Felipe Ramírez

Modelación de Bases de Datos

y su implementación en SQL Estándar

APRENDA
EDICIONES

Modelación de Bases de Datos
y su implementación en SQL Estándar

La memoria de todo lo que pasa en la organización se almacena en bases de datos.

En este libro, el Dr. Felipe Ramírez, fiel a su estilo, te explica de forma minuciosa la forma correcta en que un modelo de datos almacena la evidencia de los fenómenos que ocurren en el mundo real, y cómo esos modelos de datos pueden servir para generar información para la toma de decisiones.

Después de leer este libro:

- Aprenderás de qué manera los datos describen la realidad.
- Aprenderás todos los conceptos de base de datos relacionales.
- Aprenderás a analizar una situación del mundo real, y modelarla.
- Aprenderás a pensar desde una perspectiva de inteligencia de negocios.
- Aprenderás a normalizar una base de datos.
- Aprenderás a elaborar diccionarios de datos.
- Aprenderás a elaborar diagramas de entidad relación (DER).
- Aprenderás a implementar modelos de datos usando SQL Estándar.
- Aprenderás a manipular y consultar datos usando SQL Estándar.

Este libro es de lectura obligada para: ***a)*** Estudiantes de tecnología, informática y sistemas, que cursen materias de base de datos; ***b)*** Profesionales que tengan la necesidad de modelar una base de datos, o interactuar con una; ***c)*** Usuarios de herramientas de inteligencia de negocios que requieren extraer información de bases de datos corporativas.

El **Dr. Felipe Ramírez** es profesor investigador en la Universidad Autónoma de Nuevo León, México. Es tecnólogo, abogado y doctor en administración; es autor de más de 13 libros de tecnología que han vendido miles de copias en América Latina y España; es especialista en bases de datos (***SQL Server, Oracle, MySQL, Access***), programación (***C#, vb.net, JavaScript, Python***), desarrollo web (***html5, css3, php, asp.net, bootstrap***), gestión de proyectos (***PMP, Project***), estadística e inteligencia de negocios (***Tableau, Power BI, Power Query, Excel***).

9 786079 597931

Modelación de Bases de Datos

y su implementación en SQL Estándar

Felipe Ramírez, Phd.

Facultad de Contaduría Pública y Administración

Universidad Autónoma de Nuevo León

APRENDA

EDICIONES

RAMÍREZ, Felipe
Modelación de Bases de Datos, y su implementación en SQL Estándar.
1ª Ed. México. Editorial: Aprenda Practicando, 2020.
ISBN: **978-607-95979-3-1**

Págs: 149 Formato: 17.5 x 22.75 cm.

APRENDA PRACTICANDO
SAN FELIX 5432-D, VISTA SOL,
GUADALUPE, NUEVO LEÓN,
MÉXICO.
AÑO DE EDICIÓN: 2020
1ª EDICION.
ISBN: 978-607-95979-3-1

Reservados todos los derechos. Ni la totalidad ni parte de esta publicación, así como los materiales complementarios que acompañan la obra, pueden reproducirse, registrarse o transmitirse, por un sistema de recuperación de información, en ninguna forma ni por ningún medio, sea electrónico, mecánico, fotoquímico, magnético o electroóptico, por fotocopia, grabación o cualquier otro conocido o por conocer, sin permiso previo y por escrito del titular de los derechos.

El software, productos y marcas utilizadas en este libro son propiedad intelectual de sus autores, y su uso queda sujeto a los términos del contrato licencia que aparece al instalar los mismos, así como de la legislación vigente.

El préstamo, alquiler o cualquier otra forma de cesión de uso de este ejemplar requerirá también la autorización del titular de los derechos o de sus representantes.

A Dalia.

Este libro es y siempre será

Gratuito

cortesía de:

APRENDA

creative commons

BY NC SA

El **Dr. Felipe Ramírez** es profesor investigador en la Facultad de Contaduría Pública y Administración, de la UANL, México. Es autor de más de 15 libros que han vendido miles de copias en américa latina y España.

Es instructor premium de diversos temas, por ejemplo, base de datos (SQL Server, MySQL, Oracle, Transact SQL, PL/SQL, APEX), programación (C#, JavaScript, Python, R), desarrollo Web (ASP.NET, MVC, PHP, HTML5, CSS3, BOOTSTRAP, LARAVEL, VUE.JS), software de productividad (Excel, Power Query, Power Pivot, Power BI, Microsoft Project), y modelos de gestión (ITSM, ITIL, SCRUM).

El contenido de este libro forma parte del curso "**Modelación de Base de Datos**", disponible en múltiples plataformas:

Udemy.com

AprendaEnLinea.mx

AprendaPracticando.com

INSTRUCTORES: Este material solo puede utilizarse sin fines de lucro.

Para fines comerciales, se puede adquirir por una cuota mínima en la modalidad de *Courseware*, y obtén otros beneficios: Trípticos, Mapas mentales, Presentaciones en Power Point, Guías de instalación de sala, Cuadernos de ejercicios, Plataforma en línea para exposición. info@aprenda.mx

Contenido

2 FENOMENOLOGÍA DE LOS DATOS 11

2.1 Los fenómenos 11
2.2 Los datos 12
2.3 La información 14
2.4 Las ideas y los conceptos 16
2.5 El pensamiento 17
2.6 Importancia de la fenomenología en la modelación 18
2.7 Analizando la naturaleza de los datos 20
i) Disponibilidad de los datos 21
ii) Tareas básicas de la información 22
iii) Tipos de datos 23
iv) Uso de los datos en inteligencia de negocios 24

3 CONCEPTOS DE BASE DE DATOS 25

3.1 Concepto de bases de datos 25
i) Es una colección 25
ii) Es auto descriptiva 25
iii) Está formada por registros integrados 26
3.2 Concepto de tablas, registros, campos 26
i) Tablas 26
ii) Campos 27
iii) Tipos de datos comunes 28
iv) Pasos para crear una tabla 32
v) Registros 33
EJERCICIO 03.01: Definición de tablas 34
Identificar de sujetos, eventos y estados 36
Identificar atributos de las tablas 36
Determinar tipos de datos y propiedades especiales 37
3.3 Concepto de llaves y relaciones 39
i) Llave primaria 39
ii) Llave foránea 40
iii) Relaciones entre tablas 41
iv) Concepto de dominio 41
EJERCICIO 3.2: LLaves y relaciones 42
Numeralia de la base de datos 43

4 NORMALIZACIÓN DE LA BASE DE DATOS 45

4.1 Vicios del modelo relacional 45
i) Redundancia 45
ii) Inconsistencia 46
iii) Falta de integridad 47
4.2 El proceso de normalización 48
i) Formas normales básica 49
ii) Caso práctico 49
iii) Dependencias funcionales 50
iv) 1NF (Primera Forma Normal) 51
v) 2NF (Segunda Forma Normal) 53
vi) 3NF (Tercera Forma Normal) 57
vii) BCNF (Forma Normal Boyce Codd) 58
viii) Desnormalización 58
EJERCICIO 4.1: Normalización 60
Normalizar hasta 1NF 61
Normalizar hasta 2NF 62
Normalizar hasta 3NF 66
Normalizar hasta BCNF 67

