemáticas clasificándose en 4 subcategorías.

   *a*) Booleanos: Tiene solo dos valores 0 y 1, que en la mayoría de los casos significa falso y verdadero respectivamente.

   *b*) Entero: Contiene el conjunto de números enteros.

   *c*) Decimal: Los números en este conjunto pueden guardar posiciones decimales hasta cierto límite.

   *d*) Reales: En este conjunto se tienen incluídos el entero y decimal, con la diferencia de que en la parte decimal se guarda mayor cantidad de posiciones.

2. **Carácter:** A diferencia de los tipos de datos numéricos, los datos de caracteres no están restringidos a números. Por lo tanto, se pueden introducir cualquier letra del alfabeto, caracteres especiales y combinaciones de ambos.

3. **Fecha/Hora:** Este tipo de datos es utilizado para representar fechas y horas.

Una vez que se definieron los tipos de datos más importantes para este trabajo, es necesario definir el **dominio** de datos que puede tener cada columna. Por ejemplo, supongamos que una columna registra las edades de niños que estudian la escuela primaria, entonces, para este caso podemos delimitar con un intervalo que permita introducir edades que estén entre 7 y 13 años.

Cuadro 4.4: Tipos de datos

| **Tipo de dato** | **Sintaxis `MySQL`** | **Ejemplo** |
|---|---|---|
| Booleano | `bit` | {TRUE, FALSE} |
| Entero | `int` | {..., -2,-1,0,1,2, ...} |
| Decimal | `decimal` | 4.56 |
| Real | `float` | 23.210389 |
| Longitud variable | `varchar` | 'Andrés Izq' |
| Longitud fija | `char` | '12B-102' |
| Fecha | `date` | '2023-05-07' |
| Fecha y hora | `datetime` | '1998-11-30 08:34:12' |

### 4.3.3. Reglas de integridad

Las reglas de integridad de una base de datos relacional son muy importantes para evitar inconsistencias, además, permiten tener un modelo más sofisticado acercándose más a la realidad.

Estas reglas tienen como objetivo establecer un límite al momento de almacenar los datos, además de que garantizan que la información sea coherente. Las reglas de integridad se clasifican de la siguiente manera:

1. *Integridad de la entidad:* Va dirigida a la llave primaria y con esto no se permite la entrada de un valor nulo.

2. *Integridad de dominio:* Permite que cada valor del campo o columna deba contener un valor del tipo de dato específico, es decir, que corresponda a su dominio y no a otro. Por ejemplo, podemos definir un campo llamado goles, con valores enteros positivos.

3. *Integridad de columna:* Es parecida a la integridad de dominio con la diferencia de que es más especifica, por ejemplo, mientras la integridad de dominio defina entradas con valores enteros, la integridad de columna permite valores entre 10 y 40.

4. *Integridad referencial:* Va dirigida a la llave foránea y se refiere cuando se hacen relaciones con otra entidad. Indica que todo valor de una llave externa debe de existir dentro de una llave primaria de alguna tabla.

5. *Integridad definida por el usuario:* Son reglas semánticas definidas por el diseñador de la base de datos que implementan restricciones más especificas.

Con la implementación de la integridad definida por el usuario se definen las **reglas de negocio** que se definen como descripciones breves, precisas y no ambiguas de una política, procedimiento o restricción dentro de una organización especifica.

Ejemplo: Una institución Bancaria almacena sus contratos en una tabla de la siguiente manera:

| TABLA_ REGISTRO_CONTRATOS | | | | | |
|---|---|---|---|---|---|
| ID_Contrato | ID_Vendedor | Fecha de concertación | Instrumento | Monto | Fecha vencimiento |
| 1 | 2 | 20/05/2023 | Opción Put | $1,000,000.00 | 20/08/2029 |
| 2 | 1 | 20/05/2023 | Forward | $ 500,000.00 | 20/07/2027 |
| 3 | 3 | 20/05/2023 | Forward | $ 120,000.00 | 14/07/2024 |
| 4 | 2 | 20/05/2023 | Opción Call | $ 200,000.00 | 01/01/2025 |
| 5 | 1 | 20/05/2023 | Opción Put | $ 650,000.00 | 07/08/2028 |

Figura 4.6: Registro contratos

Si observamos podemos identificar que se cumplen las reglas de integridad.

- *Integridad de la entidad:* existe una llave primaria que se llama ID_Contrato.
- *Integridad de dominio:* en las columnas Fechas, solo se aceptan datos con formato fecha y en monto solo se aceptan números enteros positivos.
- *Integridad de columna:* en la columna instrumento solo se aceptan las operaciones que tengan que ver con los derivados que opere la institución.
- *Integridad referencial:* la tabla tiene una llave foránea que hace referencia a otra tabla que debe contener los datos del vendedor. Esta referencia debe de estar incluida de manera forzosa en la otra entidad.
- *Integridad definida por el usuario:* viene definido por una política del banco, puede ser que el banco solo permite contratos con montos menores a $1,500,000.00

### 4.3.4. Tipos de relaciones

Para dotar de contenido semántico a una relación hay que especificar de qué modo se relacionan entre si las ocurrencias de las distintas entidades, estableciendo ámbitos, límites y restricciones.

Como primer concepto tenemos a la cardinalidad (relación), la cual indica el número máximo de ocurrencias de una entidad con las que se pueden relacionar una ocurrencia de otra entidad.

- **1:N** Una ocurrencia de una entidad puede relacionarse con varias de otra entidad, pero cada ocurrencia de la segunda entidad solo puede relacionarse con una única ocurrencia de la primera entidad.

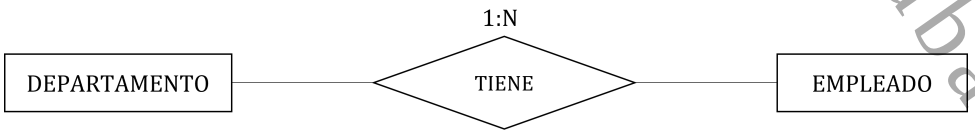



Figura 4.7: Relación 1:N

En la figura 4.7 nos indica la relación que hay entre departamentos y empleados. Por ejemplo, un departamento puede tener más de un empleado, pero un empleado no puede tener más de un departamento asignado.

- **M:N** Cada ocurrencia de una entidad puede relacionarse con varias de otra entidad, y cada ocurrencia de la segunda entidad también puede relacionarse con varias de la primera.

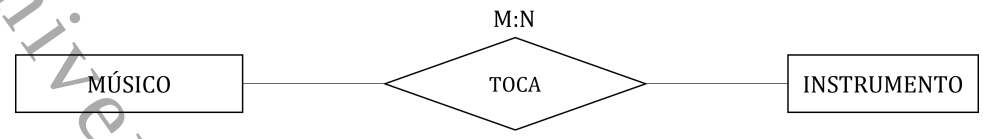



Figura 4.8: Relación M:N

En la figura 4.8 se muestra la relación de músicos e instrumentos que tocan. Por ejemplo, un músico puede tocar más de un instrumento y por el otro lado un instrumento puede ser tocado por más de un músico.

- **1:1** Una ocurrencia de una entidad se relaciona con otra ocurrencia de otra entidad y viceversa.

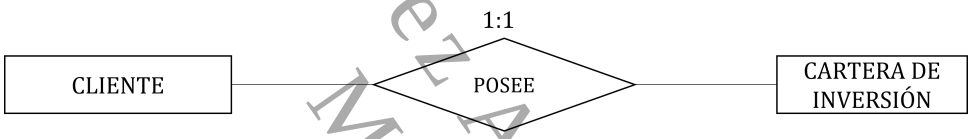



Figura 4.9: Relación 1:1

En la figura 4.9 podemos observar la relación que hay entre cliente y cartera de inversión, siendo ésta uno a uno, es decir, a cada cliente le pertenece solo una cartera de inversión y a cada cartera de inversión le corresponde solo un cliente.

Para aplicar debidamente estas relaciones al momento de desarrollar diseños de bases de datos, hay que tener en cuenta los siguientes puntos:

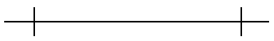
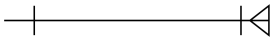
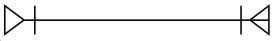
- La relación **1:M** es el ideal del modelo relacional. Por tanto, este tipo de relación debe ser la norma en cualquier diseño de bases de datos relacional.
- La relación **1:1** debe ser rara en cualquier diseño de bases de datos relacional.
- Las relaciones **M:N** no se pueden implementar como tal en el modelo relacional. Pero en estos casos se puede pasar de una relación **M:N** a una **1:M**.

Existe una notación que se llama “pata de gallo” para poder representar las relaciones, la cual se muestra a continuación:

La relación **M:N** no es posible implementarla en un modelo relacional, la razón es porque viola reglas de integridad, a continuación se muestra cómo convertir una relación **M:N** a **1:N** y así poder implementarla en el modelo relacional.

Se tiene una relación **M:N** que tiene que ver con la entidad ALUMNOS y la entidad CLASES. Que viene dado por el siguiente diagrama:

Cuadro 4.5: Simbolo pata de gallo

| Símbolo | Comentario |
|---|---|
|  | Indica una relación **1:1** para el modelo relacional |
|  | Indica una relación **1:M** para el modelo relacional |
|  | Indica una relación **M:N** para el modelo relacional |

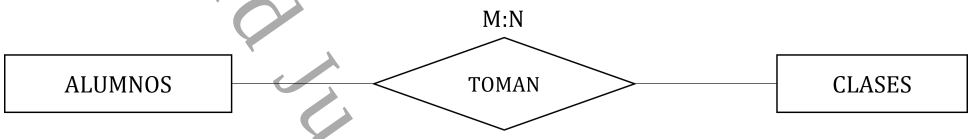



Figura 4.10: Relación M:N

En la figura 4.10 se tiene que un alumno puede tener más de una clase y en una clase puede haber más de un alumno. Si logramos representarlo en el modelo relacional, se tiene el siguiente diagrama:

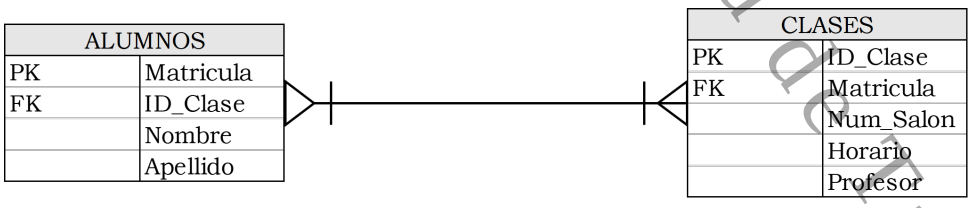



Figura 4.11: Modelo Relacional

Se ha representado un modelo relacional **M:N** que relaciona las entidades ALUMNOS y CLASES y al mismo tiempo podemos observar los campos que componen cada entidad. A continuación se muestran las tablas que componen el modelo:

| TABLA_ALUMNOS | | | |
|---|---|---|---|
| Matrícula | Nombre | Apellido | ID_Clase |
| 12-LFG-007 | Jorge | Sánchez | MAT3 |
| 13-KFD-004 | Alberto | Alejandro | ESP4 |
| 13-KFD-004 | Alberto | Alejandro | MAT3 |
| 17-OPA-078 | David | González | QUIM2 |
| 17-OPA-078 | David | González | MAT3 |
| 17-OPA-078 | David | González | ESP4 |
| 15-AIS-789 | Wilbert | Segura | MAT3 |
| 15-AIS-789 | Wilbert | Segura | QUIM2 |

| TABLA_CLASES | | | | |
|---|---|---|---|---|
| ID_Clase | Matrícula | Num_Salón | Horario | Profesor |
| MAT3 | 12-LFG-007 | 2 | 8 am - 9 am | Andrés |
| MAT3 | 17-OPA-078 | 2 | 8 am - 9 am | Andrés |
| MAT3 | 13-KFD-004 | 2 | 8 am - 9 am | Andrés |
| MAT3 | 15-AIS-789 | 2 | 8 am - 9 am | Andrés |
| ESP4 | 13-KFD-004 | 3 | 10 am - 11 am | Lucia |
| ESP4 | 17-OPA-078 | 3 | 10 am - 11 am | Lucia |
| QUIM2 | 17-OPA-078 | 5 | 12 pm - 1 pm | Maricela |
| QUIM2 | 15-AIS-789 | 5 | 12 pm - 1 pm | Maricela |

Figura 4.12: Relación M:N

En la figura 4.12 podemos observar que las llaves primarias contienen registros duplicados y esto viola la integridad. Por lo tanto, se tiene que pasar de un modelo donde exista una relación **1:M**, la cual se consigue de la siguiente manera:

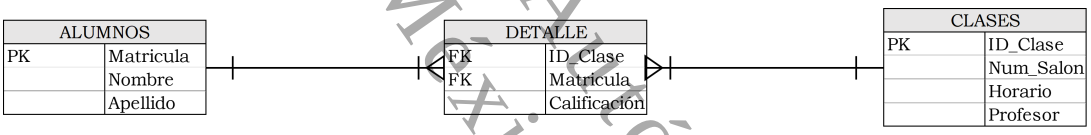



Figura 4.13: Conversión M:N a 1:N

En la figura 4.13 se crea una entidad "Puente" que relaciona las entidades ALUMNOS y CLASES, esta entidad puente se le asigna el nombre DETALLE el cual permite almacenar las repeticiones que hay en Matricula y ID_Clase. También es importante señalar que para estos casos podemos anexar otros campos que ayuden a un mejor entendimiento de las consultas, tal es el caso del atributo Calificación.

| TABLA_ALUMNOS | | |
|---|---|---|
| Matrícula | Nombre | Apellido |
| 12-LFG-007 | Jorge | Sánchez |
| 13-KFD-004 | Alberto | Alejandro |
| 15-AIS-789 | Wilbert | Segura |
| 17-OPA-078 | David | González |

| TABLA_DET | | |
|---|---|---|
| ID_Clase | Matrícula | Calificación |
| MAT3 | 12-LFG-007 | 10 |
| MAT3 | 17-OPA-078 | 9 |
| MAT3 | 13-KFD-004 | 8 |
| MAT3 | 15-AIS-789 | 7 |
| ESP4 | 13-KFD-004 | 10 |
| ESP4 | 17-OPA-078 | 8 |
| QUIM2 | 17-OPA-078 | 9 |
| QUIM2 | 15-AIS-789 | 6 |

| TABLA_CLASES | | | |
|---|---|---|---|
| ID_Clase | Num_Salón | Horario | Profesor |
| MAT3 | 2 | 8 am - 9 am | Andrés |
| ESP4 | 3 | 10 am - 11 am | Lucia |
| QUIM2 | 5 | 12 pm - 1 pm | Maricela |

Figura 4.14: Tablas de M:N a 1:N

En la figura 4.14 se observa que cada entidad tiene una llave primaria única, para el caso de la entidad TABLA_DET se puede asignar un campo que tenga una llave primaria, por ejemplo, Num_Registro que guarda el orden en que se registró cada clase. Pero para

este ejemplo podemos recurrir a una llave candidata, que conforman la combinación de ID_Clase y Matricula, que al mismo tiempo se puede considerar como llave primaria.

En general, la entidad "Puente" tiene las siguientes características:

1. Es dependiente de entidades, es decir, tiene que existir otra entidad para que esta pueda existir. En el ejemplo anterior, la entidad DETALLE no puede existir sin las entidades ALUMNOS y CLASES.
2. No contiene valores `NULL`, por el hecho de formar una llave primaria compuesta por las llaves foráneas.
3. Pueden contener atributos adicionales que no desempeñan ninguna función en el proceso de conexión

## 4.4. Normalización

La tabla constituye el componente fundamental en el diseño de bases de datos, por lo que su estructura resulta especialmente relevante. El objetivo básico del modelado lógico es desarrollar una "buena" descripción de los datos, sus relaciones y sus restricciones.

La normalización es un proceso para evaluar y corregir estructuras de tablas a fin de minimizar redundancias de datos [12], con lo cual se reduce la probabilidad de anomalías de datos. La normalización tiene como objetivo crear un conjunto sólido de relaciones que refleje con precisión las operaciones de la empresa. Al aplicar sus principios, se obtiene un diseño altamente flexible, capaz de adaptarse para incorporar nuevos atributos, conjuntos de entidades y relaciones cuando sea necesario.

La normalización funciona por medio de una serie de etapas llamadas formas normales.

- Primera forma normal **(1NF)**.
- Segunda forma normal **(2NF)**.
- Tercera forma normal **(3NF)**.
- Forma normal de Boyce-Codd **(BCNF)**.
- Cuarta forma normal **(4NF)**.

Cada forma normal está anidada respecto a la anterior. Es decir, una relación en la segunda forma normal está también en primera forma normal, y una relación en **4NF** está así mismo en BCNF, 3NF, 2NF, 1NF.

Existe una quinta forma normal (5NF) que tiene que ver con relaciones que pueden subdividirse en subrelaciones, pero no se pueden reconstruir. La condición bajo la cual se origina esta situación está fuera del alcance de esta tesis, pero puede realizarse su consulta en el trabajo de Date [16].

Aunque la normalización es un aspecto clave en el diseño de bases de datos, no siempre se debe asumir que alcanzar el nivel más alto de normalización es la opción más adecuada.

Para analizar las diferentes formas normales, es necesario empezar por una tabla que contenga errores de integridad. Tal como se muestra en la figura 4.15, donde se puede notar lo siguiente:

1. El número de proyecto contiene valores nulos.

2. La tabla invita a tener inconsistencia de datos, por ejemplo, en la clase de trabajo se pude introducir Act. como Actuario.

3. Presenta anomalías de actualización, inserción y eliminación.

| FORMATO REPORTE | | | | | | |
|---|---|---|---|---|---|---|
| Num_Proy | Nombre_Proy | Num_Empleado | Nombre_Empleado | Clase_Trabajo | Pago_Hora | Horas |
| 7 | Diseño informático | 104 | Alberto Hernández López | Programador | 100 | 27.5 |
| | | 102 | Daniel Pérez Alejandro | Programador | 100 | 21 |
| | | 106 | Consuelo Diaz Olgado | Analista de Sistema | 90 | 25 |
| | | 107 | Roberto León Cortez | Diseñador de DB | 120 | 26 |
| | | 103 | Abigail Montes Ruiz | Diseñador de DB | 120 | 26 |
| 13 | Riesgo y control | 115 | Cristian Morales Núñez | Matemático | 150 | 31 |
| | | 119 | Denis Madrigal Alves | Actuario | 170 | 35.5 |
| | | 105 | Luis Lázaro Alejandro | Matemático | 150 | 27 |
| | | 113 | Andrés Izquierdo Morales | Actuario | 170 | 35.5 |
| 19 | Marketing | 106 | Consuelo Diaz Olgado | Analista de Sistema | 90 | 25 |
| | | 105 | Luis Lázaro Alejandro | Ventas | 80 | 39 |
| | | 114 | Alex Jiménez Leal | Marketing | 85 | 18 |
| | | 112 | Jaime Ruiz Morales | Marketing | 85 | 18 |
| | | 107 | Roberto León Cortez | Diseñador de DB | 120 | 26 |
| 29 | Control y Calidad | 108 | Brandon García Leal | Matemático | 150 | 35 |
| | | 116 | Oscar Osorio Cerino | Ingeniero Prod. | 80 | 28 |
| | | 102 | Daniel Pérez Alejandro | Programador | 100 | 29 |
| | | 111 | Josué Santiago Mendoza | Ingeniero Prod. | 80 | 28 |
| | | 109 | Carlos González Alejandro | Ingeniero Prod. | 80 | 27 |
| | | 119 | Denis Madrigal Alvez | Actuario | 170 | 35 |
| | | 113 | Andrés Izquierdo Morales | Actuario | 170 | 40 |

Figura 4.15: Tabla Reporte

### 4.4.1. Conceptos preliminares

En el contexto de la normalización, cualquier atributo que forme parte, al menos parcialmente, de una llave, se conoce como **atributo primo** en lugar del término más comúnmente utilizado de atributo llave. Por el contrario, un **atributo no primo**, no es parte de una llave candidata.

Dos conceptos de especial interés en la normalización son las dependencias parciales y las transitivas. Una **dependencia parcial** existe cuando hay dependencias funcionales en la que el determinante es parte de la llave primaria. Por ejemplo, si $(A, B) \rightarrow (C, D), B \rightarrow C$ y $(A, B)$ es la llave primaria, entonces la dependencia funcional $B \rightarrow C$ es una dependencia parcial porque solo parte de la llave primaria $(B)$ se necesita para determinar el valor de $C$.

Existe **dependencia transitiva** cuando hay dependencias funcionales tales que $X \rightarrow Y, Y \rightarrow Z$ y $X$ es la llave primaria. En ese caso, la dependencia $X \rightarrow Z$ es una dependencia transitiva porque $X$ determina el valor de $Z$ por medio de $Y$. Se presentará una dependencia transitiva solo cuando existe dependencia funcional entre atributos no primos. En el ejemplo previo, la dependencia transitiva real es $X \rightarrow Z$. No obstante, la dependencia $Y \rightarrow Z$ es señal de que existe dependencia transitiva. En consecuencia, en toda la exposición del proceso de normalización, la existencia de una dependencia funcional entre atributos no primos será considerada como signo de dependencia transitiva.

Otro concepto es un **grupo repetidor**, el cual como su nombre indica es un grupo de múltiples entradas del mismo tipo que existen en cualquier instancia de atributo.

### 4.4.2. Primera forma normal *(1NF)*

El objetivo de esta primera forma es identificar todas las dependencias, dar un formato adecuado de tabla, sin grupos repetidos y llave primaria identificada.

**Paso 1: eliminar los grupos repetidores.**

Se representan los datos en un formato tabular, en el cual cada celda contiene un único valor y no se permiten grupos repetidos.

| FORMATO REPORTE | | | | | | |
|---|---|---|---|---|---|---|
| Num_Proy | Nombre_Proy | Num_Empleado | Nombre_Empleado | Clase_Trabajo | Pago_Hora | Horas |
| 7 | Diseño informático | 104 | Alberto Hernández López | Programador | 100 | 27.5 |
| 7 | Diseño informático | 102 | Daniel Pérez Alejandro | Programador | 100 | 21 |
| 7 | Diseño informático | 106 | Consuelo Diaz Olgado | Analista de Sistema | 90 | 25 |
| 7 | Diseño informático | 107 | Roberto León Cortez | Diseñador de DB | 120 | 26 |
| 7 | Diseño informático | 103 | Abigail Montes Ruiz | Diseñador de DB | 120 | 26 |
| 13 | Riesgo y control | 115 | Cristian Morales Núñez | Matemático | 150 | 31 |
| 13 | Riesgo y control | 119 | Denis Madrigal Alves | Actuario | 170 | 35.5 |
| 13 | Riesgo y control | 105 | Luis Lázaro Alejandro | Matemático | 150 | 27 |
| 13 | Riesgo y control | 113 | Andrés Izquierdo Morales | Actuario | 170 | 35.5 |
| 19 | Marketing | 106 | Consuelo Diaz Olgado | Analista de Sistema | 90 | 25 |
| 19 | Marketing | 105 | Luis Lázaro Alejandro | Ventas | 80 | 39 |
| 19 | Marketing | 114 | Alex Jiménez Leal | Marketing | 85 | 18 |
| 19 | Marketing | 112 | Jaime Ruiz Morales | Marketing | 85 | 18 |
| 19 | Marketing | 107 | Roberto León Cortez | Diseñador de DB | 120 | 26 |
| 29 | Control y Calidad | 108 | Brandon García Leal | Matemático | 150 | 35 |
| 29 | Control y Calidad | 116 | Oscar Osorio Cerino | Ingeniero Prod. | 80 | 28 |
| 29 | Control y Calidad | 102 | Daniel Pérez Alejandro | Programador | 100 | 29 |
| 29 | Control y Calidad | 111 | Josué Santiago Mendoza | Ingeniero Prod. | 80 | 28 |
| 29 | Control y Calidad | 109 | Carlos González Alejandro | Ingeniero Prod. | 80 | 27 |
| 29 | Control y Calidad | 119 | Denis Madrigal Alvez | Actuario | 170 | 35 |
| 29 | Control y Calidad | 113 | Andrés Izquierdo Morales | Actuario | 170 | 40 |

Figura 4.16: Tabla en *1NF*

**Paso 2: identificar la llave primaria**

En la figura 4.16 se representa más de un simple cambio de relleno. Podemos identificar que Num_Proy no es una llave primaria, por otro lado, podemos identificar una llave compuesta por la combinación de Num_Proy y Num_Empleado la cual si identifica de manera única el valor de cualquier atributo. Por lo tanto Num_Proy y Num_Empleado es nuestra llave primaria.

$$\{Num\_Proy, Num\_Empleado\} \quad \rightarrow \quad \text{Llave primaria}$$

**Paso 3: identificar todas las dependencias**

Con la identificación de la llave primaria en el paso 2, se tiene identificada la siguiente dependencia.

$\{Num\_Proy, Num\_Empleado\} \quad \rightarrow \quad \{Nombre\_Proy\}, \{Nombre\_Empleado\}, \{Clase\_Trabajo\}, \{Pago\_Hora\}, \{Horas\}$

Esto es posible demostrar, con la justificación de que Num_Proy y Num_Empleado forman la llave primaria, misma que es una llave candidata.

Podemos identificar otras dependencias las cuales se tiene en la siguiente lista:

- $\{Num\_Proy\} \quad \rightarrow \quad \{Nombre\_Proy\}$ Es una dependencia parcial.
- $\{Num\_Empleado\} \quad \rightarrow \quad \{Nombre\_Empleado\}, \{Clase\_Trabajo\}, \{Pago\_Horas\}$ Son dependencias parciales.
- $\{Clase\_Trabajo\} \quad \rightarrow \quad \{Pago\_Hora\}$ Es una dependencia transitiva.

Podemos representar estas dependencias en un diagrama tal como se muestra en la figura siguiente:

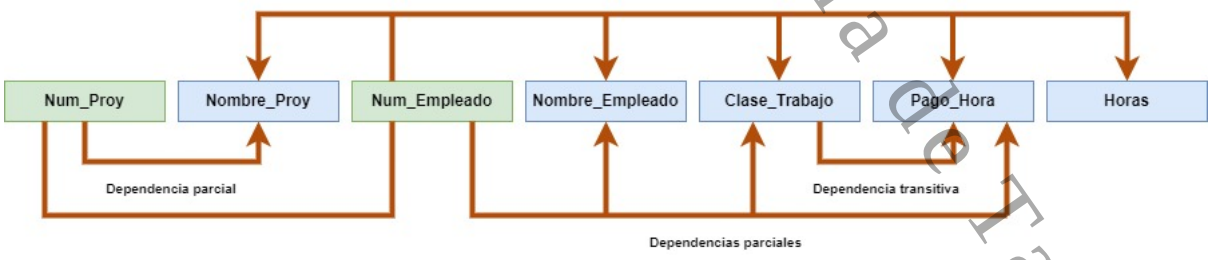



Figura 4.17: Diagrama de dependencia de la forma normal *1NF*

El término **primera forma normal** describe el formato tabular en el que:

- Todos los atributos llave están definidos.
- No hay grupos repetidores en la tabla. En otras palabras, cada intersección de renglón/columna contiene un y solo un valor, no un conjunto de ellos.
- Todos los atributos son dependientes de la llave primaria.

### 4.4.3. Segunda forma normal *(2NF)*

La conversión a 2NF se hace solo cuando la 1NF tiene una llave primaria compuesta. Si la 1NF tiene una llave primaria de un solo atributo, entonces la tabla está automáticamente en la 2NF. Para el ejemplo anterior se tiene llave primaria compuesta, por lo tanto se tiene que realizar la 2NF.

**Paso 1: crear nuevas tablas para eliminar dependencias parciales**

Para cada componente de la clave primaria que funcione como determinante en una dependencia parcial, se crea una nueva tabla que utiliza una copia de ese componente como clave primaria. Sin embargo, es fundamental que dichos componentes se mantengan tanto en las nuevas tablas como en la tabla original. Es primordial que los determinantes continúen en la tabla original porque serán llaves foráneas para las relaciones que se requieran para vincular estas nuevas tablas a la original.

Observe que en la imagen 4.18 se crearon 2 tablas nuevas (PROYECTOS, EMPLEADOS) con las dependencias parciales que tenían y se dejó una que contuviera ambas llaves (ASIGNACIONES).

**Paso 2: reasignar atributos dependientes correspondientes**

Los atributos que son dependientes en una dependencia parcial se remueven de la tabla original y se colocan en la nueva tabla con su determinante. Cualquier otro atributo que no sea dependiente en una dependencia parcial permanecerán en la tabla original.

EMPLEADOS

| Num_Empleado | Nombre_Empleado | Clase_Trabajo | Pago_Hora |
|---|---|---|---|
| 104 | Alberto Hernández López | Programador | 100 |
| 102 | Daniel Pérez Alejandro | Programador | 100 |
| 106 | Consuelo Diaz Olgado | Analista de Sistema | 90 |
| 107 | Roberto León Cortez | Diseñador de DB | 120 |
| 103 | Abigail Montes Ruiz | Diseñador de DB | 120 |
| 115 | Cristian Morales Núñez | Matemático | 150 |
| 119 | Denis Madrigal Alves | Actuario | 170 |
| 105 | Luis Lázaro Alejandro | Matemático | 150 |
| 113 | Andrés Izquierdo Morales | Actuario | 170 |
| 114 | Alex Jiménez Leal | Marketing | 85 |
| 112 | Jaime Ruiz Morales | Marketing | 85 |
| 108 | Brandon García Leal | Matemático | 150 |
| 116 | Oscar Osorio Cerino | Ingeniero Prod. | 80 |
| 111 | Josué Santiago Mendoza | Ingeniero Prod. | 80 |
| 109 | Carlos González Alejandro | Ingeniero Prod. | 80 |

PROYECTOS

| Num_Proy | Nombre_Proy |
|---|---|
| 7 | Diseño informatico |
| 13 | Riesgo y control |
| 19 | Marketing |
| 29 | Control y Calidad |

ASIGNACIONES

| Num_Proy | Num_Empleado | Horas |
|---|---|---|
| 7 | 104 | 27.5 |
| 7 | 102 | 21 |
| 7 | 106 | 25 |
| 7 | 107 | 26 |
| 7 | 103 | 26 |
| 13 | 115 | 31 |
| 13 | 119 | 35.5 |
| 13 | 105 | 27 |
| 13 | 113 | 35.5 |
| 19 | 106 | 25 |
| 19 | 105 | 39 |
| 19 | 114 | 18 |
| 19 | 112 | 18 |
| 19 | 107 | 26 |
| 29 | 108 | 35 |
| 29 | 116 | 28 |
| 29 | 102 | 29 |
| 29 | 111 | 28 |
| 29 | 109 | 27 |
| 29 | 119 | 35 |
| 29 | 113 | 40 |

Figura 4.18: Tabla en *2NF*

En la figura 4.18 se visualizan de manera puntual los cambios realizados en las tablas para que se muestren en 2NF, mientras en la figura 4.19 se muestra el diagrama de dependencia de dichas tablas.

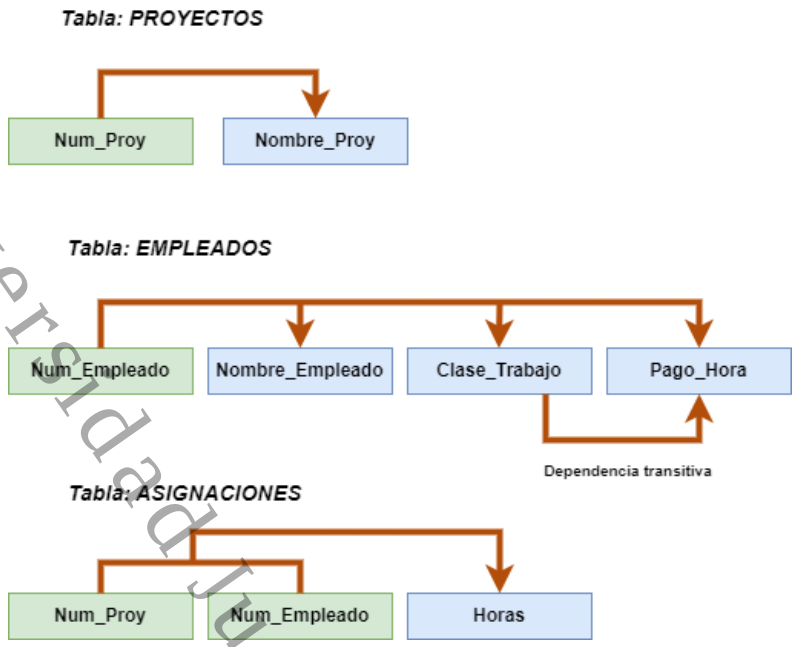



Figura 4.19: Diagrama de dependencia en *2NF*

Una tabla está en **segunda forma normal** cuando:

- Está en 1NF
- No incluye dependencias parciales: esto es, ningún atributo es dependiente de una parte de la llave primaria.

### 4.4.4. Tercera forma normal *(3NF)*

La conversión a 3NF se realiza solo cuando la 2NF contiene dependencias transitivas.