5 DOCUMENTACIÓN DE BASE DE DATOS 69

5.1 Diccionario de datos 69
i) Campos básicos 69
ii) Campos particulares 70
EJERCICIO 5.1: Diccionario de datos 71
Diccionario de datos 71
5.2 Diagrama de Entidad Relación 73
i) Representación de tablas 73
i) Representar las relaciones 74
ii) Representación de cardinalidad y opcionalidad 75
iii) Relaciones especiales 77
EJERCICIO 5.2: Diagrama de Entidad Relación 80
Distribución concéntrica preliminar 81
Representar las tablas 82
Representar las relaciones y la cardinalidad 82
Representar la opcionalidad 83

6 STRUCTURED QUERY LANGUAGE (SQL) 85

6.1 SQL Estándar 85
6.2 Trabajando con bases de datos 87
EJERCICIO 6.1: Creación de una base de datos 88

6.3 Trabajando con tablas 89
EJERCICIO 6.2: Codificación de Tablas 91
6.4 Restricciones de llave primaria (*primary key*) 93
EJERCICIO 6.3: Restricciones de llave primaria 94
6.5 Restricciones de llave foránea (*foreign key*) 96
EJERCICIO 6.4: Restricciones de llave foránea 97
6.6 Restricciones de unicidad (*unique*) 98
6.7 Restricciones de valor por omisión (*default*) 99
6.8 Restricciones de valor de auto incremento (*identity*) 100
6.9 Métodos abreviados de creación de objetos 101
EJERCICIO 6.5: Codificación abreviada 102

7 MANIPULACIÓN BÁSICA DE DATOS 105

7.1 Agregado de registros, usando INSERT 105
EJERCICIO 7.1: Cargando información a la base de datos 108
7.1 Expresiones lógicas 111
7.2 Modificando de registros, usando UPDATE 115
EJERCICIO 7.2: Modificando datos 116
7.3 Eliminando registros, usando DELETE 117
EJERCICIO 7.3: Modificando datos 118
7.4 Consultando información usando SELECT 119
EJERCICIO 7.4: Modificando datos 121

8 EJERCICIO INTEGRADOR 127

8.1 Entiende a tu cliente 127
8.2 Analiza los requerimientos 128
8.3 Enfócate en el fenómeno 131
8.4 La documentación como fuente 132
8.5 Obtén la información faltante 135
8.6 Identificación de sujetos, eventos y estados 137
8.7 Identificación de atributos 138
8.8 Determinación de tipos de datos 140
8.9 Normalización 140
8.10 Elaboración de Diccionario de datos 141
8.11 Elaboración del Diagrama de Entidad Relación 141
8.12 Codificación de la estructura de datos (DDL) 141
8.13 Datos de comprobación (DML). 142

9 ÍNDICE 145

1 INTRODUCCIÓN

Hola, cómo estás.

Yo soy Felipe Ramírez, profesor investigador en la Universidad Autónoma de Nuevo León, y autor de más de trece libros de tecnología que han vendido miles de copias en américa latina y España.

He desarrollado este libro porque pienso que hay una enorme necesidad de conocimiento teórico respecto a la modelación de base de datos, útil para todo aquél profesional que maneja información hoy en día. Tanto si eres un administrador de base de datos experimentado, como si eres un ingeniero o un contable haciendo consultas de datos en Google Sheets o Excel, este curso no tiene desperdicio: es breve, conceptual sin ser tedioso, y además es práctico.

La modelación de base de datos, hay que decirlo, es un tema antiguo que ya ha sido explicado muchas veces y de muchas maneras. Un gran número de personas podrían decir incluso que es un tema agotado.

La realidad es ésta: la modelación de base de datos no ha cambiado mucho en los últimos treinta años, es cierto, pero las personas que necesitan aprender modelación no son las mismas que eran antes; hace dos décadas los estudiosos del tema eran administradores de base de datos, programadores y arquitectos de software, a quienes se les explicaba el tema asumiendo una cierta preparación de especialidad tecnológica; ahora los estudiantes del tema incluyen a contadores, analistas, ingenieros, administradores, gente de recursos humanos, economistas y financieros, que ven las bases de datos desde una óptica de inteligencia de negocios.

Estos profesionales utilizan Tableau, Power BI y lenguaje R, y nadie asume que son "*gente de sistemas*". A esos usuarios nadie les ha hablado de modelación de bases de datos, pensando en las necesidades específicas de sus disciplinas. Este libro pretende precisamente eso.

El tema me apasiona, y tengo modelando datos desde hace muchos años, a mucha profundidad. Constituyó todo un reto para mí cambiar los conceptos clásicos relacionados al tema, y adecuarlos a personas que no tienen perfil técnico, sin perder rigurosidad. Los que ya han estudiado bases de datos se darán cuenta que no reproduzco las definiciones típicas, sino que construyo nuevas definiciones, ajustadas a la nueva dimensión y uso que se le da a la información.

En resumen, este libro explica temas viejos, vistos con ojos experimentados, con la finalidad de mostrar conceptos renovados a mentes interesadas en el almacenamiento y la recuperación de datos, con el fin de conocer hechos y estimar resultados.

Este libro está estructurado de la siguiente manera.

a) Primero, te explico qué es la fenomenología de los datos, y por qué es importante ver a las bases de datos como representaciones de la realidad.
b) Segundo, te explico todo el marco conceptual de las bases de datos relacionales.
c) Tercero, te explico cómo realizar un diccionario de datos, y cómo hacer un diagrama de entidad relación.
d) Cuarto, te explico las sentencias básicas para implementar tus bases de datos en lenguaje SQL estándar.
e) Quinto, te explico las sentencias básicas para manipular los datos de tu base, usando lenguaje SQL estándar,

Este pequeño libro es conocimiento puro. Ojalá lo disfrutes.

2 FENOMENOLOGÍA DE LOS DATOS

Este capítulo contiene los textos de una de mis conferencias más conocidas, titulada "*Fenomenología de los datos*"; en esa conferencia examino, desde un punto de vista filosófico, cuál es la razón de ser de toda recopilación de datos, en general, para luego explicar la naturaleza de los datos y la forma en que podemos aprovecharlos de una mejor manera.

Este capítulo es muy conceptual: analiza el por qué, y el para qué, de la recopilación de datos; si lo que te interesa es el *cómo*, puedes omitir el estudio de este capítulo sin que afecte tú aprendizaje de la modelación de datos.

2.1 Los fenómenos

Según la RAE, un *fenómeno* es «*algo que se hace presente en la conciencia de una persona a través de su percepción*»; en palabras de Immanuel Kant, un fenómeno es «*el objeto de la experiencia sensible*». Si juntamos ambos conceptos, podemos concluir que un fenómeno «*es todo aquello que percibimos de forma consciente a través de los sentidos*».