**Paso 1: hacer nuevas tablas para eliminar dependencias transitivas**

Por cada dependencia transitiva, se debe escribir una copia de su determinante como llave primaria para una nueva tabla. Un **determinante** es cualquier atributo cuyo valor determina otros valores dentro de un renglón. Si tenemos tres dependencias transitivas diferentes, tendremos tres determinantes. Al igual que en la conversión a 2NF, es importante que el determinante permanezca en la tabla original para servir como llave foránea. Para nuestro ejemplo, tenemos un determinante en la dependencia transitiva, el cual es:

- $\{Clase\_Trabajo\} \rightarrow \{Pago\_Hora\}$

**Paso 2: reasignar atributos dependientes correspondientes**

Con la identificación de los atributos dependientes que se vio en el paso 1, se tiene que colocar nuevas tablas con sus determinantes. En el ejemplo, se eliminó Pago_Hora de la tabla EMPLEADOS para dejar la definición de dependencia de la tabla EMPLEADOS como:

- $\{Num\_Empleado\} \rightarrow \{Nombre\_Empleado\},\{Clase\_Trabajo\}$

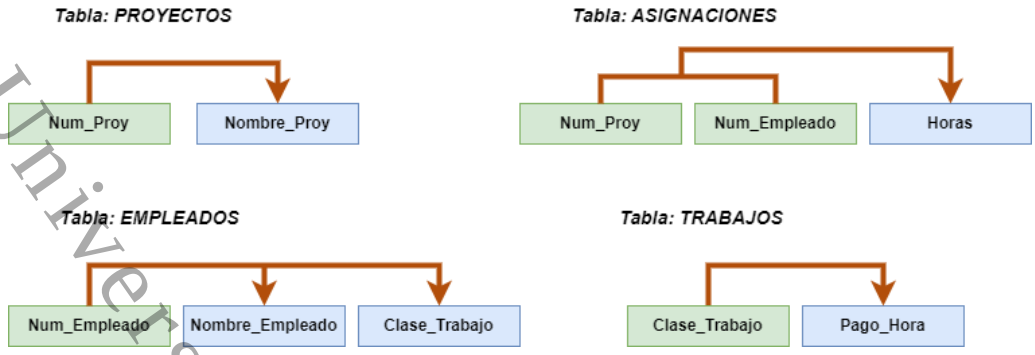



Figura 4.20: Diagrama de dependencia en *3NF*

TRABAJO

| Clase_Trabajo | Pago_Hora |
|---|---|
| Programador | 100 |
| Analista de Sistema | 90 |
| Diseñador de DB | 120 |
| Matemático | 150 |
| Actuario | 170 |
| Marketing | 85 |
| Ingeniero Prod. | 80 |

PROYECTOS

| Num_Proy | Nombre_Proy |
|---|---|
| 7 | Diseño informatico |
| 13 | Riesgo y control |
| 19 | Marketing |
| 29 | Control y Calidad |

EMPLEADOS

| Num_Empleado | Nombre_Empleado | Clase_Trabajo | Pago_Hora |
|---|---|---|---|
| 104 | Alberto Hernández López | Programador | 100 |
| 102 | Daniel Pérez Alejandro | Programador | 100 |
| 106 | Consuelo Diaz Olgado | Analista de Sistema | 90 |
| 107 | Roberto León Cortez | Diseñador de DB | 120 |
| 103 | Abigail Montes Ruiz | Diseñador de DB | 120 |
| 115 | Cristian Morales Núñez | Matemático | 150 |
| 119 | Denis Madrigal Alves | Actuario | 170 |
| 105 | Luis Lázaro Alejandro | Matemático | 150 |
| 113 | Andrés Izquierdo Morales | Actuario | 170 |
| 114 | Alex Jiménez Leal | Marketing | 85 |
| 112 | Jaime Ruiz Morales | Marketing | 85 |
| 108 | Brandon García Leal | Matemático | 150 |
| 116 | Oscar Osorio Cerino | Ingeniero Prod. | 80 |
| 111 | Josué Santiago Mendoza | Ingeniero Prod. | 80 |
| 109 | Carlos González Alejandro | Ingeniero Prod. | 80 |

ASIGNACIONES

| Num_Proy | Num_Empleado | Horas |
|---|---|---|
| 7 | 104 | 27.5 |
| 7 | 102 | 21 |
| 7 | 106 | 25 |
| 7 | 107 | 26 |
| 7 | 103 | 26 |
| 13 | 115 | 31 |
| 13 | 119 | 35.5 |
| 13 | 105 | 27 |
| 13 | 113 | 35.5 |
| 19 | 106 | 25 |
| 19 | 105 | 39 |
| 19 | 114 | 18 |
| 19 | 112 | 18 |
| 19 | 107 | 26 |
| 29 | 108 | 35 |
| 29 | 116 | 28 |
| 29 | 102 | 29 |
| 29 | 111 | 28 |
| 29 | 109 | 27 |
| 29 | 119 | 35 |
| 29 | 113 | 40 |

Figura 4.21: Tablas en *3NF*

En la imagen 4.20 se muestran 4 modelos de las cuales podemos identificar que no hay dependencias transitivas y cada llave primaria determina los demás atributos que la componen sin tener dependencias parciales.

El resultado de la figura 4.20 se muestra en la figura 4.21 donde podemos observar que hay 4 tablas.

Una tabla está en **tercera forma normal** cuando:

- Está en 2NF; *y además*
- no contiene dependencias transitivas.

### 4.4.5. Recomendaciones para el mejoramiento del diseño

Las estructuras de una tabla se limpian para eliminar dependencias parciales y transitivas problemáticas. Ahora podemos concentrarnos en mejorar la capacidad de la base de datos para dar información y en mejorar sus características operacionales. A continuación se dan algunos recomendaciones para el mejoramiento del diseño.

**Evaluar asignaciones de una llave primaria**

Podemos asignar un nuevo campo en la tabla TRABAJO que guarde el Id_Trabajo. Con esto se evitaría ingresar nombres de trabajos incompletos y causar violaciones de integridad referencial e inconsistencia de datos.

| TRABAJO | | |
|---|---|---|
| Id_Trabajo | Clase_Trabajo | Pago_Hora |
| 1 | Programador | 100 |
| 2 | Analista de Sistema | 90 |
| 3 | Diseñador de DB | 120 |
| 4 | Matemático | 150 |
| 5 | Actuario | 170 |
| 6 | Marketing | 85 |
| 7 | Ingeniero Prod. | 80 |

Figura 4.22: Asignación de llave sustituta

En la figura 4.22 se observa la introducción de un campo que contiene una llave sustituta, la cual es una llave primaria artificial introducida por el diseñador con el fin de simplificar la asignación de llaves primarias a la tablas.

**Evaluar convenciones para dar un nombre**

Un elemento importante que se usa al momento de trabajar con colaboradores en un diseño de bases de datos es colocar un nombre adecuado a cada campo y puntualizar que tipo de datos serán introducidos y a qué tabla permanecen. Por ejemplo, en nuestro diseño podemos hacer los siguientes cambios de nombres:

- Pago_Hora se cambia por Trab_P_Hora con la finalidad de asociarlo con la tabla TRABAJO.
- Clase_Trabajo se ajusta mejor a Descrip_Trabajo.
- Horas se cambia a Horas_Asignadas

**Refinar atomicidad de atributo**

Un atributo atómico es el que no puede subdividirse en más categorías, para formar nuevas columnas; se dice que tal atributo muestra atomicidad. Un ejemplo de atributo no atómico es Nombre_Empleado por el hecho de que se puede dividir en Nombre, Primer Apellido y Segundo Apellido. Al mejorar el grado de atomicidad también se obtiene la flexibilidad de consulta. Por lo tanto, se generan los siguientes campos a la tabla Empleados.

- Nombre
- Primer_Apellido
- Segundo_Apellido

**Identificar nuevos atributos**

Si la tabla EMPLEADOS se usara en un ambiente real, es posible que se tengan que agregar nuevos atributos. Por ejemplo, pagos del salario bruto, pagos del seguro social y servicio médico. Para nuestro ejemplo solo se agregará la fecha de contratación del empleado (Fecha_Contrat) la cual se puede usar para calcular el tiempo de servicio en la empresa.

**Identificar nuevas relaciones**

De acuerdo con el informe original, los usuarios necesitan rastrear cuál empleado está actuando como gerente de cada proyecto. Esto puede implementarse como una relación entre EMPLEADOS y PROYECTOS. Una regla de negocio muy importante es el hecho

de que cada proyecto tiene un solo gerente. Por lo tanto, la capacidad del sistema para dar información detallada acerca del gerente de cada proyecto está asegurada con el uso de Num_Empleado como llave foránea en PROYECTOS.

### Pulir llaves primarias según sea necesario para granularidad de datos

La granularidad se refiere al nivel de detalle representado por los valores guardados en el renglón de una tabla. Se dice que los datos guardados en su nivel de granularidad más bajo son datos atómicos.

En el mundo ideal (de diseño de bases de datos), el nivel de granularidad deseado se determina en el diseño conceptual o en la fase de recolección de requisitos. En un ambiente real, cambiar requisitos de granuladidad podría dictar modificaciones en la selección de una llave primaria, las cuales podrían en última instancia requerir el uso de llaves sustitutas.

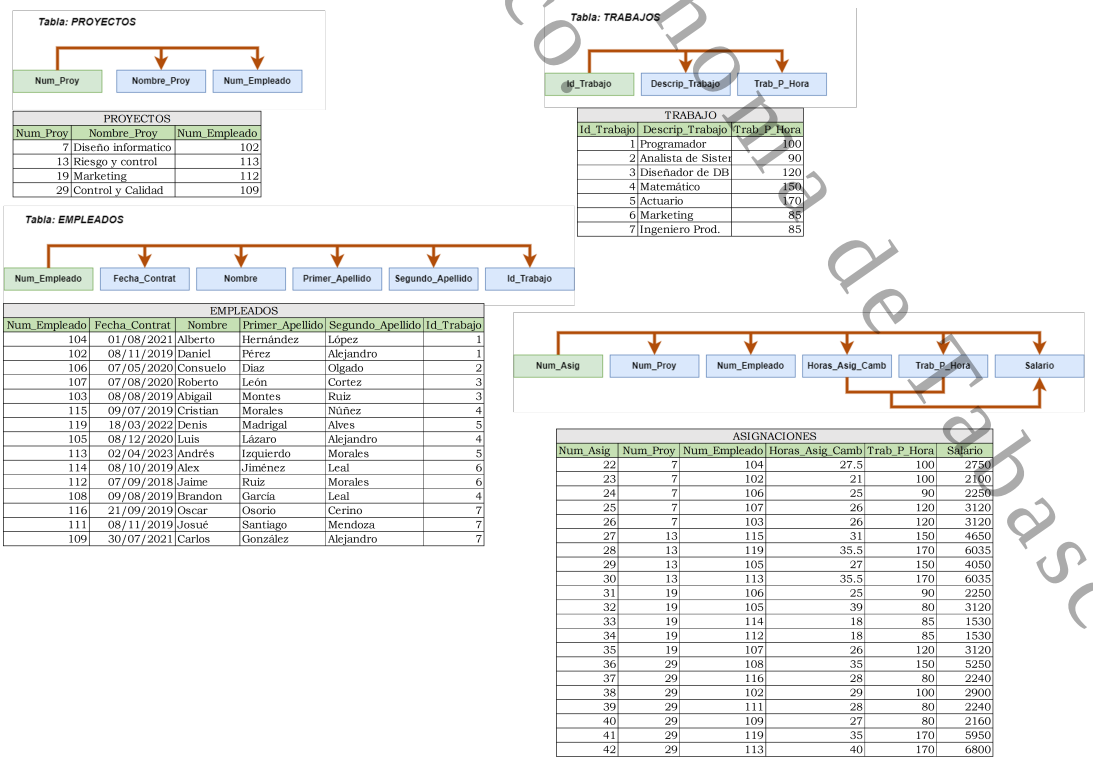
### Mantener precisión histórica

Es muy importante para una institución guardar los datos a lo largo del tiempo. Para nuestro ejemplo, en tabla ASIGNACIONES podemos guardar cada cierto tiempo de pago los esquemas, esto quiere decir que los registros de Horas_Asignadas guardarán en el nuevo campo Horas_Asig_Cam con diferentes valores a lo largo del tiempo.

### Evaluar el uso de atributos derivados

Un atributo derivado es un nuevo campo que se deriva de otro u otros campos, con el fin de mostrar mayor información a la hora de realizar consultas. En nuestro ejemplo se tiene el siguiente atributo derivado para la tabla ASIGNACIONES.

- $\{Horas\_Asig\_Camb\} + \{Trab\_P\_Hora\} \rightarrow \{Salario\}$




PROYECTOS

| Num_Proy | Nombre_Proy | Num_Empleado |
|---|---|---|
| 7 | Diseño informatico | 102 |
| 13 | Riesgo y control | 113 |
| 19 | Marketing | 112 |
| 29 | Control y Calidad | 109 |

TRABAJO

| Id_Trabajo | Descrip_Trabajo | Trab_P_Hora |
|---|---|---|
| 1 | Programador | 100 |
| 2 | Analista de Siste | 90 |
| 3 | Diseñador de DB | 120 |
| 4 | Matemático | 150 |
| 5 | Actuario | 170 |
| 6 | Marketing | 85 |
| 7 | Ingeniero Prod. | 85 |

EMPLEADOS

| Num_Empleado | Fecha_Contrat | Nombre | Primer_Apellido | Segundo_Apellido | Id_Trabajo |
|---|---|---|---|---|---|
| 104 | 01/08/2021 | Alberto | Hernández | López | 1 |
| 102 | 08/11/2019 | Daniel | Pérez | Alejandro | 1 |
| 106 | 07/05/2020 | Consuelo | Diaz | Olgado | 2 |
| 107 | 07/08/2020 | Roberto | León | Cortez | 3 |
| 103 | 08/08/2019 | Abigail | Montes | Ruiz | 3 |
| 115 | 09/07/2019 | Cristian | Morales | Núñez | 4 |
| 119 | 18/03/2022 | Denis | Madrigal | Alves | 5 |
| 105 | 08/12/2020 | Luis | Lázaro | Alejandro | 4 |
| 113 | 02/04/2023 | Andrés | Izquierdo | Morales | 5 |
| 114 | 08/10/2019 | Alex | Jiménez | Leal | 6 |
| 112 | 07/09/2018 | Jaime | Ruiz | Morales | 6 |
| 108 | 09/08/2019 | Brandon | García | Leal | 4 |
| 116 | 21/09/2019 | Oscar | Osorio | Cerino | 7 |
| 111 | 08/11/2019 | Josué | Santiago | Mendoza | 7 |
| 109 | 30/07/2021 | Carlos | González | Alejandro | 7 |

ASIGNACIONES

| Num_Asig | Num_Proy | Num_Empleado | Horas_Asig_Camb | Trab_P_Hora | Salario |
|---|---|---|---|---|---|
| 22 | 7 | 104 | 27.5 | 100 | 2750 |
| 23 | 7 | 102 | 21 | 100 | 2100 |
| 24 | 7 | 106 | 25 | 90 | 2250 |
| 25 | 7 | 107 | 26 | 120 | 3120 |
| 26 | 7 | 103 | 26 | 120 | 3120 |
| 27 | 13 | 115 | 31 | 150 | 4650 |
| 28 | 13 | 119 | 35.5 | 170 | 6035 |
| 29 | 13 | 105 | 27 | 150 | 4050 |
| 30 | 13 | 113 | 35.5 | 170 | 6035 |
| 31 | 19 | 106 | 25 | 90 | 2250 |
| 32 | 19 | 105 | 39 | 80 | 3120 |
| 33 | 19 | 114 | 18 | 85 | 1530 |
| 34 | 19 | 112 | 18 | 85 | 1530 |
| 35 | 19 | 107 | 26 | 120 | 3120 |
| 36 | 29 | 108 | 35 | 150 | 5250 |
| 37 | 29 | 116 | 28 | 80 | 2240 |
| 38 | 29 | 102 | 29 | 100 | 2900 |
| 39 | 29 | 111 | 28 | 80 | 2240 |
| 40 | 29 | 109 | 27 | 80 | 2160 |
| 41 | 29 | 119 | 35 | 170 | 5950 |
| 42 | 29 | 113 | 40 | 170 | 6800 |

Figura 4.23: Base de datos completa

### 4.4.6. La forma normal de Boyse-Codd *(BCNF)*

Una tabla está en la forma de Boyse-Codd cuando todo determinante de ella sea una llave candidata. De manera que cuando una tabla contiene solo una llave candidata, la 3NF y la BCNF son equivalentes.

Para comenzar idealizando cuándo se tiene que normalizar a la forma Boyse-Codd, es necesario recordar que existe una dependencia transitiva cuando un atributo no primo es dependiente de otro atributo no primo. Pero ¿qué pasa en caso en el que un atributo no primo es el determinante de un atributo llave?. Esta condición no viola la 3NF, pero tampoco satisface los requisitos de la BCNF porque ésta requiere que todo determinante de la tabla sea una llave candidata. Esta situación se muestra en la figura 4.24.

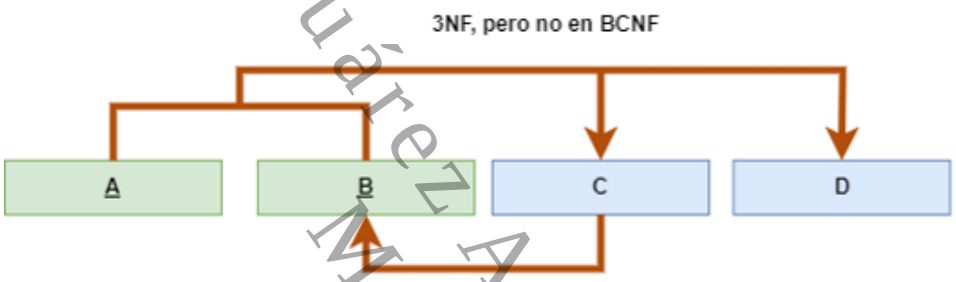



Figura 4.24: Una tabla que está en 3NF pero no en BCNF

En la imagen anterior podemos observar las siguientes dependencias funcionales.

- $A+B \rightarrow C,D$
- $A+C \rightarrow B,D$
- $C \rightarrow B$

En el modelo presentado se observa que hay dos llaves candidatas: *(A+B)* y *(A+C)*, además, dicha estructura no tiene dependencias parciales ni transitivas. Así, la estructura del modelo satisface los requisitos de 3NF. No obstante, la condición $C \rightarrow B$ hace que la tabla no satisfaga los requisitos de la BCNF.