Toma en cuenta que los sentidos tienen una naturaleza fáctica, es decir, son receptores de lo que pasa en la realidad; aquello que pasa en la mente nada más, son conjeturas o pensamientos, y existen, pero no suceden. Podemos concluir que sólo lo que sucede en la realidad puede ser percibido por los sentidos; cuando decimos «vi en un sueño», realmente no es cierto, porque teníamos los ojos cerrados. Dada nuestra existencia en el tiempo y el espacio, la acción de percibir sucede de forma personal, en un lugar y momento dados, lo que da a los fenómenos un carácter de únicos e irrepetibles.

En ocasiones nuestros sentidos son tan limitados, nuestra memoria es tan corta y nuestra atención tan distraída, que nos vemos en la

necesidad de apoyarnos en instrumentos y herramientas con el fin de poder percibir los fenómenos, y recopilar evidencias de forma permanente. La mayoría de los fenómenos que ocurren en una organización, por ejemplo, una venta, el acceso de una persona a un área restringida, una orden de compra de materiales, deben terminar en bases de datos —almacenes de evidencias—, que son alimentadas a través de sistemas de información —instrumentos o herramientas—.

2.2 Los datos

Los seres humanos tendemos a describir lo que percibimos, y lo hacemos haciendo uso de signos o símbolos. La simple exposición a un fenómeno no sería utilizable para nuestro raciocinio si no pudiéramos explicarlo de alguna manera, con símbolos o palabras. Las personas *verbalizamos* las cosas, y las entendemos en la medida que las podemos explicar con signos y símbolos. Como diría Thomas Harris, «sólo podemos ver aquello que está en la mente».

Decía John Locke que las palabras son representaciones simbólicas de lo que percibimos, y su importancia radica en el significado que asociamos a ellas. Puedes comprobarlo por ti mismo, fácilmente: piensa en la palabra «éxito», por ejemplo; luego, piensa en lo que *tú* entiendes por éxito; luego, busca a otra persona, la primera que te encuentres, y pregúntale qué entiende por «éxito». Es muy probable que confirmes que hay discrepancias respecto a lo que cada quién entiende por «éxito». La palabra es la misma, pero al final, lo que importa es lo que cada uno entiende desde su punto de vista personal.

Dicho de otra manera, la cosa en sí, y la palabra con la cual la nombramos nunca serán lo mismo: nosotros aderezamos la cosa, agregándole o restándole significado, en función a nuestro conocimiento, experiencia y expectativa. Esto quiere decir que una persona nunca podrá ver las cosas de forma objetiva, porque no es un objeto; las personas vemos las cosas de manera subjetiva, porque somos sujetos, y entre la

percepción que tiene una persona y la que tiene otra, es muy posible que haya discrepancias.

Ponte a pensar en esto: si un niño, un astrónomo y un poeta ven pasar en el cielo una estrella fugaz, y se le pregunta a cada uno «¿qué fue lo que viste?» Cada uno de ellos describirá el fenómeno de diferente manera, y lo más interesante de todo es que ninguno de los tres estará equivocado respecto a lo que vio, porque lo vio desde sus capacidades y sus limitaciones.

Independientemente de los sesgos y las discrepancias posibles, las palabras son la forma más avanzada de representación de cosas que tiene el ser humano: alimentamos a los sistemas de información con palabras, con símbolos. ¿Qué nos hace pensar que los sesgos y discrepancias que tenemos respecto a las palabras y los símbolos no se trasladan a las bases de datos y los sistemas? En eso nos tenemos que poner a pensar un poco, para minimizar malentendidos.

Dado que los fenómenos son irrepetibles, la única manera de saber que ocurrieron es manteniendo una evidencia de que sucedieron. Un *dato* es la representación simbólica, ya sea mediante números o letras, de una característica cualitativa o cuantitativa que facilita la deducción de una conclusión o el registro de un fenómeno.

Entonces, los datos componen la evidencia de los fenómenos. Tienen un propósito revelador, pues nos permiten figurarnos algo que es o algo que sucedió —fenómeno—, hayamos estado expuestos o no a él. Ahora bien, ¿qué datos consideraría el poeta, el niño o el astrónomo, para describir el fenómeno de la estrella fugaz? Queda claro que la misma estrella fugaz no volverá a surcar el cielo, así que más valiera haber mirado lo que pasó con atención, en esa ocasión irrepetible. Si sólo el niño hubiera presenciado el fenómeno, y se lo explicara al astrónomo ¿crees que quedaría satisfecho con la explicación?

Parece que estoy hablando pura teoría, pero trata de analizarlo en una situación del mundo real: imagina que una persona va a una tienda de autoservicio, y compra un poco de carne de res. Cuando la paga, en la caja hay una báscula que pesa con precisión la cantidad de carne que el

cliente está comprando —instrumento—; la báscula mide el peso, y la operación se registra en los sistema de punto de venta (POS / Point Of Sale) de la tienda —memoria—; en el sistema queda registro de que un cliente compró esa cantidad de carne, ese día, a esa hora, en esa caja, atendida por ese cajero, y pagó una determinada cantidad de dinero específica —datos—. La operación sólo se registra si el hecho sucede, y sucede de forma única e irrepetible. Nunca más podrá suceder exactamente de la misma manera.

Pero ¿y qué pasa si el cajero se equivoca, y en lugar de registrar el código de la carne de res, registra el código de la carne de pollo? Los datos serán incorrectos, y la verdad de lo que pasó se habrá perdido para siempre. Al hacer un inventario, habría menos carne de res, y sobraría carne de pollo, y nadie sabría qué pasó.

Hay que entenderlo: en las organizaciones, los datos son la memoria de lo que pasó, y la oportunidad de registrar correctamente las operaciones es único e irrepetible. El registro de lo que pasó estará sesgado por las capacidades y limitaciones de las personas que viven el fenómeno, y si el registro es incorrecto, el pasado no se podrá reconstruir de manera confiable.

2.3 La información

Los datos son útiles cuando se tiene la cantidad mínima y suficiente y necesaria para representar un fenómeno.

Cuando un conjunto de datos tiene la cantidad y forma adecuada para aumentar el conocimiento o reducir la incertidumbre respecto a sujetos, eventos o estados, en una circunstancia particular, recibe el nombre de *información*.

Analicemos los conceptos de sujeto, evento y estado, porque son el fundamento de toda base de datos.

Los *sujetos* son aquellas entidades que hacen cosas, o a las que les suceden cosas. Los sujetos *son*, y, por tanto, es importante saber cuándo

son —y cuando no son—. Al recopilar datos de sujetos, debemos ser capaces de describirlos, de tal manera que conozcamos qué elementos son los que les dan su identidad —características / atributos—, estando así en posibilidades de juzgar si un sujeto *es*, o *no es*.

Ejemplos típicos de sujetos son: Clientes, Empleados, Compañías, Productos, Máquinas, etcétera. También se consideran sujetos aquellos *atributos clasificados*, donde el dato sólo puede tomar un valor predefinidos; aquí estamos hablando de los catálogos típicos, como por ejemplo colores, géneros, tamaños y así.

Los *eventos*, por otra parte, son aquellas cosas que el sujeto hace, o aquellas cosas que al sujeto le pasan; los eventos ocurren en un lugar y momento determinado. Los eventos *suceden*, y, por tanto, es importante saber si sucedieron o no, y en caso de que lo hayan hecho, dónde y cuándo. También es importante reconocer que están ligados a un sujeto, pues es quien lo hace o a quien le sucede. En ocasiones, es importante además conocer datos accesorios que le dan identidad al evento.

Ejemplo típico de un evento: La compra de un producto; queda claro que es realizado por un sujeto —cliente—, y que lo realiza en un lugar y momento determinado —lugar, fecha y hora de la compra—, y que tiene datos accesorios que le dan identidad al evento —cantidad de productos comprados, monto de lo comprado, etcétera—.

Finalmente, los *estados* son las situaciones en que se encuentra alguien o algo, en función a los diferentes modos en los que puede estar. Los estados *están*—aunque se escuche redundante—, y, por tanto, es importante saber qué elementos son los representativos de una situación, y la forma en que varían.

Al recopilar datos de estados, debemos ser capaces de indicar qué elementos representativos caracterizan a un estado, y el rango de valores que indican los diferentes grados o modos en que se puede estar. También son considerados estados aquellos datos que indican una relación de dependencia, por ejemplo, las tablas de relación.

Ejemplos típicos de un estado: Estado que guarda un pedido, temperatura ambiente, estado civil de una persona, tipo de cambio del

día, etcétera. En base de datos, también son estados las relaciones entre cosas, por ejemplo, qué actores actúan en qué películas, y así.

2.4 Las ideas y los conceptos

Se puede decir que cuando entendemos conscientemente a las palabras, generamos ideas.

Una *idea*, por definición, es el más obvio de los actos de entendimiento, pues se limita al simple conocimiento de algo. Las ideas son la manera particular y subjetiva en que entendemos una palabra y lo que representa. Cuando decimos «no tengo idea», queremos decir que ni siquiera nos hemos puesto a pensar que entendemos respecto a un concepto o situación en particular.

Nuestro entendimiento está determinado en función a la utilidad y uso que le damos a las palabras según nuestras propias necesidades. Las palabras son mitad de quien las expresa, y mitad de quien las interpreta: no siempre lo que se expresa es entendido de la manera que se esperaba.

Para la formación de ideas, la mente humana tiene capacidad de análisis y de síntesis.

El proceso mental de *análisis* nos permite aislar una palabra de otras, dándonos la capacidad de otorgarle el significado que desde nuestra propia opinión tiene, negándole en ese mismo momento a otras palabras la oportunidad de significar exactamente lo mismo.

Como decía John Locke, es difícil que dos personas, al escuchar una misma palabra, se formen en la mente la misma idea. El principio de una buena comunicación es la homologación de ideas asociados a las palabras.

Una *proposición* es la expresión de un juicio que puede ser considerado como verdadero o falso; generalmente, se trata de una afirmación o negación, inclusión o exclusión, donde intervienen un sujeto y un predicado, en donde se puede evaluar si el dicho es cierto o falso.

Ejemplo de una proposición podría ser: «El agua se congela a los cero grados centígrados».

Se puede decir que cuando utilizamos nuestras propias ideas —entendimiento propio de las cosas— para formular proposiciones, generamos *conceptos*, que por definición son opiniones o juicios que tenemos de las cosas.

Los seres humanos y nuestra forma de pensar no se limitan a la formación de ideas, sino al razonamiento a partir de ellas y la formulación de juicios e valor. Formarnos conceptos implica poner las ideas en acción y a prueba.

Para la formación de conceptos, el proceso mental de *síntesis* nos permite relacionar las ideas y formarnos juicios y opiniones respecto a ellas. Lo más importante de todo, es que, a partir de eso, podemos proponer estructuras sujetas a validación.

Las palabras, al ser entendidas conscientemente, nos permiten formarnos conceptos. Entonces, los conceptos son la manera particular y subjetiva en que entendemos una palabra y lo que representa. Nuestro entendimiento está determinado en función a la utilidad y uso que le damos a las palabras según nuestras propias necesidades.

2.5 El pensamiento

Cuando somos capaces de combinar los conceptos que tenemos en la mente con la finalidad de imaginar, considerar, inventar o sostener nuevos conceptos, entonces estamos teniendo *pensamientos*.

Los pensamientos implican inferencia, pues a partir de los conceptos existentes se desarrollar y fundamentan nuevos conceptos —proposiciones / juicios—, lo cual enriquece el conocimiento en general.

Mucho se especula que a la gente «no le gusta pensar». La verdad de las cosas es que *pensar* implica un camino arduo.

Empecemos poniendo atención a lo que pasa a nuestro alrededor y percibirlo a través de nuestros sentidos; en caso de que eso no sea posible, debemos confiar en que existen datos que representan de manera confiable los fenómenos que queremos conocer, y que los datos están en la cantidad y la calidad adecuada como para que nosotros seamos capaces de explicarnos a nosotros mismos en palabras el fenómeno, de tal manera que lo concibamos de acuerdo a nuestro propio contexto, que nos formemos ideas acerca de él, y luego juicios de valor que culminen en un ejercicio de imaginación que nos permita ir más allá del fenómeno original, ampliando nuestro conocimiento.

Como podrás darte cuenta, en el camino del pensamiento hay muchas dificultades: Puedes no haber experimentado un fenómeno directamente, puedes disponer de datos poco confiables o equivocados, o enfrentar la ausencia de datos, de plano; puede que el fenómeno esté ahí, pero que no tengas las palabras exactas para explicar el fenómeno, y no seas capaz de formarte una idea, o bien te formas una idea errónea que sólo servirá para la formulación de juicios erróneos que te conduzcan a pensamientos inexactos.

Pensar requiere rigurosidad.

2.6 Importancia de la fenomenología en la modelación

La importancia de la fenomenología de los datos se puede ilustrar a través de un caso hipotético.

Imagina que es día de elecciones para presidente en un país, y una persona te contrata para obtener cierta información que requiere saber, con el fin de poder tomar una decisión importante.

Al principio del día, te dice que necesita tener la información necesaria para poder calcular, al final del día, la proporción de hombres y proporción de mujeres que votaron.

La cosa es fácil: te ubicas en una posición donde puedas ver a los votantes, y en una hoja electrónica pones una columna que diga «GÉNERO». Si la persona que vota es mujer, registras una «M»; si es hombre, registras una «H».

Al final del día, cuentas cuántos registros tienes, lo que da el «TOTAL DE VOTOS»; luego cuentas los registros tienen el valor «M» y tienes el «TOTAL MUJERES»; luego cuentas los registros tienen el valor «H» y tienes el «TOTAL HOMBRES»; si divides TOTAL HOMBRES entre TOTAL DE VOTOS, tendrás la «PROPORCIÓN DE HOMBRES», y si divides TOTAL MUJERES entre TOTAL DE VOTOS, tendrás la «PROPORCIÓN DE MUJERES».

Así lo haces a la perfección.

Al final del día, tu cliente llega, y le muestras lo que obtuviste. El cliente te sorprende con lo siguiente: pero ¿cuántas de esas mujeres tenían edad de entre 18 y 30, 31 y 40, 41 y 50, y más de 50? ¿Y cuántas de esas mujeres vestían pantalones? ¿Y a qué horas votaron más hombres que mujeres?