Los pasos que se tienen que realizar, para que el modelo esté en la forma normal Boyse-Codd, se muestran a continuación.

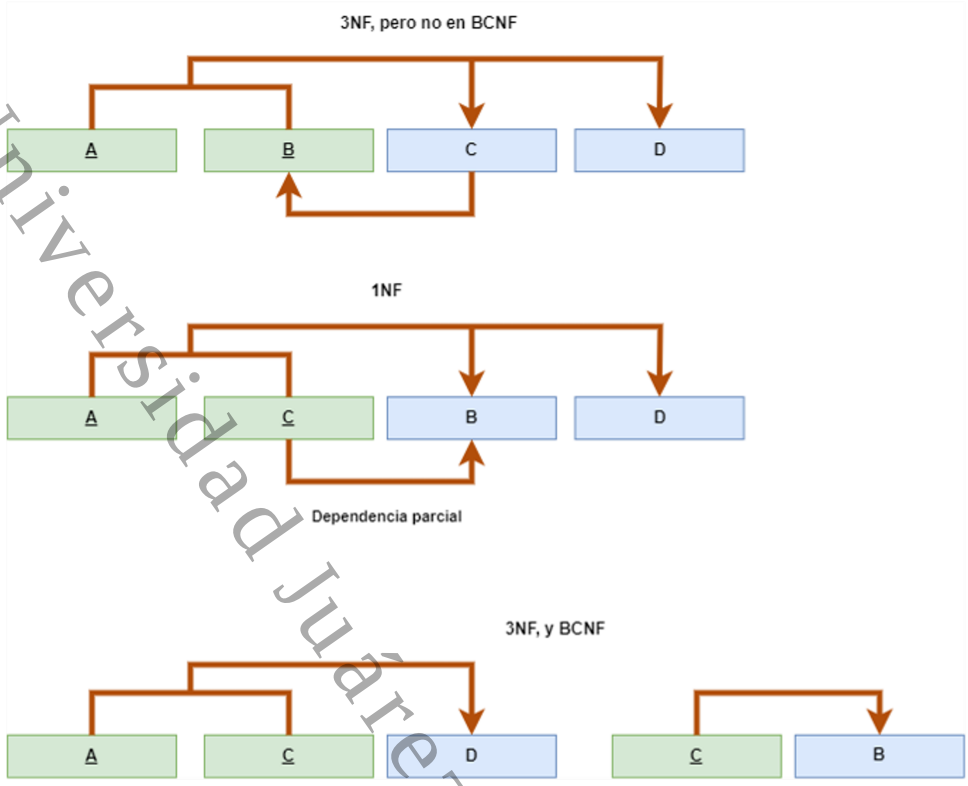



Figura 4.25: De 3NF a BCNF

En la figura 4.25 se muestra un cambio de llave primaria, es decir, se pasa de *(A,B)* a *(A,C)*, lo cual provoca una dependencia parcial volviéndose 1NF, esto provoca realizar el proceso de las siguientes formas normales hasta presentarse un modelo que no tenga dependencia parcial y esté en BCNF.

Una tabla está en la **forma normal de Boyce-Codd** cuando todo determinante de la tabla es una llave candidata.

### 4.4.7. Cuarta forma normal *(4NF)*

Aunque la forma normal Boyce-Codd es suficiente para remover cualquier anomalía debida a dependencias funcionales, aún no se está exento de otro tipo de problemas de dependencias que puedan causar problemas de diseño. Éstas son dependencias multivaluadas.

Para comprender una dependencia multivaluada observe la imagen 4.26 del lado izquierdo que contiene datos de una relación entre estudiantes, materias y deportes. Suponga que los estudiantes pueden inscribirse en varias materias y participar en diferentes deportes. En este caso, la única llave primaria es la combinación de todos los atributos.

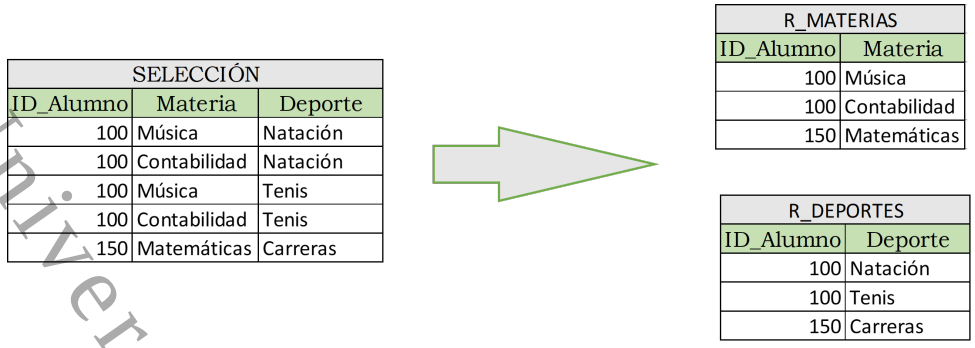



Figura 4.26: Dependencia multivaluada.

¿Qué relación hay entre ID_ Alumno y Materias? No es una dependencia funcional porque los alumnos pueden tener varias materias, es decir, un solo valor de ID_ Alumno puede tener asignadas varias materias así como varios deportes. Las dependencias multivaluadas conducen anomalías de modificación por el simple hecho de repetirse muchas veces el ID_ Alumno.

Para evitar estas anomalías se tiene que eliminar las dependencias multivaluadas. Esto se hace dividiendo la tabla en dos, tal como se muestra en la figura 4.26 del lado derecho. También podemos observar en la figura 4.27 un panorama completo de la base de datos.

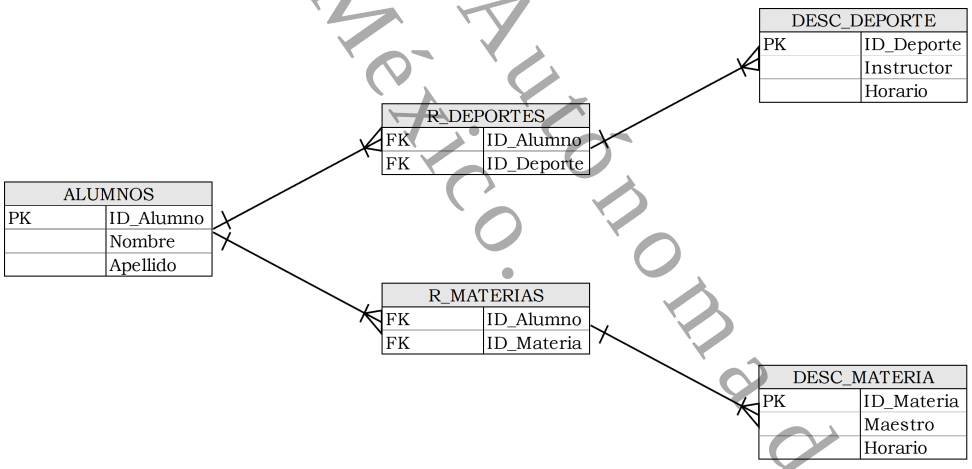



Figura 4.27: Diagrama completo en 4NF.

Una tabla está en **cuarta forma normal** si está en la forma BCNF y no tiene dependencias multivaluadas.

## 4.5. Reglas de Codd

En 1985 Edgar Codd publicó una lista de 12 reglas para definir un sistema de base de datos relacional. La razón por la que Codd publicó la lista fue su preocupación por muchas empresas que estaban vendiendo productos como “relacionales” cuando no satisfacían los mínimos estándares relacionales. A continuación se muestra un marco de referencia de lo que se menciona en cada regla.

1. **Información:** Toda información de una base de datos relacional debe estar representada lógicamente como valores en columnas y en renglones dentro de tablas.
2. **Acceso garantizado:** Todo valor en una tabla está garantizado para ser accesible mediante una combinación de nombre de tabla, valor de llave primaria y nombre de columna.
3. **Tratamiento sistemático de nulos:** Los datos nulos debe ser representados y tratados de forma sistemática, independiente del tipo de datos.
4. **Catálogo dinámico en línea basado en el modelo relacional:** Los metadatos deben guardarse y manejarse como datos ordinarios, es decir, en tablas dentro de la base de datos. Estos datos deben de estar disponibles a usuarios autorizados que usen SQL en base de datos.
5. **Sublenguaje completo de datos:** La base de datos relacional debe soportar un lenguaje bien definido y declarativo, con soporte para la definición de datos, definición de vista, manipulación de datos (interactiva y por programa), restricciones de integridad, autorización y administración de transacción (empezar, confirmar y retornar).
6. **Actualización de vista:** Cualquier vista que sea teóricamente actualizable debe ser actualizable a través del sistema.
7. **Inserción, actualización y eliminación de alto nivel:** La base de datos debe soportar inserciones, actualizaciones y eliminaciones a nivel de conjunto.
8. **Independencia física de los datos:** Los programas de aplicación y dispositivos ad hoc no son afectados lógicamente cuando se cambien métodos de acceso físico y estructurales de almacenamiento. (cambiar dispositivos de almacenamientos, dispositivo USB a dispositivo CD).
9. **Independencia lógica de datos:** Los programas de aplicación y dispositivos ad hoc no son afectados lógicamente cuando se hacen cambios a las estructuras de una tabla que preserven los valores originales de tabla (cambiar de orden de columnas o insertar columnas).
10. **Independencia de integridad:** Todas las restricciones de integridad relacionales deben ser definibles en el lenguaje relacional y guardadas en el catálogo del sistema, no al nivel de aplicación.
11. **Independencia de distribución:** Los usuarios finales y programas de aplicación no están enterados y no son afectados por la ubicación de datos (base de datos distribuidas vs bases de datos locales).
12. **No subversión:** Si el sistema soporta un acceso de bajo nivel a los datos, no debe haber forma de saltarse las reglas de integridad de la base de datos.
13. **Regla cero:** Todas las reglas precedentes están basadas en la noción de que para que una base de datos sea considerada relacional, debe usar dispositivos relacionales exclusivamente para manejar la base de datos.

# Capítulo 5

# Lenguaje estructurado de consultas (SQL)

En este capítulo se presenta el lenguaje SQL, utilizando el DBMS MySQL el cuál nos ayudará en el siguiente capítulo a su implementación con un ejemplo real. Los conceptos, consultas y sintaxis pueden ser utilizados para otros modelos de datos.

Las referencias que fueron utilizadas en éste capítulo son: [13], [19],[24], [25].

## 5.1. Introducción

El nombre SQL se construye a partir de las palabras en inglés *Structured Query Language*. Es común que algunos profesionistas se refieren a SQL como SEQUEL (Structured English QUEry Language) y fue diseñado e implementado en la compañia IBM Research como la interfaz para un sistema de base de datos relacional experimental llamado SISTEMA R [25].

Un esfuerzo conjunto con los organismos *American National Standards Institute* (ANSI) y el *International Standards Organization* (ISO) han dado lugar la norma (ANSI/ISO)[12] logrando que SQL sea un lenguaje estándar para todo software que maneje bases de datos relacionales, es decir, todas las bases de datos que soporten SQL deben tener la misma sintaxis a la hora de aplicar el lenguaje.

SQL se divide en 4 sublenguajes, que en conjunto permiten al DBMS cumplir con las funcionalidades requeridas por Codd.

- **Lenguaje DDL:** o lenguaje de definición de datos (*Data Definition Language*). Este lenguaje permite crear toda la estructura de una base de datos (desde tablas hasta usuarios). Sus comandos son del tipo DROP (Eliminar objetos) y CREATE (Crear objetos). En capítulos posteriores se estudiarán a detalle el lenguaje DDL.
- **Lenguaje DML:** o lenguaje de manipulación de datos (*Data Manupulation Language*). Este lenguaje permite con 4 comandos sencillos poder seleccionar determinados datos (SELECT), insertar datos (INSERT), modificarlos (UPDATE) o incluso borrarlos (DELETE). En las siguientes secciones se estudiarán a detalle la sintaxis de cada comando.

- **Lenguaje DCL:** o lenguaje de control de datos (*Data Control Language*). Incluye los comandos (`GRANT, REVOKE`) que permiten al administrador gestionar el acceso a los datos contenidos en la base de datos.
- **Lenguaje TCL:** o lenguaje de control de transacciones. El propósito de este lenguaje es permitir ejecutar varios comandos de forma simultánea como si fuera un comando atómico o indivisible. Si es posible ejecutar todos los comandos, se aplica la transacción (`COMMIT`), y si, en algún paso de la ejecución sucede algo inesperado, se pueden deshacer todos los pasos dados (`ROLLBACK`).

`SQL` contiene un conjunto básico de comandos o peticiones que tiene un vocabulario de menos de 100 palabras, por lo que puede ser fácil de aprender. Aunado a esto, `SQL` es un lenguaje que no es de procedimientos: solo se realiza la petición de lo que hay que hacer y no preocuparse de cómo lo va a realizar.

A pesar de que `SQL` es un lenguaje estándar existen pequeñas variaciones. Sin embargo las diferencias de estos son de poca importancia. Motivo por el cual en esta sección usaremos como `DBMS` a `MySQL`, por lo tanto, todo código mostrado será generado por `MySQL`.

Antes de generar código, es necesario instalar el `DBMS` que utilizará para ir de la mano con la ejecución de código. Por lo tanto la siguiente sección será dirigida a conocer `MySQL`.

## 5.2. Lenguaje de definición de datos (DDL)

En este apartado se estudiará la sintaxis de los diferentes comandos o peticiones para la definición de datos.

- `CREATE:` permite crear objetos como bases de datos, tablas, vistas, índices, triggers y procedimientos almacenados. Es decir, va acompañado de lo siguiente:
  - `DATABASE:` permite crear una base de datos
  - `TABLE:` permite crear una tabla, dentro de una base de datos.
  - `VIEW:` crea una tabla virtual basada en el conjunto de resultados de una declaración SQL.
  - `TRIGGER:` crea un script que contiene acciones a realizar en una base de datos, es decir, desencadena una serie de determinadas acciones de forma automática en las tablas de la base de datos cuando se insertan, modifican y se añaden nuevos datos.
  - `FUNCTION:` crea una rutina la cual es almacenada en el servidor para futuras consultas.
  - `SCHEMA:` crea un grupo de objetos de bases de datos que van pertenecer a un solo usuario o aplicación.
- `ALTER:` modifica la estructura de algún objeto ya creado.
- `DROP:` borra tablas, vistas y otros objetos (como índices) de la base de datos.
- `TRUNCATE:` suprime todas las filas de una tabla, es decir, borra el contenido mas no el objeto tabla.

Antes de todo se tienen que completar dos tareas importantes: primero, crear la estructura de base de datos y, segundo, crear las tablas que contendrán la base de datos.

Cuadro 5.1: Sintaxis `CREATE {DATABASE | SCHEMA}`

**Código general para crear una base de datos**

```
CREATE {DATABASE | SCHEMA} [IF NOT EXISTS] db_name
    [create_option] ...

create_option: [DEFAULT] {
    CHARACTER SET [=] charset_name
  | COLLATE [=] collation_name
  | ENCRYPTION [=] {'Y' | 'N'}
}
```

Del cuadro 5.1 se obtiene que `CREATE DATABASE` y `CREATE SCHEMA` son sinónimos. Se produce un error si la base de datos existe y no se especificó `IF NOT EXISTS`.

Cada `create_option` especifica una característica. Las características de la base de datos se almacenan en el diccionario de datos.

- La opción `CHARACTER SET` especifica el juego de caracteres predeterminado, es decir, si el usuario no introduce un dato, el sistema pone uno por defecto en la base de datos. La opción `COLLATE` especifica la intercalación predeterminada de la base de datos, es decir, va ordenando cada registro.
- La opción `ENCRYPTION` define el cifrado predeterminado de la base de datos, que heredan las tablas creadas en la base de datos. Los valores permitidos son 'Y' (cifrado habilitado) y 'N' (cifrado deshabilitado).

Cuadro 5.2: Sintaxis `CREATE TABLE`

**Código general para crear una tabla**

```
CREATE TABLE [IF NOT EXISTS] tbl_name(
    column1 datatype    characteristics1,
    column2 datatype    characteristics2,
    column3 datatype    characteristics3,
   ....

   column_definition1,
   column_definition2,
   column_definition3,
   ....
);
```

En el apartado `datatype` va el tipo de entrada que se le asignará a cada columna, por ejemplo, si se registra una fecha se tiene que definir el tipo de dato como `date`. En el cuadro 4.4 se definieron los datos más utilizados con su respectiva sintaxis en `MySQL`.

Para el caso de `characteristics` se tiene que definir qué propiedad tendrá la columna, por ejemplo, si es una columna tendrá como propiedad ser autoincrementable para

después ser una llave primaria, a continuación se mencionan algunas características que pueden tener las columnas:

1. `AUTO_INCREMENT`: Asigna a cada registro un número que no se repetirá, por lo general son valores autoincrementables de uno en uno, comúnmente se utiliza para obtener una llave primaria.
2. `NOT NULL`: Hace que sea obligatorio llenar el campo, es decir, no se puede dejar vacio.
3. `NULL DEFAULT "tex_defaul"`: En caso de dejar el blanco el campo, asigna un texto predeterminado.
4. `BINARY NULL`: Hace que se reciban valores booleanos. ({TRUE, FALSE}, {1, 0})
5. `UNSIGNED NULL`: Indica que si el campo contiene un número, éste no podrá ser negativo, haciendo que sea más óptimo la validación de datos en el campo, al tener un menor subconjunto de números. Solo si estas 100 % seguro de que no podrá tener negativos.
6. `ZEROFILL NULL`: Si el campo es numérico, rellena con 0 todas las posiciones a la izquierda para completar su longitud máxima.
7. `GENERATED ALWAYS AS () VIRTUAL`: Se puede considerar un atributo derivado, es decir, su valor proviene de alguna otra columna o columnas, por ejemplo, el resultado de multiplicar un campo por otro, podemos indicarle si queremos guardar ese valor (stored) o no(virtual).