No tienes respuesta a eso. Lo peor del asunto es que los fenómenos que mediste fueron únicos e irrepetibles, y la única manera en que pudieras tener la nueva información que se te solicita es reuniendo de nuevo a todos los votantes, haciendo que vistieran de la misma manera, que votaran a la misma hora, y que les preguntaras la edad. En otras palabras, es imposible.

Todos sabemos que recopilar información cuesta, así que las organizaciones sólo recopilan la menor cantidad de datos posibles, siempre y cuando permitan obtener la información que necesitan.

Es muy común que los sistemas se elaboren en un momento determinado, y que el cliente de la información en ese momento sólo requiera ciertas cosas (como el *género*, en nuestro ejemplo). Las bases de datos se construyen de manera que sus necesidades se resuelvan, e incluso los sistemas se desarrollan con el objetivo de resolver esas necesidades específicas de las personas empoderadas.

Si se presenta la situación donde cambia el cliente de la información, y el nuevo cliente requiere nuevos datos que el primer cliente no había requerido (*edad, vestimenta, hora*, en nuestro ejemplo), así que se debe modificar la base de datos y las aplicaciones, y es cuento de nunca acabar. Lo malo es que será imposible tener información histórica de los nuevos datos, simple y sencillamente porque el fenómeno ya pasó, es irrepetible, y no hay manera de reproducirlo.

Los diseñadores de base de datos *deben orientarse al fenómeno, no al cliente*; deben asegurarse de que el modelo de datos y los sistemas sean capaces de registrar el fenómeno con la mayor fidelidad posible, entendiendo que, si no se hace así, será imposible reproducirlos en el futuro y tendremos bancos de datos incompletos.

La ventaja de usar sistemas de clase mundial, como **SAP Business One**, **Microsoft Dynamics**, **Oracle Netsuite**, es que recopilan información con independencia del cliente, lo que, en cierta manera, les sirve a muchos tipos de cliente: están orientados al fenómeno, no al cliente.

Los modelos de base de datos deben contener las estructuras que nos permitan almacenar sujetos, eventos y estados, y que nos permitan disponer de información a través de la cual pongamos nuestra experiencia (ideas, conceptos y pensamientos) a trabajar hacia un fin específico.

2.7 Analizando la naturaleza de los datos

No sólo debemos entender a los datos como una evidencia de los fenómenos, sino también como el conducto para disponer de información que nos permita tener ideas, formar conceptos, y tener pensamientos. Para ello, es recomendable entender la naturaleza de los datos desde diferentes ópticas.

i) Disponibilidad de los datos

Partamos del supuesto que necesitamos cierta información, y para poder disponer de ella, requerimos datos. La pregunta es, ¿los tenemos? Si analizamos el estado que guarda la disponibilidad de los datos, podemos saber qué necesitamos hacer para hacernos de datos, en caso de que no los tengamos.

En cuanto a su disponibilidad, tenemos los siguientes estados.

a) **Se tienen.** Es cuando los datos existen, tenemos privilegios para usarlos y están disponibles para su procesamiento.
b) **Se calculan.** Es cuando los datos no existen, pero pueden calcularse a partir de otros datos existentes y disponibles.
    a. Bajo este supuesto, debemos tener acceso a los datos que utilizaremos para calcular nuevos datos.
    b. Bajo este supuesto, debemos tener especificadas las fórmulas o procesamientos que nos llevarán a los datos calculados.
    c. Bajo este supuesto, aplican algunas reglas de los algoritmos, como son ser precisos (los pasos siempre son los mismos), definidos (los mismos datos de entrada generan siempre las mismas salidas), y finito (el procesamiento tiene un fin).

c) **Se estiman.** Es cuando los datos no existen, pero existen otros datos semejantes y anteriores, a partir de los cuales se puede hacer una estimación con cierto grado de confianza.
    a. A este supuesto le aplican las mismas reglas que al cálculo de datos, a diferencia que no contamos con datos base, y se tienen que suponer.
    b. Bajo este supuesto, los cálculos se apoyan en técnicas estadísticas de cálculo de estimaciones.

d) **Se preguntan.** Es cuando los datos no existen, y tienen que preguntarse para ser obtenidos, ya que no pueden ser ni estimados ni calculados.

ii) Tareas básicas de la información

La responsabilidad que tenemos respecto a la información determina el acceso y los permisos que debemos tener, así como las habilidades que debemos dominar.

Relacionado con las tareas a desarrollar con la información, tenemos tres variantes.

a) **Se obtiene.** Es cuando tenemos la responsabilidad de recopilarla desde las fuentes en donde se encuentra, y en caso de no existir, coordinar su captura o adquisición de alguna manera.
    a. Si lo que vas a hacer es obtener información, debes tener claros los parámetros de calidad de la información: la cantidad requerida, la forma, la exactitud, la consistencia.
    b. Debes saber discriminar buenas fuentes y malas fuentes de información, y reconocer las reglas de comunicación y privacidad que apliquen.

b) **Se transforma.** Es cuando tenemos la responsabilidad de procesar los datos existentes con el fin de ordenarlos, filtrarlos, cambiarles la forma o el nivel de detalle, así como calcular y estimar los datos no existentes.
    a. Si lo que vas a hacer es transformar la información, debes tener claros los parámetros de ordenamiento y filtrado que debes aplicar.
    b. Debes saber aplicar cálculos agregados (sumarización, promedio, máximos, mínimos, desviaciones estándar, etcétera).
    c. Debes pensar algorítmicamente respecto a los cálculos, y conocer las salidas que debes obtener ante determinadas entradas.

c) **Se divulga.** Es cuando tenemos la responsabilidad de hacerla llegar oportunamente y en el formato más

conveniente a las personas indicadas y autorizadas para recibirlas.

a. Si lo que vas a hacer es divulgar información, requieres conocer a la perfección los criterios de formato, autorización y periodicidad de las entregas de información.
b. Preferentemente, debes tener habilidades de manejo de información: dominio de analítica de datos y habilidades de ilustración gráfica.
c. En el caso de la divulgación, debe tenerse algo de conocimientos de formato de medias: conexiones a base de datos, archivos CSV, datos serializados XML / JSON, etcétera.

iii) Tipos de datos

Los *tipos de datos* determinan las operaciones que se pueden realizar con los datos, y también determinan la precisión de algunas transformaciones que pueden realizarse con ellos.

Tipo	Sub-clase	Descripción
Etiqueta	Nominales	Datos textuales descriptivos.
	Ordinales	Datos textales que implican un orden.
Numéricos	Escala de intervalo	Datos numéricos que no tienen un cero absoluto.
	Escala de razón	Datos numéricos que tienen un cero absoluto.
Lógico / booleano	Booleano	Dato lógico que significa un dato binario mutuamente excluyente. Verdadero / Falso, Sí / No.
Fecha	Fecha / Serie	Dato numérico equivalente a una fecha de calendario.

La tabla anterior enumera los diferentes tipos de datos que puedes encontrarte en un ambiente de negocios. Como analista, te recomiendo

que trates de saber en todo momento qué tipo de datos son los que estás manejando.

iv) Uso de los datos en inteligencia de negocios

La *inteligencia de negocios* o *business intelligence* (*BI*) es el conjunto de procesos que facilitan la extracción, transformación y carga de datos contenidos en los sistemas de gestión empresarial y otras fuentes confiables, con el fin de disponer de repositorios utilizados por herramientas avanzadas de ordenamiento, filtrado, cómputo y presentación de información, que apoyan la toma de decisiones operativas y estratégicas.

La inteligencia de negocios ha revolucionado las estructuras de datos: si antes importaba cuidar el almacenamiento y la recuperación, ahora es importante también el uso que se le dará a los datos.

Uso	Tipo sugerido / Descripción	Ejemplo
Identificación	Etiqueta o número que permite identificar los registros como únicos, dentro de una muestra.	Número de empleado. Código de producto.
Categoría / Segmento	Etiqueta que permite segmentar o categorizar los datos.	Departamento. Línea de producto.
Descripción	Etiqueta que permite describir los datos.	Nombre de producto. Razón social del cliente.
Temporalidad	Fecha/hora que permite analizar la temporalidad de los eventos.	Fecha de venta. Hora de conclusión.
Estado	Etiqueta o número que permite representar un estado dentro del registro.	Activo / Inactivo. Pendiente/Iniciado/Concluido
Valor	Número que permite representar una cantidad cuantificable.	Monto de venta. Cantidad requerida.

3 CONCEPTOS DE BASE DE DATOS

3.1 Concepto de bases de datos

David Kroenke define *base de datos* como "una la colección auto descriptiva de registros integrados".

i) Es una colección

Una base de datos es una *colección* porque se compone de un conjunto de datos. Generalmente, debemos estar hablando de una cantidad significativa de información procesable, bien estructurada y de difícil análisis contemplativo.

ii) Es auto descriptiva

Una base de datos es *autodescriptiva* porque dentro de sí misma contiene información que describe la manera en que los datos se organizan y almacenan dentro de la base.

Se le conoce como *estructura* de la base de datos, al conjunto de información que describe la definición de tablas, campos, tipos de datos, llaves, relaciones y restricciones.

Además de la estructura, la base de datos contiene información de *ambientación*, que contiene información de objetos especiales, usuarios, permisos, alojamiento físico de archivos, entre otras cosas.

A la información de estructura y ambientación contenida en una base de datos, se le conoce como *metadatos*.

Este rasgo es muy importante para distinguir una base de datos de un archivo de datos. Por ejemplo, un archivo CSV que contiene información de clientes, no es una base de datos; aunque se trate de una colección de datos, no es información autodescriptiva.

iii) Está formada por registros integrados

Antes de hablar de registros, debemos hablar de los datos.

Los *datos* son expresiones literales significativas, de naturaleza atómica y con dominio propio, asociadas a una característica de un sujeto, evento o estado.

En una base de datos, el contenido se divide en *registros* de diferente naturaleza, que se interrelacionan entre sí para ganar significado.

Una base de datos es integrada, porque los diferentes registros que contiene se integran para generar información. A lo mejor tener una relación de clientes es valioso, pero cuando esa información se relaciona con la información de pedidos, con la información de cobranza, con la información de mercadotecnia, pues gana significado.

3.2 Concepto de tablas, registros, campos

i) Tablas

Una base de datos se compone de tablas.

Las *tablas*, llamadas también *entidades* o *tuplas*, son datos almacenados en forma de arreglo bidimensional, que almacenan información de una misma naturaleza.

En ese arreglo bidimensional de datos, las *columnas* representan *campos* o *atributos*, mientras que las *filas* representan *registros*.

Algunos teóricos definen a una tabla como un conjunto de registros, pero yo no comparto esa definición, dado que una tabla puede existir aún y cuando no tenga registros; yo más bien pienso que una tabla es un conjunto de campos, porque no puede existir una tabla que no tenga campos, se tengan registros o no.

¿Qué almacena una tabla? Pues bien, una tabla puede mantener el registro de tres cosas.

En primer lugar, una tabla puede contener información de *Sujetos*, es decir, quien ejecuta alguna acción, o a quien le sucede algo. Ejemplos de este tipo de información son clientes, proveedores, proveedores, departamentos, artículos, etcétera.

También se consideran sujetos aquellos *atributos clasificados*, es decir, atributos de valor único que son referidos a través de una clave, como por ejemplo color, tamaño, estilo, y así, a lo que se conoce también como catálogos.

En segundo lugar, una tabla puede contener información de *Eventos*, es decir, todo aquello que hace un sujeto, o que le sucede a un sujeto. Ejemplos de este tipo de tablas son facturas, movimientos de inventario, asistencia laboral, y así. Este tipo de tablas se reconoce fácilmente porque generalmente tienen datos que indican secuencia o momento de realización.

En tercer lugar, una tabla puede contener información de *Estados*, es decir, todo aquello que no es ni sujeto, ni evento, pero que es requerido como referencia. Un ejemplo de este tipo de tabla puede ser aquella que almacena información de tipos de cambio del día, tablas de factores de conversión, y así.

Una tabla sólo almacena información de una misma naturaleza única, sin ambigüedades.

ii) Campos

Los *campos*, conocidos también como *columnas* o *atributos*, son características particulares de un sujeto, evento o estado, que son representados a través de un dato.

Cada campo posee un identificador único, un dominio, un tipo de dato, y una longitud, que puede ser fija o variable.

Además de sus características fundamentales, los campos pueden tener restricciones, siendo las principales las de unicidad y de nulidad.

La restricción de *unicidad* indica que el dato no puede repetir valores dentro de una colección, y la restricción de *nulidad* implica que puede desconocerse el valor que tiene, es decir, es un dato nulo.

Un error que se comete con frecuencia es decir "tiene valor nulo": recuerda que «nulo» no es un valor; de hecho, cuando un dato es nulo, quiere decir que se desconoce su valor.

iii) Tipos de datos comunes

Los *tipos de datos* son dominios nominados, entendiendo por *dominio* el conjunto de valores válidos para un dato.

De esa manera, al decir `int`, por ejemplo, representamos un dato numérico, entero, consecutivo, con signo, que tiene una escala determinada. Las escalas dependen de la herramienta con la que se esté trabajando la base de datos, y debe ser verificada.

Los tipos de datos podemos dividirlos en 4 categorías: Numéricos, Fecha, Cadena o String, y Misceláneos.

En este curso sólo enumero lo más común, pero no te limites, ni te confíes. Es muy importante que una vez que decidas qué herramienta de base de datos utilizarás para trabajar tus datos, confirmes si soporta o no un determinado tipo de datos, y cuáles son las escalas de valores, pues puede haber variantes.

Tipos de datos numéricos

Primero están los *tipos de datos numéricos*, que almacenan valores numéricos con los cuales se pueden hacer operaciones aritméticas. Se dividen en enteros y decimales.