En el caso de `colum_definition` se coloca la siguiente sintaxis:

1. `PRIMARY KEY()`: Hace que la columna sea una llave primaria, lo más común es que se defina antes `AUTO_INCREMENT`, se puede dar el caso de que el valor sea autoincrementable pero que no sea una llave primaria.
2. `UNIQUE INDEX()`: Garantiza que los valores en una o varias columnas sean únicos en todas las filas (o registros) de una tabla. Evita cualquier entrada duplicada en la tabla, incluso si no es una llave primaria.
3. `CHECK()` Realiza una validación de datos para cada registro, es decir, los valores ingresados tienen que cumplir cierta restricción que se haga mención dentro del parentesis.

Cuadro 5.3: Sintaxis `CREATE FUNCTION`

**Código general para crear una función**

```
CREATE FUNCTION [IF NOT EXISTS] function_name
([func_parameter[,...]]) RETURNS data_type
BEGIN
[characteristic ...] routine_body
END
);
```

Con `CREATE FUNCTION` se crea y guarda una rutina asociada a la base de datos. A continuación se detalla parte del comando.

- `function_name` es el nombre que se debe utilizar en las peticiones `SQL` para invocar la función.
- `(func_parameter[,...])` se colocan las variables de entrada que tendrá el proceso.
- `RETURNS` indica el tipo de valor de retorno de la función.
- `[characteristic ...] routine_body` se coloca el procedimiento a realizar.

Cuadro 5.4: Sintaxis `CREATE VIEW`

**Código general para crear una vista**

```
CREATE VIEW [IF NOT EXISTS] view_name AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;
);
```

Una petición de `VIEW` es una tabla virtual basada en el conjunto de resultados de una declaración `SQL`, contiene filas y columnas, además, los campos están compuestos de uno o más campos de tablas reales de la base de datos. Una petición de `VIEW` siempre muestra datos actualizados. El motor de base de datos recrea la vista cada vez que un usuario la consulta.

Cuadro 5.5: Sintaxis `CREATE TRIGGER`

**Código general para crear un disparador**

```
CREATE TRIGGER [IF NOT EXISTS] name_trigger
    trigger_time trigger_event
    ON tbl_name FOR EACH ROW
    [trigger_order]
    BEGIN
    trigger_body
    END

trigger_time: { BEFORE | AFTER }

trigger_event: { INSERT | UPDATE | DELETE }

trigger_order: { FOLLOWS | PRECEDES } other_trigger_name
```

Una petición de `TRIGGER` funciona como un disparador asociado a una tabla, el cual se activa cuando ocurre un evento en la tabla `(INSERT, UPDATE, DELETE)`. A continuación se detalla cada línea de la sintaxis.

- El comando `TRIGGER` se asocia con la tabla denominada `tbl_name`, que debe hacer referencia a una tabla permanente. No puede asociar un activador con una petición `TEMPORARY` tabla o una petición `VIEW`.
- El `trigger_name` se puede repetir siempre y cuando estén asociados a diferentes tablas.

- `trigger_time` es el tiempo de acción del disparador.Puede ser `BEFORE` o `AFTER` para indicar que el disparador se activa antes o después de cada fila a modificar.
- `trigger_event` indica el tipo de operación que activa el disparador.
- Es posible definir varios disparadores para una tabla determinada que tenga el mismo evento disparador y tiempo de acción. Por ejemplo, puede tener dos disparadores `BEFORE UPDATE` para una tabla. De manera predeterminada se activarán los disparadores en el orden en que hallan sido creadas, por lo tanto, si se requiere cambiar el orden de ejecución se tiene que especificar `trigger_order` con `FOLLOWS` o `PRECEDES`. Donde `FOLLOWS` indica que se activa después del disparador existente (`other_trigger_name`), con `PRECEDES` se activa antes.
- Por último, `trigger_body` es la declaración que se ejecutará cuando se active el disparador.

Una vez creados los objetos con los que se trabajará en la base de datos, es necesario tener una sintaxis para modificar alguna característica de los objetos ya creados, para ello usaremos `ALTER` seguido del objeto a modificar. Para fines prácticos se utilizarán los objetos ya creados con el comando `CREATE`.

Cuadro 5.6: Sintaxis `ALTER DATABASE`

**Código para modificar una base de datos**

```
ALTER {DATABASE | SCHEMA} [db_name]
    alter_option ...

alter_option: {
    [DEFAULT] CHARACTER SET [=] charset_name
  | [DEFAULT] COLLATE [=] collation_name
  | [DEFAULT] ENCRYPTION [=] {'Y' | 'N'}
  | READ ONLY [=] {DEFAULT | 0 | 1}
}
```

Recordemos que `DATABASE` es sinónimo de `SCHEMA` y en cada opción de `alter_opcion` se tienen los mismos parámetros mostrados en el cuadro 5.1 con la nueva opción de `READ ONLY` que controla si se permite la modificación de la base de datos y los objetos que contiene. Con los valores `DEFAULT`, 0 (modificable) y 1 (solo lectura). Esta opción es útil para la migración de bases de datos porque una base de datos, para la cual `READ ONLY` está habilitada, se puede migrar a otra instancia de `MySQL` sin preocuparse de que la base de datos pueda cambiar durante la operación.

En una tabla se pueden modificar, añadir y eliminar las columnas. También se pueden realizar modificaciones de las características de cada columna. El comando `ALTER TABLE` cambia la estructura de una tabla.

Cuadro 5.7: Sintaxis `ALTER TABLE`

**Código para modificar una tabla**

```
ALTER TABLE tbl_name
    [alter_option [, alter_option] ...]
alter_option: {
    table_options
  | ADD [COLUMN] col_name column_definition
        [FIRST | AFTER col_name]
  | ADD [COLUMN] (col_name column_definition,...)
  | DROP [COLUMN] col_name
  | ALTER [COLUMN] col_name {
        SET DEFAULT {literal | (expr)}
      | SET {VISIBLE | INVISIBLE}
      | DROP DEFAULT
    }
```

Cuadro 5.8: Sintaxis `ALTER FUNCTION`

**Código para modificar una función**

```
ALTER FUNCTION func_name [characteristic ...]

characteristic: {
    COMMENT 'string'
  | LANGUAGE SQL
  | { CONTAINS SQL | NO SQL | READS SQL DATA |
  MODIFIES SQL DATA }
  | SQL SECURITY { DEFINER | INVOKER }
}
```

El comando `ALTER FUNCTION` se puede utilizar para cambiar las características de una función almacenada. Se puede especificar más de un cambio en una petición `ALTER FUNCTION`. Sin embargo, no se pueden cambiar los parámetros o el cuerpo de una función almacenada utilizando esta declaración; para realizar dichos cambios, debe eliminar y volver a crear la función usando `DROP FUNCTION` y `CREATE FUNCTION`.

Esta declaración cambia la definición de comando `VIEW`, que ya existe. La sintaxis es similar a la de `CREATE VIEW`.

Por último, veremos la sentencia `DROP` la cual borra tablas, vistas, sinónimos, alias y otros objetos (como índices) de la base de datos. La sentencia `DROP` es muy genérica puesto que no se requiere definir otros parámetros, simplemente se coloca el nombre el objeto `DATABASE, TABLE, VIEW, TRIGGER`, ect.

Se debe tener cuidado antes de eliminar una base de datos o algún objeto. ¡Eliminar una base de datos resultará en la pérdida de la información completa almacenada en la base de datos!. En algunos casos se require autorización para utilizar la sentencia `DROP`.

Cuadro 5.9: Sintaxis ALTER VIEW

**Código para modificar una vista**

```
ALTER
    [ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}]
    [DEFINER = user]
    [SQL SECURITY { DEFINER | INVOKER }]
    VIEW view_name [(column_list)]
    AS select_statement
    [WITH [CASCADED | LOCAL] CHECK OPTION]
```

Cuadro 5.10: Sintaxis DROP

**Código para eliminar un objeto**

```
DROP {SQL_OBJECT_NAME} [IF EXISTS] name
```

Cuadro 5.11: Sintaxis RENAME TABLE

**Código para renombrar una tabla**

```
RENAME TABLE
    tbl_name TO new_tbl_name
    [, tbl_name2 TO new_tbl_name2] ...
```

Esta sintaxis es muy sencilla `RENAME TABLE` cambia el nombre de una o más tablas.

La petición `TRUNCATE TABLE` vacía una tabla por completo. Aunque parezca similar a `DELETE` se clasifica como una declaración DDL en lugar de una declaración DML. Se diferencía de `DELETE` en las siguientes formas:

- Las operaciones de truncar descartan y vuelven a crear la tabla, lo cual es mucho más rápido que eliminar filas una por una, particularmente para tablas grandes.
- Las operaciones de truncamiento no se pueden realizar si la sesión mantiene un bloqueo de tabla activo.

## 5.3. Lenguaje de manipulación de datos (DML)

Una vez creado el objeto tabla, se tienen que registrar los datos que permite cada columna, dando forma a una base de datos. En esta sección se analizaran los comandos básicos `INSERT, SELECT, UPDATE, DELETE` entre otros.

Con la petición `INSERT INTO` se insertan nuevos registros en una tabla existente. `tbl_name` es el nombre de la tabla en la que se deben insertar las filas. Después se tiene que especificar a qué columnas se le estarán realizando los registros y por último, se tiene que realizar el registro colocando `VALUES` antes del listado. Para poder ingresar datos correctamente siga lo siguiente:

- El registro se realiza entre paréntesis, donde `value_list` hace referencia a todo un registro.

Cuadro 5.12: Sintaxis `TRUNCATE TABLE`

**Código para truncar una tabla**

```
TRUNCATE [TABLE] tbl_name
```

Cuadro 5.13: Sintaxis `INSERT INTO`

**Código para ingresar datos a una tabla**

```
INSERT INTO tbl_name
    [(col_name [, col_name] ...)]
    VALUES (value_list) [, (value_list)] ...
```

- Los valores de caracteres y de fecha deben ingresarse entre apóstrofos (').
- Las entradas numéricas no se encierran entre apóstrofos.
- Las entradas de atributo se separan en comas.

Cuadro 5.14: Sintaxis `SELECT`

**Código seleccionar datos**

```
SELECT
    [(col_name [, col_name] ...)]
    FROM table_references
    [WHERE where_condition]
    [ORDER BY] col_name
    [ASC | DESC], ... [WITH ROLLUP]]
```

El comando `SELECT` se utiliza para recuperar filas seleccionadas de una o más tablas. La lista de columnas `[(col_name [, col_name] ...)]` representa uno o más atributos de la tabla de referencia, pero cuando se coloca `SELECT * FROM table_references` el asterisco $*$ hace referencia a todos los atributos de la tabla.

Para hacer consultas un poco más estructuradas en los renglones, se utiliza la petición `WHERE` para poder agregar restricciones condicionales al enunciado `SELECT`, pero, si no hay renglones que correspondan a los criterios especificados en el comando `WHERE`, no se mostrarán resultados.

Se pueden tener numerosas restricciones condicionales en el contenido de la tabla seleccionada como pueden ser:

- Operadores de comparación.

Ejemplo: Muestra todos los campos donde se cumple que los valores de `colum1` son iguales o menores a 58.

```
SELECT * FROM table_references
    WHERE colum1 <= 58
```

- Operadores lógicos.

Ejemplo: Se muestran todos los campos que cumplen exactamente con las dos condiciones mencionadas

| Símbolo | Significado |
|---|---|
| = | Igual a |
| < | Menor que |
| <= | Menor igual |
| > | Mayor que |
| >= | Mayor igual |
| <> o != | Diferente de |

| Operador | Significado |
|---|---|
| AND | Valida el cumplimiento exacto de dos o más restricciones. |
| OR | Valida que al menos una restricción se cumpla. |
| NOT | Valida el complemento de una restricción |

```
SELECT * FROM table_references
    WHERE colum1 <= 58 AND colum2='Pick Up'
```

- Operadores especiales.

| Operador | Significado |
|---|---|
| BETWEEN | Verifica si un valor de atributo está dentro de un margen. |
| IS NULL | Verifica si un valor es nulo |
| LIKE | Verifica si un valor de atributo satisface un patrón determinado de cadena |
| IN | Verifica si un valor de atributo corresponde con cualquier atributo dentro de una lista de valores |
| EXISTS | Verifica si una subconsulta entrega algunos renglones |

Ejemplo: Se muestran las columnas 2, 3 y 8, donde en la columna 3 se muestran todos los nombres que empiecen con Jo.

```
SELECT colum2, colum3, colum8 FROM table_references
    WHERE colum3 LIKE Jo%
```

Una de las ventajas de las consultas en MySQL es su capacidad para producir consultas complejas con gran facilidad a través de su lenguaje. Por ejemplo, con los comandos que ya fueron expuestos se pueden realizar muchas combinaciones y obtener consultas bien elaboradas.

Teniendo en cuenta lo anterior, veremos algunos casos especiales y funciones de consulta que se realizan con otras peticiones. Tal es el caso de ORDER BY que es muy útil cuando

uno requiere ordenar los resultados.

Ejemplo: Muestra todos los campos donde los valores de la columna 3 estén entre 45 y 100, además, ordena los datos de la columna 1 de forma ascendente.

```
SELECT * FROM table_references
    WHERE colum3 BETWEEN 45 AND 100
    ORDER BY colum1 ASC
```

En MySQL se pueden efectuar varios resúmenes matemáticos; por ejemplo, contar el número de renglones que contenga una condición especificada, encontrar valores mínimos y máximos para algún atributo especificado, sumar los valores en una columna especificada o bien, promediar los valores. Estas funciones se muestran en el cuadro 5.15.

Cuadro 5.15: Funciones básicas de consulta

| Función | Salida |
|---|---|
| COUNT | El número de renglones que contengan valores no nulos |
| MIN | Valor mínimo de atributo que se encuentre en una columna dada |
| MAX | Valor máximo que se encuentre en una columna dada |
| SUM | Suma de todos los valores para una columna determinada |
| AVG | Media aritmética para una columna especificada |

También podemos realizar operaciones aritméticas con los campos de la tabla. Cuando se realicen operaciones matemáticas en atributos, hay que recordar la jerarquía de operaciones que son las que establecen el orden en que se completan los cálculos. Por ejemplo, nótese el orden de la siguiente secuencia computacional:

1. Realice operaciones dentro de paréntesis.
2. Realice operaciones de potencias.
3. Realice multiplicaciones y divisiones.
4. Realice sumas y restas.

Ejemplo:Ordena toda la tabla de forma ascendente de la columna ciudad si son mayores al promedio de consumo del producto x.

```
SELECT * FROM Ventas
    WHERE price >= AVG(precio*cantidad)
    ORDER BY ciudad ASC
```

Las distribuciones de frecuencia se pueden crear rápida y fácilmente con el comando GROUP BY y también se le puede asignar una condición con HAVING. La cláusula GROUP BY se usa generalmente cuando el usuario tiene columnas de atributo combinadas con funciones agregadas en el enunciado SELECT. También hay que aclarar que es válida cuando se usan funciones básicas de consulta del cuadro 5.15. A continuación se muestra un ejemplo del comando ORDER BY, suponga una tabla donde se guardan las ventas de una tienda, donde hay campos de vendedor y los productos que vendieron.

Cuadro 5.16: Agrupamiento de datos con GROUP BY y HAVING

**Código seleccionar datos**

```
SELECT
    [(col_name [, col_name] ...)]
    FROM table_references
    [WHERE where_condition]
    [GROUP BY col_name]
    [HAVING col_name]
    [ORDER BY] col_name
    [ASC | DESC], ... [WITH ROLLUP]]
```

El siguiente código nos muestra una tabla con la cantidad de productos vendidos por cada vendedor.

```
SELECT Vendedor, COUNT(Producto) FROM Ventas
    GROUP BY Vendedor
```

Una extensión particularmente útil de la función GROUP BY es el comandoHAVING, que opera de una manera muy semejante al comando WHERE del SELECT, con la diferencia de que SELECT se aplica a una salida de una operación y expresión de renglones individuales, mientras que HAVING se aplica a la salida de una operación GROUP BY.

A continuación se muestra el código de los vendedores que obtuvieron más de 20 ventas, ordenados de forma descendente.

```
SELECT Vendedor, COUNT(Producto) FROM Ventas
    GROUP BY Vendedor
    HAVING COUNT(Producto)>=20
    ORDER BY Vendedor DESC
```

Cuadro 5.17: Sentencia UPDATE

**Código seleccionar datos**

```
UPDATE table_reference
    SET assignment_list
    [WHERE where_condition]
    [ORDER BY ...]
    [LIMIT row_count]
```

La instrucción UPDATE actualiza las columnas de las filas existentes en la tabla nombrada con nuevos valores. El comando SET indica qué columnas modificar y los valores que se les deben dar. Cada valor se puede proporcionar como una expresión o la palabra clave DEFAULT para establecer una columna explícitamente en su valor predeterminado. El comando WHERE, si se proporciona, especifica las condiciones que identifican qué filas actualizar. Sin el comando WHERE, todas las filas se actualizan. Si se especifica la cláusula ORDER BY, las filas se actualizan en el orden especificado. El comando LIMIT impone un límite al número de filas que se pueden actualizar.

En el siguiente código se muestran los primeros 10 clientes con mayor monto que tienen

el CP=86600, además, se realiza una actualización en el campo Ciudad, en este caso "Paraiso", donde la columna CP es igual a 86600.

```
UPDATE Clientes
    SET Ciudad= 'Paraiso'
    WHERE CP=86600
    ORDER BY Monto DESC
    LIMIT 10
```

Ahora analizaremos una petición que su uso requiere de mucho cuidado y es utilizada para eliminar registros en una tabla. El comando `DELETE` elimina filas de una tabla declarada y devuelve el número de filas eliminadas.

Cuadro 5.18: Sentencia `DELETE`

| **Código para eliminar datos** |
|---|
| `DELETE FROM table_name`<br>`    [WHERE where_condition]` |

Como ya se explico anteriormente la declaración `DELETE` se utiliza para eliminar registros existentes en una tabla. Su sintaxis es sencilla; las condiciones de la petición `WHERE` identifican qué filas eliminar. Sin la petición `WHERE`, se eliminan todas las filas. Por lo tanto, hay que usar con precaución esta petición.

## 5.4. Operadores relacionales de conjunto

Dentro del lenguaje de manipulación de datos, hay operaciones orientadas a conjuntos; esto es, que operan en conjuntos completos de renglones y columnas a la vez. Con el uso de conjuntos, se pueden combinar dos o más conjuntos para crear nuevos conjuntos (o relaciones). Para poder realizar todas estas operaciones se trabajará con los enunciados `UNION, INTERSECT` y `EXCEPT`. Para poder utilizar los enunciados y las tablas con las que se trabaje, éstas deben ser compatibles, esto es, deben entregar el mismo número de atributos y tipos similares de datos.

```
CREATE TABLE t1 (x INT, y INT);
INSERT INTO t1 VALUES ROW(4,-2), ROW(5,9),ROW(1,2);
CREATE TABLE t2 (a INT, b INT);
INSERT INTO t2 VALUES ROW(1,2), ROW(3,4);
```

`SELECT*FROM t1;`

| x | y |
|---|---|
| 4 | -2 |
| 5 | 9 |
| 1 | 2 |

`SELECT*FROM t2;`

| a | b |
|---|---|
| 1 | 2 |
| 3 | 4 |

- `UNION`: combina la salida de dos consultas `SELECT` o en su caso dos tablas. Para este operador, notar que si se repiten valores, la salida será solo un valor y no dos.

  Ejemplo:

```
TABLE t1 UNION TABLE t2;
```

| x | y |
|---|---|
| 4 | -2 |
| 5 | 9 |
| 1 | 2 |
| 3 | 4 |

```
SELECT * FROM t2 UNION SELECT * FROM t1;
```

| a | b |
|---|---|
| 1 | 2 |
| 3 | 4 |
| 4 | -2 |
| 5 | 9 |

- `UNION ALL`: combina la salida de dos consultas `SELECT` o en su caso dos tablas. A diferencia de `UNION`, en esta operación se muestran los valores repetidos.

  Ejemplo:

  ```
  SELECT*FROM t1 UNION ALL SELECT*FROM t2;
  ```

| x | y |
|---|---|
| 4 | -2 |
| 5 | 9 |
| 1 | 2 |
| 1 | 2 |
| 3 | 4 |

- `INTERSECT`: muestra los registros que están en el primer conjunto de datos que también están en el segundo conjunto de datos.

  Ejemplo:

  ```
  TABLE t2 INTERSECT TABLE t1;
  ```

| a | b |
|---|---|
| 1 | 2 |

- `EXCEPT`: muestra los registros que pertenecen al primer conjunto de datos sin los registros que comparte con otro conjunto, es decir, se le restan los datos del otro conjunto.

  ```
  TABLE t1 EXCEPT TABLE t2;
  ```

| x | y |
|---|---|
| 4 | -2 |
| 5 | 9 |

## 5.5. Lenguaje de control de transacciones (TCL)

En este apartado, se mencionarán algunas sentencias utilizadas para el lenguaje de transacciones con sus usos. No se adentrará en el tema pero quedará una noción de lo que significa.

Una transacción es una unidad de la ejecución de un programa. Puede consistir en varias operaciones de acceso a la base de datos [14]. Está delimitada por constructoras como `begintransaction` y `end-transaction`.

Ejemplos de transacciones:

- Depósito de una cuenta A a una B.
- Cálculo de una prima para cierto cliente.
- Solicitud automática de proveedores dado el análisis de inventarios.

Como se observa en los ejemplos mostrados una transacción es un proceso que se realiza con las bases de datos, la cual tiene un objetivo específico. Por lo tanto, es necesario que la transacción se cumpla o no se cumpla. No es posible que se quede sin completar.

En bases de datos se denomina **ACID** a las características de los parámetros que permiten clasificar las transacciones de los sistemas de gestión de bases de datos.

1. **Atomicity**(Atomicidad): es la propiedad de las transacciones que permite observarlas como operaciones atómicas: ocurren totalmente o no ocurren.
2. **Concistency**(Consistencia): la ejecución aislada de la transacción conserva la consistencia de la base de datos. Cualquier cambio debe conducir de un estado válido de la base de datos a otro estado válido de acuerdo con las restricciones y el esquema de datos.
3. **Isolation**(Aislamiento): para cada par de transacciones que puedan ejecutarse concurrentemente $T_i$ y $T_j$, se cumple que para los efectos de $T_i$:
   - $T_j$ ha terminado antes de que comience $T_i$
   - $T_j$ ha comenzado después de que termine $T_i$
   - Las transacciones son independientes entre si.

   Un cambio no debe afectar a otros cambios que se estén ejecutando al mismo tiempo sobre la base de datos.
4. **Durability**(Durabilidad): el sistema gestor de bases de datos asegura que perduren los cambios realizados por una transacción que termina con éxito. Es decir, una vez completado el cambio, éste debe conservarse, aunque se produzcan fallos en la base de datos o el sistema completo.


Figura 5.1: Propiedades de las transacciones

Estas declaraciones proporcionan control sobre el uso de las transacciones:

- `START TRANSACTION` o `BEGIN` da comienzo a una nueva transacción.
- `COMMIT` confirma la transacción actual, haciendo que sus cambios sean permanentes.
- `ROLLBACK` revierte la transacción actual y cancela sus cambios.
- `SET autocommit` deshabilita o habilita el modo de confirmación automática predeterminado para la sesión actual.

De forma predeterminada, `MySQL` se ejecuta con el modo de confirmación automático habilitado. Esto significa que, cuando no está dentro de una transacción, cada declaración es atómica, como si estuviera rodeada por `START TRANSACTION` y `COMMIT`. No puede utilizar `ROLLBACK` para deshacer el efecto; sin embargo, si se produce un error durante la ejecución de la declaración, la declaración se revierte.

Cuadro 5.19: Sentencia `START TRANSACTION, COMMIT y ROLLBACK`

**Uso**

```
START TRANSACTION
    [transaction_characteristic
    [, transaction_characteristic] ...]

transaction_characteristic: {
    WITH CONSISTENT SNAPSHOT
  | READ WRITE
  | READ ONLY
}

BEGIN [WORK]
COMMIT [WORK] [AND [NO] CHAIN] [[NO] RELEASE]
ROLLBACK [WORK] [AND [NO] CHAIN] [[NO] RELEASE]
SET autocommit = {0 | 1}
```

## 5.6. Lenguaje de control de datos (DCL)

En este apartado se hará mención del funcionamiento que tienen algunas sentencias usadas para el control de acceso a los datos y a su contenido.

La seguridad comienza con el usuario administrativo o el súper usuario que se crea al instalar `MySQL`, como se hace mención en la figura A.6. Como usuario administrativo, debe crear y autorizar a otros usuarios. Cuando crea usuarios por primera vez, no pueden ver o hacer nada. Cuando otorga a los usuarios más privilegios, ellos pueden acceder a más objetos de base de datos.

- Para el comando `IDENTIFIED BY` define la contraseña que será asignada al usuario creado.
- La petición `DEFAULT ROLE` define qué roles se activan cuando el usuario se conecta al servidor y se autentica. Estos roles deben existir en el momento en que se ejecuta `CREATE USER`; de lo contrario, la declaración genera un error.

Cuadro 5.20: Sentencia `CREATE USER`

**Crear nuevo usuario.**

```
CREATE USER [IF NOT EXISTS]
    user name_user
    IDENTIFIED BY 'password_string'
    DEFAULT ROLE role [, role ] ...
    [REQUIRE {NONE | tls_option [[AND] tls_option] ...}]
    [WITH resource_option [resource_option] ...]
    [password_option | lock_option] ...
    [COMMENT 'comment_string' | ATTRIBUTE 'json_object']
```

- Para establecer un comentario de usuario, agregue `COMMENT` a la declaración `CREATE USER`, donde `user_comment` es el texto del comentario del usuario.

Cuadro 5.21: Sentencia `CREATE ROLE`

**Crear roles**

```
CREATE ROLE [IF NOT EXISTS] role [, role ] ...
```

La petición CREATE ROLE crea uno o más roles, que se denominan colecciones de privilegios.

Cuando se crea un rol está bloqueado, no tiene contraseña y se asigna por default con una identificación predeterminada.

Cuadro 5.22: Sentencia `GRANT`

**Asignar roles y privilegios a usuarios.**

```
GRANT
    priv_type [(column_list)]
      [, priv_type [(column_list)]] ...
    ON [object_type] priv_level
    TO user_or_role [, user_or_role] ...
    [WITH GRANT OPTION]
    [AS user
        [WITH ROLE
            DEFAULT
          | NONE
          | ALL
          | ALL EXCEPT role [, role ] ...
          | role [, role ] ...
        ]
    ]
```

La declaración `GRANT` permite a los administradores del sistema otorgar privilegios y roles, que se pueden otorgar a cuentas y roles de usuario. Se aplican estas restricciones de sintaxis:

- GRANT no puede combinar la concesión de privilegios y roles en la misma declaración. Una declaración GRANT determinada debe otorgar privilegios o roles.
- El comando ON distingue si la declaración otorga privilegios o roles:
  - Con ON, la declaración otorga privilegios.
  - Sin ON, la declaración otorga roles.
  - Está permitido asignar privilegios y roles a una cuenta, pero debe usar declaraciones GRANT separadas, cada una con una sintaxis adecuada a lo que se va a otorgar.

La última sentencia es la declaración REVOKE que permite a los administradores del sistema revocar privilegios y roles, que pueden revocarse de cuentas y roles de usuario.

Cuadro 5.23: Sentencia REVOKE

**Quitar roles y privilegios a usuarios.**

```
REVOKE [IF EXISTS]
    priv_type [(column_list)]
      [, priv_type [(column_list)]] ...
    ON [object_type] priv_level
    FROM user_or_role [, user_or_role] ...
    [IGNORE UNKNOWN USER]
```

A continuación se detallan los requerimientos para la sentencia REVOKE:

- priv_type: hace mención a que privilegios se van a quitar, los cuales pueden ser:
  - INSERT
  - UPDATE
  - DELETE

  Entre otros más.
- object_type: hace mención a que objetos de la base de datos queremos quitar privilegios, puede ser una o todas las tablas de la base de datos.
- user_role: se indica que usuario o usuarios le estamos revocando privilegios.

# Capítulo 6

# Ejemplo

En este capítulo se presenta un modelo de datos para una empresa de distribución. En el modelo se presentan las reglas de negocio que son propias de la institución, además se muestra el código `SQL` que fue implementado.

## 6.1. Conocimiento del entorno

Antes de implementar algún diagrama es necesario entrar en contexto, conocer los procesos, visualizar si hay algún flujo, conocer las reglas de negocio y si es necesario entrevistar a los involucrados del proceso.

En este ejemplo, se tiene una empresa distribuidora de productos de la canasta básica de consumo. Su proceso consiste en ser intermediario entre compradores que venden en retail o venden comida y vendedores al por mayor, es decir, se encarga de suministrarles productos a tiendas, restaurantes, fruterías, etc. a través de proveedores a mayoristas.

La empresa distribuidora tiene el nombre de DISTRIBUIDORA MORIZ la cual ofrece un servicio de logística efectivo a sus clientes.

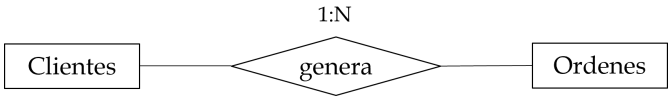
El diseño de la base de datos será sencillo, MORIZ solo necesita almacenar sus datos para poder realizar consultas que le ayuden a tomar decisiones.

Su flujo es el siguiente:

- Los empleados vigilan que los productos que ofrecen estén disponibles, en caso contrario contactan al proveedor.
- MORIZ tiene empleados que se encargan de realizar órdenes con sus clientes.
- Una vez capturada la orden se le asigna un camión para transportar los productos.

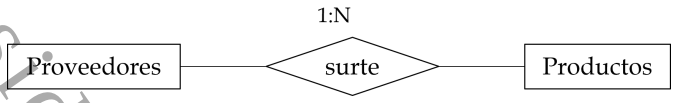
## 6.2. Reglas de negocio y diagramas

1. Un cliente puede generar muchas órdenes.



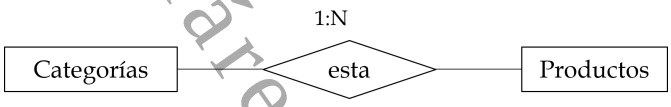
2. Un empleado puede realizar muchas órdenes.


3. Un proveedor puede surtir muchos productos.



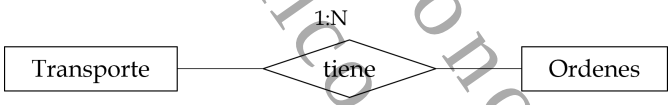

   Los productos están almacenados y distribuidos por categorías.

4. Una categoría puede tener muchos productos, pero un producto no puede pertenecer a más de una categoría.

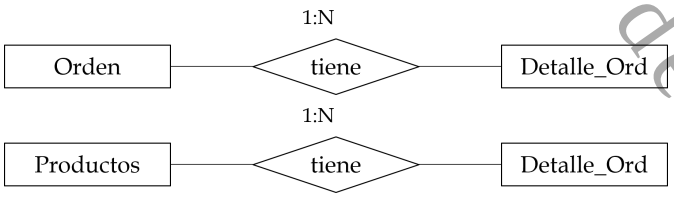



   MORIZ necesita trasladar sus productos a sus clientes por lo que decide contratar servicios de transporte.

5. Una empresa de transporte puede llevar muchas órdenes.




6. Se crea una entidad donde se guarden los detalles de las órdenes, mismo que está relacionado con la entidad orden y productos.



La forma adecuada de poder manejar estas relaciones de entidades es a través de un ID para cada entidad y el buen manejo de las llaves foráneas. Realizando un análisis se tiene el siguiente diagrama relacional.

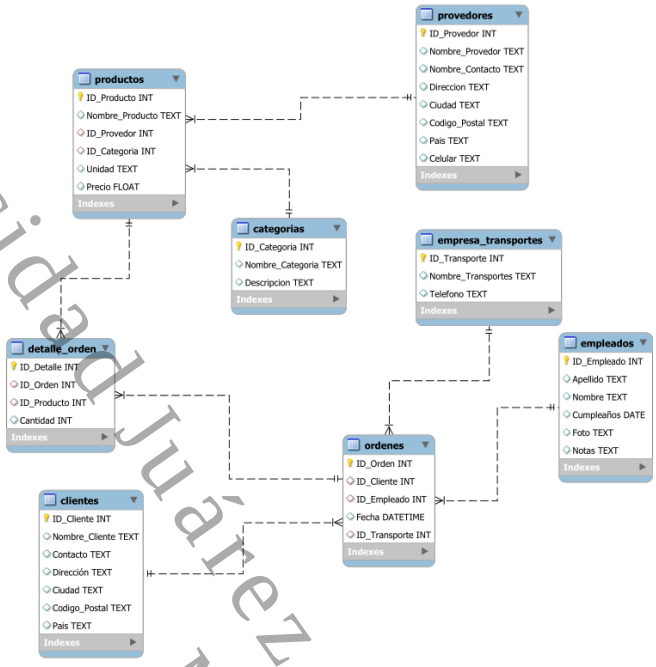



Figura 6.1: Diagrama relacional para MORIZ

El código para generar el diagrama relacional de la figura 6.1 se presenta en el cuadro 6.1.