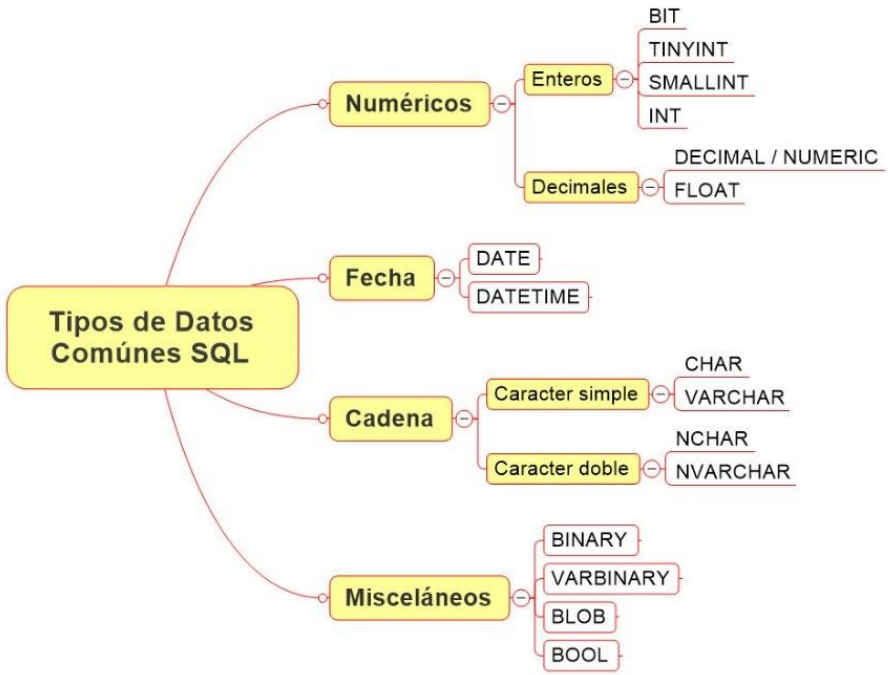


FIGURA 3.1: *Tipos de datos comunes en SQL.*

Los *tipos de datos numéricos enteros* más comunes son:

a) **BIT:** Es un dato numérico que sólo puede almacenar 0 o 1.
b) **BYTE/TINYINT:** Que son números enteros, con una escala de 0 a 255.
c) **SMALLINT:** Son números enteros, con una escala de -32,768 a 32,767.
d) **INT:** Son números enteros, con una escala de -2,147,483,648 a 2,147,483,647.

Toma en cuenta que, en general, entre más amplia la escala, mayor consumo de espacio físico en la base de datos.

Por otra parte, los *tipos de datos numéricos decimales* más comunes son:

a) **DECIMAL**, o su alias, **NUMERIC:** Que son números con precisión y escala, de naturaleza exacta, con rango de -10^38 +1 a 10^38 -1; este tipo es generalmente utilizado para cantidades monetarias.
b) **FLOAT:** Número con precisión y escala, de naturaleza aproximada, con rango de -1.79E + 308 a +1.79E + 308. Que sea de naturaleza aproximada quiere decir que puede llegar a redondear de forma poco precisa, y no se recomienda para cálculos que requieren precisión.

En este tipo de datos, cuando decimos que tienen *precisión* y *escala*, nos referimos a la manera en que se estructuran en relación con el punto flotante. Por ejemplo, el número 4,529.732 tiene una precisión de 7, porque está compuesto por 7 dígitos en total; por otro lado, su escala es de 3, porque son los dígitos que se encuentran a la derecha del punto decimal.

Una regla no escrita para datos monetarios, es usar `DECIMAL(19,4)`, por dos razones: porque el número de decimales permite buena exactitud, y porque hay varios países en, ciertas áreas, donde es obligatorio realizar cálculos a 4 posiciones decimales.

Tipos de datos de fecha

Los *tipos de datos de fecha* sirven para representar fechas y horas. Generalmente, se trata de un dato numérico que representa una fecha y hora equivalente; la parte entera del dato numérico representa la fecha, mientras que la parte decimal del dato numérico representa la hora.

Los tipos de dato de fecha más comunes son:

a) **DATE:** Son los que representan únicamente la fecha, en términos de año, mes y día.

b) **DATETIME:** Son los que representan la fecha, más la hora, expresada en horas, minutos y segundos.

Tipos de datos de cadena, o string

Los *datos de tipo cadena* o *string*, son aquellos que nos permiten almacenar información textual.

En cuanto a los caracteres disponibles, hay de dos tipos: *tipos de caracter simple*, o *ASCII*, en donde se manejan caracteres de 1 byte, y donde debemos definir una tabla de caracteres de 255 símbolos, llamada *char code*, que determina los caracteres de idiomas específicos; cuando el texto a manejar utiliza caracteres de un solo idioma o lenguaje escrito, con un solo juego de signos ortográficos, los tipos de carácter simple son más que suficientes; por otro lado, están los *tipos de caracter doble*, o *UNICODE*, que almacenan caracteres de dos bytes, es decir, que al tener 65,536 posibilidades, pueden almacenar símbolos de cualquier idioma, al menos en teoría. Si esperas almacenar información en caracteres árabes, griegos, tailandeses, español, checo y así, lo adecuado es que uses tipos de carácter doble.

En cuanto al número de caracteres máximo para una cadena, a lo que se conoce como *longitud*, tenemos dos tipos: *tipos de cadena de ancho fijo*, que son los tipos que miden siempre lo mismo, y siempre ocupan el mismo espacio físico; este tipo de datos se usa cuando todas las posiciones, o la mayoría de estas, serán ocupadas, como podría ser un ISBN, o un número de seguridad social; el consumo en bytes de este tipo de datos es igual a la longitud definida para el dato.

Por otro lado, están los *tipos de cadena de ancho variable*, llamados también *VAR types*, que son aquellos que miden solamente los caracteres que ocupa el dato; estos tipos de datos son muy usados cuando los datos pueden ser muy variables en su extensión, como podría ser un nombre completo, una descripción de un producto, o un correo electrónico. La ventaja de este tipo de datos es que ocupa menos espacio físico para almacenarse, pues solo usa lo que se ocupa. El consumo en bytes de este tipo de datos es igual a la longitud del dato almacenado, + 2 bytes,

requeridos para el manejo de inicio y fin de la cadena; esto nos lleva a pensar que, si sacamos la desviación estándar del ancho de los datos por almacenar, y da un número superior a 2, entonces es conveniente usar este tipo de datos.

Los tipos de datos de cadena más comunes son:

a) **CHAR:** Son tipos de carácter simple y ancho fijo.
b) **VARCHAR:** Son tipos de carácter simple y ancho variable.
c) **NCHAR:** Son tipos de carácter doble y ancho fijo.
d) **NVARCHAR:** Son tipos de carácter doble y ancho variable.