Cuadro 6.1: Código para el diagrama relacional

**Tablas independientes, que determinan...**

```
CREATE TABLE Categorias(
    ID_Categoria INTEGER PRIMARY KEY auto_increment,
    Nombre_Categoria TEXT,
    Descripcion TEXT);
CREATE TABLE Clientes(
    ID_Cliente INTEGER PRIMARY KEY auto_increment,
    Nombre_Cliente TEXT,
    Contacto TEXT,
    Dirección TEXT,
    Ciudad TEXT,
    Codigo_Postal TEXT,
    Pais TEXT);
CREATE TABLE Empleados(
    ID_Empleado INTEGER PRIMARY KEY auto_increment,
    Apellido TEXT,
    Nombre TEXT,
    Cumpleaños DATE,
    Foto TEXT,
    Notas TEXT);
CREATE TABLE Empresa_Transportes(
    ID_Transporte INTEGER PRIMARY KEY auto_increment,
    Nombre_Transportes TEXT,
    Telefono TEXT);
CREATE TABLE Provedores(
    ID_Provedor INTEGER PRIMARY KEY auto_increment,
    Nombre_Provedor TEXT,
    Nombre_Contacto TEXT,
    Direccion TEXT,
    Ciudad TEXT,
    Codigo_Postal TEXT,
    Pais TEXT,
    Celular TEXT);
```

Una vez creada las tablas independientes, es decir, que determinan la existencia de otra entidad, se tienen que crear las relaciones. Tal como se muestra en el cuadro 6.2.

Para el llenado de datos, se aplica el lenguaje de manipulación de datos (DML), que van ligados al diseño que se mostró en la sección anterior.

Cuadro 6.2: Código para el diagrama relacional

**Tablas dependientes, existen si hay una entidad que determina**

```
CREATE TABLE Productos(
    ID_Producto INTEGER PRIMARY KEY auto_increment,
    Nombre_Producto TEXT,
    ID_Provedor INTEGER,
    ID_Categoria INTEGER,
    Unidad TEXT,
    Precio FLOAT DEFAULT 0,
        FOREIGN KEY (ID_Categoria)
        REFERENCES Categorias (ID_Categoria),
        FOREIGN KEY (ID_Provedor)
        REFERENCES Provedores (ID_Provedor));
CREATE TABLE Ordenes(
    ID_Orden INTEGER PRIMARY KEY auto_increment,
    ID_Cliente INTEGER,
    ID_Empleado INTEGER,
    Fecha DATETIME,
    ID_Transporte INTEGER,
    FOREIGN KEY (ID_Empleado)
    REFERENCES Empleados (ID_Empleado),
    FOREIGN KEY (ID_Cliente)
    REFERENCES Clientes (ID_Cliente),
    FOREIGN KEY (ID_Transporte)
    REFERENCES Empresa_Transportes (ID_Transporte));
CREATE TABLE Detalle_Orden(
    ID_Detalle INTEGER PRIMARY KEY auto_increment,
    ID_Orden INTEGER,
    ID_Producto INTEGER,
    Cantidad INTEGER,
        FOREIGN KEY (ID_Orden)
        REFERENCES Ordenes (ID_Orden),
        FOREIGN KEY (ID_Producto)
        REFERENCES Productos (ID_Producto));
```

Con el código generado se obtiene el modelo completo para su integración al sistema, en el desarrollo del código se muestra los enlacen que se tienen con cada una de las llaves foráneas, es decir, se muestran las dependencias que hay de una tabla a otra.

# Capítulo 7

# Conclusión

La evidencia que presentamos anteriormente demuestra la relevancia que tienen los Sistemas Gestores de Bases de Datos, su uso y las ventajas que tienen las instituciones al hacer uso de estos sistemas; principalmente como herramienta para el actuario.

Las instituciones privadas y gubernamentales suelen tener problemas de tomas de decisiones, aunque guarden datos y los analicen, en ocasiones no ven el problema que conlleva una mala administración de estos, sabiendo que es un problema que viene de raíz; y con ello nos referimos a un modelo mal planteado que se viene trayendo desde que la institución tomó otras medidas de negocio o reglas de negocio que pudieron afectar al modelo de raíz. De esta manera, se toman medidas para que un actuario pueda obtener conocimientos básicos de una buena administración de base de datos, que conozca la estructura de los datos relacionales y finalmente que se introduzca al lenguaje estructurado de consultas `SQL`.

La actualización de las reglas de negocios es importante, pero, también no hay que descuidar las actualizaciones de software, hardware y otros elementos que dependan para el buen manejo de las bases de datos. La administración y análisis de bases de datos es un plus para toda institución por lo tanto, toda institución se tiene que proyectar tomando como base sus datos.

En el capítulo 1 se presentó una introducción a las bases de datos relacionales, se mostraron ejemplos y se explicaron las funcionalidades de los DBMS.

En el capítulo 2 se estableció la relación que existe entre la actuaría y las bases de datos, se expuso su importancia y se mostró las etapas que tiene que realizar un actuario cuando tiene como materia prima los datos.

En el capítulo 3 se expuso los conceptos para un modelado de datos, seguidamente de los procedimientos a seguir cuando se tiene que normalizar una tabla de datos.

En el capítulo 4 se presentó el lenguaje `SQL` cómo herramienta para el desarrollo e implementación del modelo de datos en un DBMS.

Finalmente, en el último capítulo se presentó paso a paso cómo desarrollar un modelo de datos hasta su implementación en `SQL`.

Un modelo de datos a futuro que se puede implementar tomando en cuenta este trabajo es el diseño de un sistema de auditorías a tiendas comerciales.

# Apéndice A

# Instalación de MySQL en Microsoft Windows 11 (2024)

Para instalar MySQL en Windows hay que descargar la versión actual de MySQL, por lo tanto hay que dirigirse a la página Web https://www.mysql.com/ y buscar el instalador. Una vez que cargue la página hay que dirigirse a la parte de *Downloands*, luego se da click en *MySQL Community (GPL) Downloands* como se muestra en la figura A.1.

Al momento de dar click a *MySQL Community (GPL) Downloands* aparecerán una serie de complementos con los que trabaja MySQL; para nuestro caso nos interesa en primera instancia instalar MySQL Installer for Windows, seleccionamos y nos dirige a otro página, donde seleccionaremos el instalador completo (suele ser de mayor tamaño que el otro). Todo esto se indica en la figura A.2.

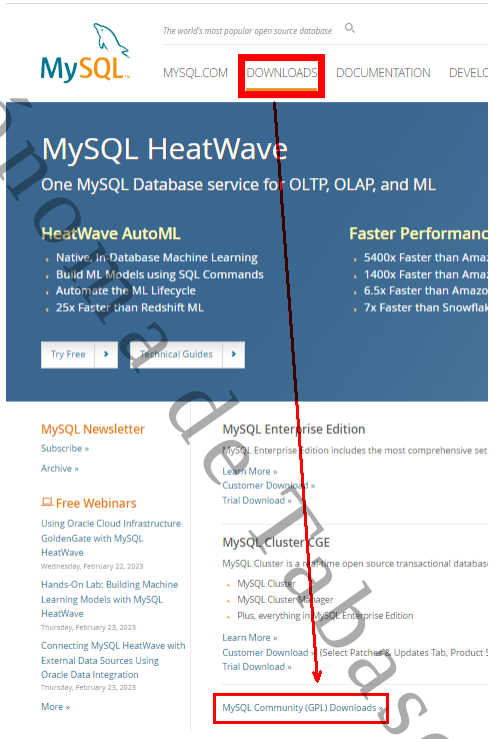
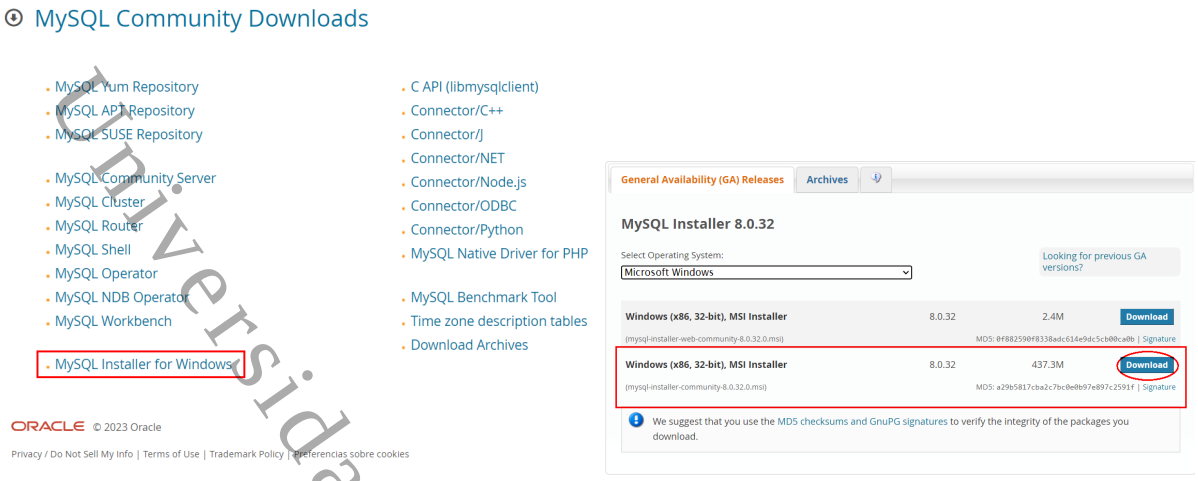



Figura A.1: Página MySQL

Una vez realizado lo anterior, tenemos que dar una ruta para guardar el instalador y así poder realizar la ejecución en el sistema Windows.

Una vez que se esté ejecutando el instalador, aparecerán una serie de opciones que van a determinar el tipo de instalación a definir, por ejemplo; la instalación predeterminada *(Default)* que proporciona una instalación adecuada y necesaria, como; MySQL Server y otras herramientas de MySQL relacionadas con el desarrollo de MySQL, como MySQL Workbench, otro tipo es la instalación servidor *(Only Server)* que instala solo MySQL Server sin otros productos, después sigue la instalación *Full* que instala todos los productos incluidos de MySQL y por último, la instalación del usuario *(Custom)* que permite seleccionar cualquier producto MySQL.[24]




(a) Productos (b) Versiones

Figura A.2: Descargar `MySQL Installer`

Antes de ejecutar el instalador, es necesario advertir que MySQL en su versión actual requiere de Microsoft Visual C++ 2019 y que la computadora sea de 64 bits para poder ejecutarse.

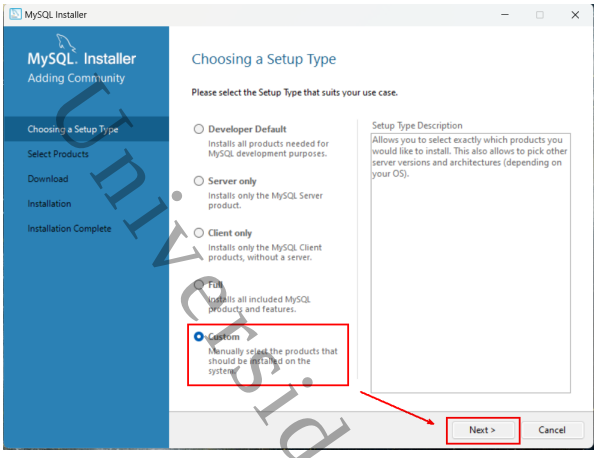
El instalador de MySQL es una aplicación independiente, diseñada para facilitar la complejidad de la instalación y configuración de productos MySQL que se ejecutan en Microsoft Windows.

Dentro del entorno instalador se muestra lo siguiente:

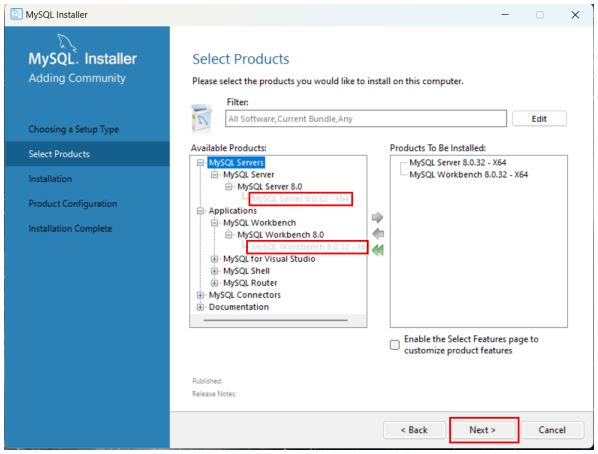
1. MySQL Server: proporciona un sistema de gestión de base de datos con capacidades de consulta y conectividad, así como la capacidad de tener una excelente estructura de datos e integración con muchas plataformas diferentes.
   - MySQL Server 8.0
   - MySQL Server 5.7
   - MySQL Server 5.6
2. MySQL Aplications: permite a los desarrolladores, arquitectos de datos y demás clientes diseñar, modelar, gestionar y generar bases de datos de manera visual o gráfica.
   - MySQL Workbench
   - MySQL Shell
   - MySQL Router
   - MySQL for Visual Studio
3. MySQL Connectors: proporciona conectividad al servidor MySQL para programas cliente, es decir, permite conectar y ejecutar comandos de MySQL desde otro lenguaje o entorno.
   - MySQL Connector/Python
   - MySQL Connector/NET
   - MySQL Connector/ODBC
   - MySQL Connector/C++
   - ect.
4. Documentation and Samples: incluye manuales de referencia y ejemplos.

Para nuestro interés usaremos la opción de instalación *Custom* la cual consiste en utilizar el servidor MySQL Server 8.0.32 con la interfaz MySQL Workbench sin instalar MySQL Connectors, puesto que no será de utilidad en este trabajo, queda al lector de manera opcional ejecutar *Documentation and Samples* para su crecimiento académico.

A continuación se muestran 11 pasos a seguir para poder instalar con éxito la instalación del usuario *(Custom)* de MySQL:



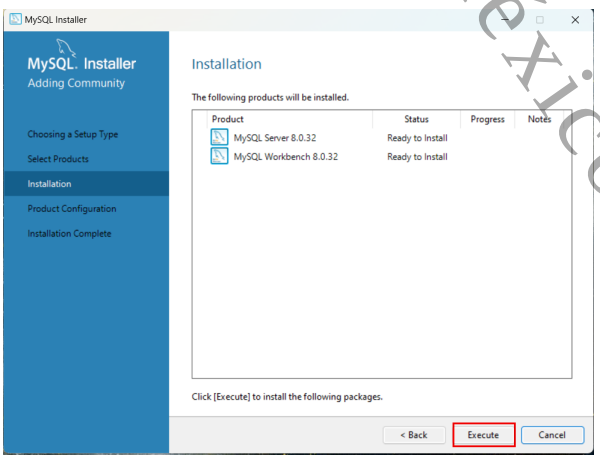

(a) Instalar *Custom*



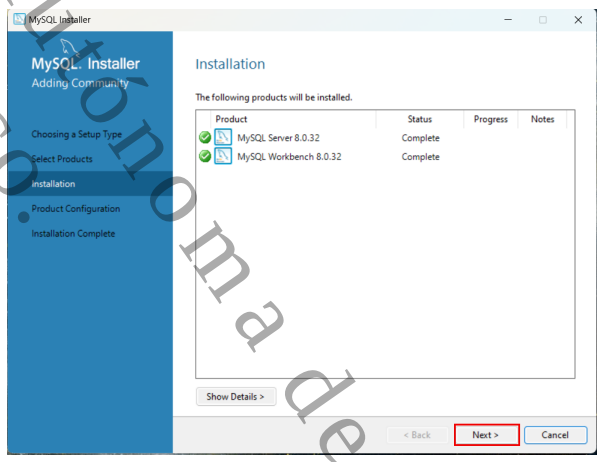

(b) Seleccionar Productos

Figura A.3: Paso 1-2

En la figura A.3 en la parte (a) se selecciona la instalación de usuario, después se da clic en Next> y se observa en la ventana (b) una serie de opciones a instalar; para nuestro caso, solo se instalará `MySQL Server 8.0` en su versión actual y `MySQL Workbench` , a continuación dar clic en Next>.




(a) Ejecutar instalación

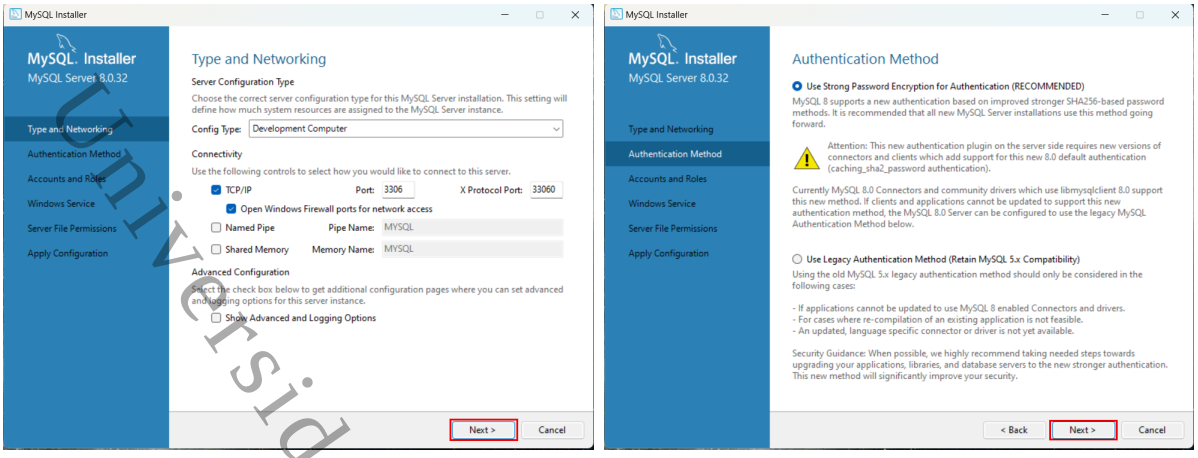



(b) Instalación terminada

Figura A.4: Paso 3-4

En la figura A.4 parte (a) se mostrarán los productos seleccionados en la figura A.3, por lo tanto, se procede a dar clic en Execute, después, en la figura (b) se muestra una ventana con los productos correctamente instalados.

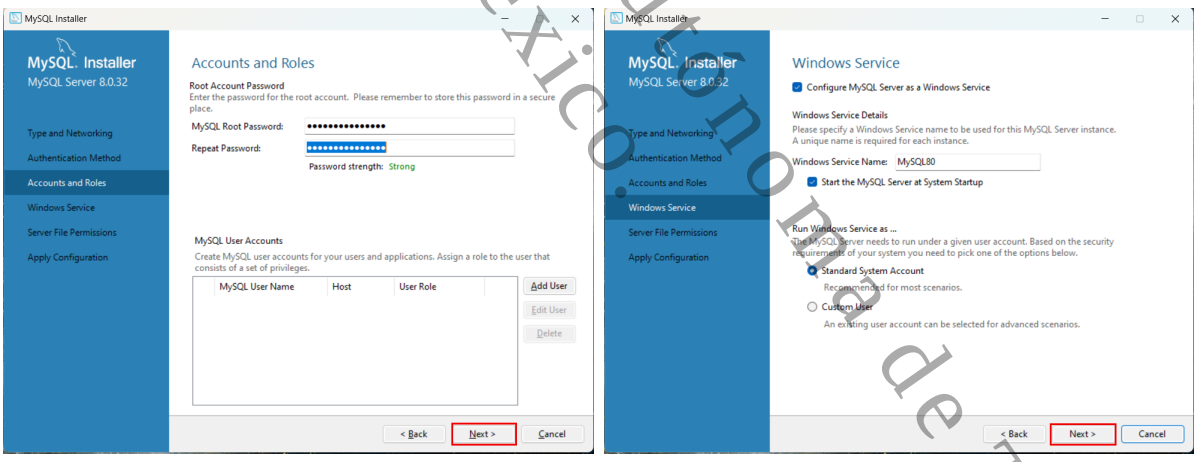
En la figura A.5 parte (a) se muestra el tipo de configuración, el cual se dejará por defecto al igual que lo mostrado en la parte (b), esto se consigue dando clic en Next> para ambos casos.




(a) Configuración del servidor (b) Método de autentificación

Figura A.5: Paso 5-6

En la figura A.6 parte (a) se crea un superusuario, el cual consiste en manejar toda la información y dar permiso a los demás usuarios, por lo tanto, es importante colocar una contraseña bien diseñada, después de colocar la contraseña, dar clic en Next> al igual que en la parte (b).

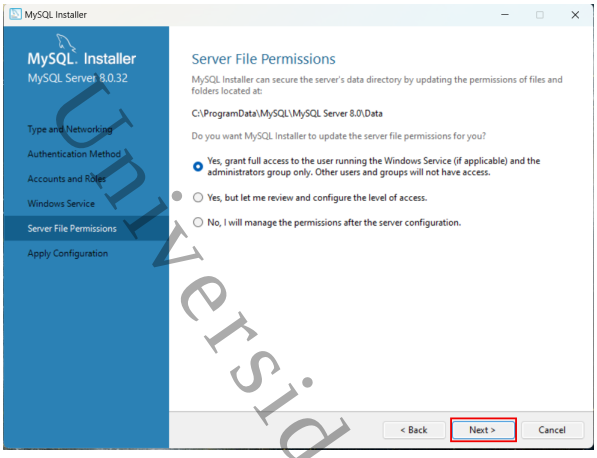



(a) Creación de superusuario (b) Servicios Windows

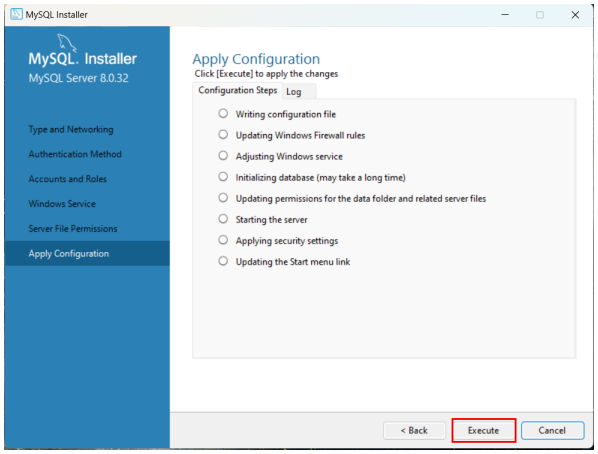
Figura A.6: Paso 7-8

En la figura A.7 parte (a) permite que `MySQL` actualice el servidor y en la parte (b) hace las actualizaciones correspondientes de cada conjunto de configuraciones, ambas opciones se dejarán por defecto dando clic en Next> y Execute respectivamente.

Por último, en la figura A.8 parte (a) se muestra todo el conjunto de configuraciones actualizadas y en la parte (b) se menciona la instalación completa.

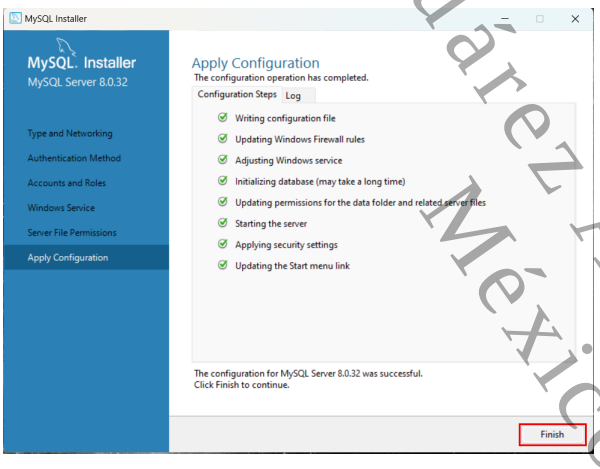



(a) Administrador servidor

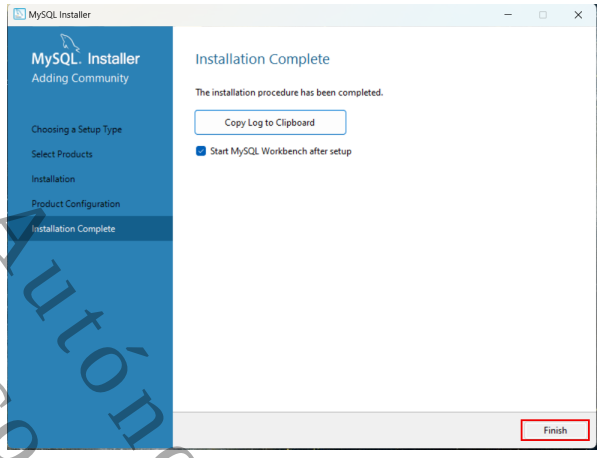



(b) Instalador de conjuntos

Figura A.7: Paso 9-10




(a) Instalación de conjuntos terminada




(b) Instalación completa

Figura A.8: Paso 10-11

## A.1. Entorno visual MySQL Workbench

MySQL Workbench proporciona una herramienta gráfica para trabajar con servidores y bases de datos MySQL, además cuenta con cinco áreas principales de funcionalidad, las cuales son:

- **SQL Development**(desarrollo SQL): Permite crear y administrar conexiones a servidores de bases de datos. Así como permitir configurar los parámetros de conexión, capacidad de ejecutar consultas SQL en las conexiones de la base de datos.
- **Data Modeling**(modelado de datos): Permite crear modelos del esquema de la base de datos de forma gráfica, permitiendo su total edición. Además el editor de tablas proporciona funciones fáciles de usar para editar tablas, columnas, índices, vistas, entre otras funcionalidades.
- **Server Administration**(administración del servidor): Permite crear y administrar instancias del servidor.
- **Data Migration**(migración de datos): Permite la migración de datos desde Microsoft SQL Server, Sybase ASE, SQLite, SQL Anywhere, PostreSQL y otras tablas, objetos y datos RBDMS a MySQL. Tambien se permite migrar versiones anteriores de MySQL.
- **MySQL Enterprise Support**(soporte empresarial MySQL): Permite compatibilidad con productos empresariales como MySQL Enterprice Backup y Auditoría MySQL.

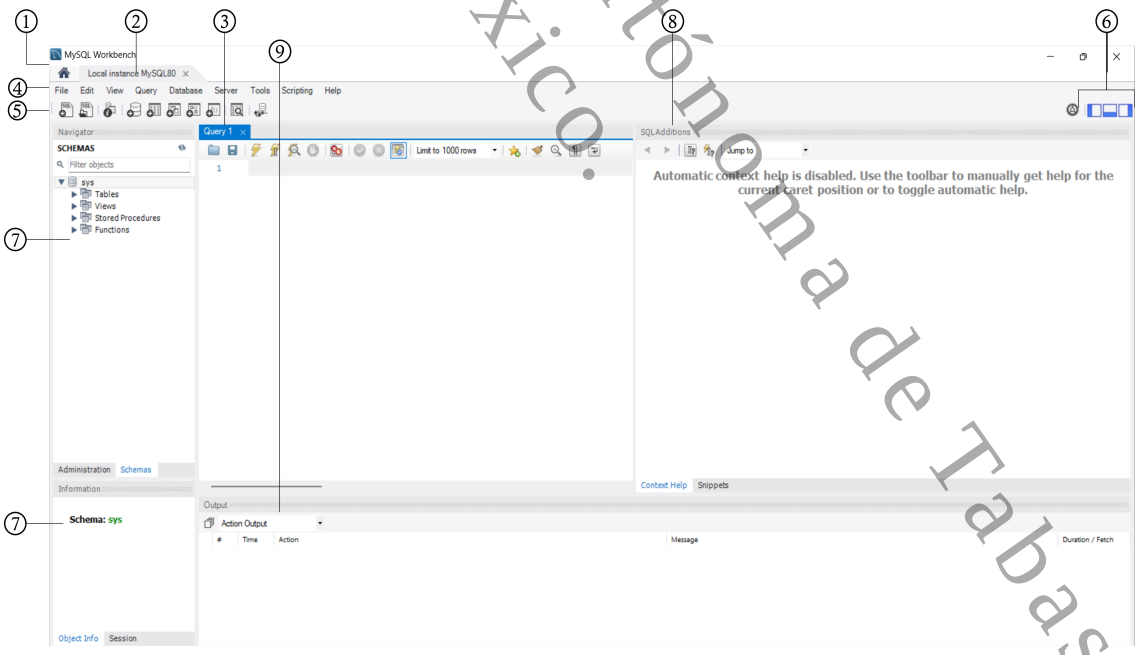



Figura A.9: Editor Visual SQL

### Descripción de los elementos del Editor Visual SQL

1. *Pestaña pantalla de inicio:* Proporciona acceso rápido a conexiones, modelos y el asistente de migración de MySQL. La pantalla de inicio no se cierra.

2. *Ficha conexión:* Cada conexión realizada al servidor MySQL es representado por una ficha de conexión independiente. Un servidor puede ser activo o inactivo cuando se abre la pestaña conexión.
3. *Ficha de consulta SQL*: Es una pestaña secundaria que se abre de forma predeterminada cuando se establece una conexión a un servidor MySQL. Cada ficha de consulta se identifica de forma única mediante un número que se incrementa.
4. *Barra de menú principal:* Contiene los siguientes menús: **Archivo**, **Editar**, **Ver**, **Consulta**, **Base de datos**, **Servidor**, **Herramientas**, **Secuencias de comandos** y **Ayuda**. Las acciones disponibles dependen de qué ficha se seleccione al hacer clic en un menú.
5. *Barra de herramientas principal*: Contiene acciones rápidas de herramientas, las cuales son:
   - Crear una nueva pestaña SQL para ejecutar consultas
   - Abrir un archivo de script SQL en una nueva pestaña de consulta
   - Abrir el inspector para el objeto seleccionado
   - Crear un nuevo esquema en el servidor conectado
   - Crear una nueva tabla en el esquema activo con el servidor conectado
   - Crear una nueva vista en el esquema activo en el servidor
   - Crear un nuevo procedimiento almacenado en el esquema activo de la carpeta servidor conectado
   - Crear una nueva función en el esquema activo en el cuadro servidor conectado
   - Buscar texto en los datos de la tabla en los objetos seleccionados
   - Volver a conectarse a DBMS
6. *Acciones de acceso directo:* proporciona los siguientes métodos abreviados:
   - Cuadro de diálogo Mostrar preferencias
   - Ocultar y mostrar el panel de la barra lateral
   - Ocultar y mostrar el panel del área de salida
   - Ocultar y mostrar el panel secundario de la barra lateral
7. *Panel de barra lateral:* Contiene dos etiquetas principales: Navegador e Información. El navegador tiene dos subpestañas: **Administración** y **Esquemas**. El área de Información proporciona el **Objeto subpestañas información** y **Sesión**, que incluyen información de solo lectura sobre un objeto seleccionado y sobre la conexión activa.
8. *Panel de barra lateral secundario (SQL Additions):* Proporciona las siguientes subfichas:
   - Ayuda contextual
   - Fragmentos de código

9. *Panel de área de salida:* El panel de salida puede mostrar un resumen de las consultas ejecutadas en las siguientes formas: salida de acción, salida de texto o salida de historial.

# Bibliografía


[1] A. S. Popelyukhin, "On hierarchy of actuarial objects: Data processing from the actuarial point of view," in *CAS Forum Winter*, pp. 219–237, 1999.

[2] C. M. Ricardo, *Bases de datos*. McGraw Hill, 2009.

[3] S. Abraham, K. H. F., S. Sudarshan, and P. F. SAENZ, *Fundamentos de bases de datos*. McGraw Hill, 1998.

[4] N. V. Sánchez, "Del soporte papel perforado y cinta magnética... al disco 3d holográfico anatómico-nanotecnológico: nuevos soportes magneto-ópticos y ópticos de almacenamiento masivo de información," in *Anales de documentación*, vol. 10, pp. 429–450, Facultad de Comunicación y Documentación y Servicio de Publicaciones de la ..., 2007.

[5] L. Vélez de Guevara, *Gestión de bases de datos*. Garceta, 2018.

[6] A. F. Cárdenas, *Sistemas de administración de bancos de datos*. México: Limusa„ 1983.

[7] G. KOCH, *Manual de referencia de Oracle*. McGraw-Hill, 1992.

[8] A. M. Rioja, *Base de datos: Diseño y gestión*. Sintesis, 2014.

[9] AMIS, "Developing a data warehouse for actuaries jan willem van der grijp," *Behavior Studies in Organizations*, 1994.

[10] CONAC, "Historia de la actuar´ıa en m´ exico," *Revista Mexicana de Investigación Actuarial Aplicada*, 2021.

[11] J. V. y Alan Angeles, *La profesi´ on actuarial. Una inducción para el estudiante.* Facultad de Ciencias, UNAM, 2017.

[12] M. Coronel, "Rob. bases de datos: Diseño, implementación y adminnistración."

[13] Á. Arias, *Bases de Datos con MySQL: 2ª Edición*. IT Campus Academy, 2014.

[14] A. Silberschatz, H. F. Korth, S. Sudarshan, F. S. Pérez, A. I. Santiago, and A. V. Sánchez, *Fundamentos de bases de datos*, vol. 11. McGraw-Hill Ciudad de México, México, 2002.

[15] G. W. Hansen, J. V. Hansen, and M. Katrib Mora, *Diseño y administración de bases de datos*. Prentice Hall, 1998.

[16] C. J. Date, *Introducción a los sistemas de bases de datos*. Pearson Educación, 2001.

[17] W. H. Inmon, *Building the data warehouse*. John wiley & sons, 2005.

[18] LinkedIn, "https://www.linkedin.com/pulse/empleos-en-auge-2023-de-linkedin-los-15-puestos-/?originalsubdomain=es, 18 de Enero de 2023," 2023.

[19] M. Loukides, *What is data science?* .º'Reilly Media, Inc.", 2011.

[20] C. Q. Ruiz, *Elementos de inferencia estadística*. Editorial Universidad de Costa Rica, 1993.

[21] J. García, J. Molina, A. Berlanga, M. Patricio, A. Bustamante, and W. Padilla, "Ciencia de datos," *Técnicas Analíticas y Aprendizaje Estadístico. Bogotá, Colombia. Publicaciones Altaria, SL*, 2018.

[22] S. M. Ross, *Introducción a la estadística*. Reverté, 2018.

[23] M. E. Rendón-Macías, M. Á. Villasís-Keeve, and M. G. Miranda-Novales, "Estadística descriptiva," *Revista Alergia México*, vol. 63, no. 4, pp. 397–407, 2016.

[24] C. Mehta, A. K. Bhavsar, H. Oza, and S. Shah, *MySQL 8 administrator's guide: effective guide to administering high-performance MySQL 8 solutions*. Packt Publishing Ltd, 2018.

[25] R. Elmasri, S. Navathe, R. Elmasri, and S. Navathe, "Fundamentals of database systems</title," in *Advances in Databases and Information Systems*, p. 139, Springer, 2015.