Tipos de datos misceláneos

Son datos que no pueden considerarse adecuadamente en otras categorías, por la forma en que funcionan. Los más comunes son los siguientes.

a) **BINARY:** Almacenan información binaria, ocupando un espacio determinado y fijo.
b) **VARBINARY:** Almacena información binaria, ocupando un espacio variable.
c) **BLOB:** Es el acrónimo de *Binary Large Objects*; similares a los datos **BINARY**, almacenan objetos binarios de gran tamaño, como podrían ser videos, música, y así.
d) **BOOL:** Almacenan valores de tipo falso o verdadero. Es muy común que este tipo de dato realmente sea un alias de un dato bit, que almacena 0 o 1, donde 0 es falso, y 1 verdadero.

iv) Pasos para crear una tabla

Para crear una tabla te recomiendo los siguientes pasos:

a) **Primero:** Analiza una situación del mundo real, y mediante un proceso intelectual de análisis, identifica los sujetos, eventos y estados.

b) **Segundo:** Para cada sujeto, evento o estado que identifiques, genera una tabla.
c) **Tercero:** Nombra la tabla en minúscula, usando *snake_notation*, y en singular.
d) **Cuarto:** Las tablas estarán formadas por un conjunto de campos: cada campo será una característica asociada con el sujeto, evento o estado.
e) **Quinto:** Identifica, para cada campo, el tipo de dato más adecuado para almacenar la información.
    a. En el caso de los datos de tipo cadena, asigna el tipo de dato adecuado, dependiendo de los símbolos a utilizar, y la variabilidad del ancho de los datos.
    b. En el caso de números que son utilizados como identificador, como por ejemplo el número de producto, o número de aula, es claro que no vas a realizar operaciones aritméticas con ellos, así que prefiere manejarlos como datos de tipo cadena.
    c. Si utilizas un dato numérico decimal, considera siempre la precisión y la escala adecuadas a lo que vas a almacenar.

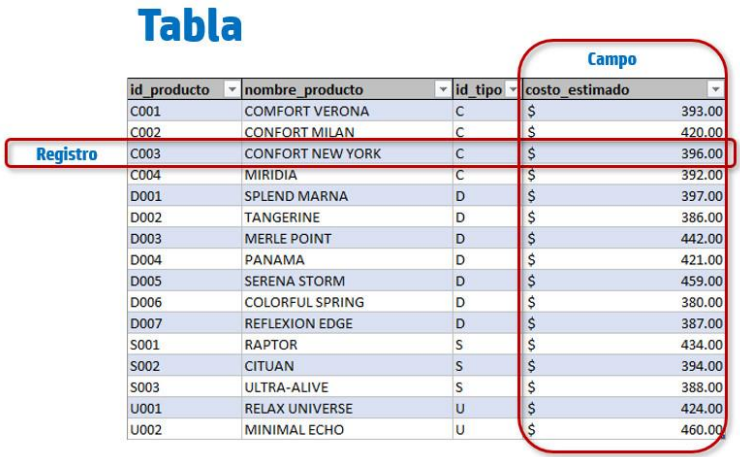
f) **Sexto:** El nombre de los campos debe ser significativo.

Las reglas sugeridas aquí no son obligatorias. Es posible que los convencionalismos existentes en ciertas organizaciones sean diferentes, y es válido.

v) Registros

Los *registros*, llamados también *filas*, son el conjunto de datos que representan el contenido de los campos de la tabla. Si la tabla tiene 5 campos, entonces un registro es un conjunto de datos correspondientes a dichos campos.

Obviamente, una tabla puede contener n registros, y todos ellos tendrán el mismo número de campos, cada uno de ellos conteniendo la misma naturaleza de información, registro a registro.



id_producto	nombre_producto	id_tipo	costo_estimado
C001	COMFORT VERONA	C	$ 393.00
C002	CONFORT MILAN	C	$ 420.00
C003	CONFORT NEW YORK	C	$ 396.00
C004	MIRIDIA	C	$ 392.00
D001	SPLEND MARNA	D	$ 397.00
D002	TANGERINE	D	$ 386.00
D003	MERLE POINT	D	$ 442.00
D004	PANAMA	D	$ 421.00
D005	SERENA STORM	D	$ 459.00
D006	COLORFUL SPRING	D	$ 380.00
D007	REFLEXION EDGE	D	$ 387.00
S001	RAPTOR	S	$ 434.00
S002	CITUAN	S	$ 394.00
S003	ULTRA-ALIVE	S	$ 388.00
U001	RELAX UNIVERSE	U	$ 424.00
U002	MINIMAL ECHO	U	$ 460.00

FIGURA 3.2: *La tabla y sus elementos.*

EJERCICIO 03.01: DEFINICIÓN DE TABLAS

Una escuela desea almacenar en una base de datos la información de los alumnos y los cursos extracurriculares que están tomando.

De los alumnos se conoce la matrícula, que es un número de 6 posiciones; el nombre completo, que nunca excede de 100 posiciones; y su fecha de nacimiento.

Aunque todos los alumnos deben inscribirse al menos a un curso extracurricular, no todos lo han hecho; esto se debe a que la escuela les da un plazo para que lo hagan, y el plazo no se ha vencido todavía.

Toma en cuenta que a escuela tiene un programa multicultural, y aunque la mayoría de los nombres pueden ser escritos con símbolos de escritura occidental, hay alumnos de Europa del este cuyo nombre contiene caracteres no occidentales.

De los cursos, se sabe el código del curso, que es un código alfanumérico de 5 posiciones; el nombre del curso, que en ninguno de los casos excede los 100 caracteres de longitud; el precio del curso, expresado en dólares americanos; y la asistencia máxima permitida para el curso, que en ningún momento puede ser mayor a 80 personas.

Los cursos ofrecidos forman parte de un convenio académico que se tiene con universidades de la Unión Europea, y tienen un código de identificación dentro de dicho convenio, que es un código alfanumérico de 10 posiciones. Este código no se puede repetir, y bien podría tratarse de una llave candidata para cursos.

Los cursos pueden tener mucha demanda o poca, y se ha dado el caso de que se programan cursos a los cuales nadie se inscribe.

Cuando un alumno se inscribe, debe quedar almacenada la siguiente información, por cuestiones de control: folio de inscripción, que es un número consecutivo; matrícula del alumno inscrito; código del curso al que se inscribió; y la fecha y hora de inscripción.

Es importante mencionar que un alumno no se puede inscribir dos veces al mismo curso, y que un alumno puede inscribirse a tantos cursos como quiera.

i. Identifica los sujetos, eventos o estados que encuentras en el ejercicio. Enuméralos.

ii. Identifica, para cada sujeto, evento o estado, las características o atributos que los componen. Cada característica, será un campo. Enuméralos.

iii. Determina, para cada campo, el tipo de dato más adecuado. En el caso de que utilices campos de tipo cadena, no olvides indicar la longitud máxima; en caso de que utilices campos numéricos con decimales, no olvides determinar la precisión y escala más adecuadas.

Identificar de sujetos, eventos y estados

Analizando el caso, se pueden identificar los siguientes sujetos; recuerda que los sujetos son quienes hacen algo, o a quienes les sucede algo: tenemos que la escuela quiere hacer el registro, los alumnos que pueden inscribirse a cursos, y los cursos en los cuales se pueden inscribir los alumnos. Debido a que las escuelas son solo el contexto, no necesitamos modelar información relacionada a escuelas, de hecho, ni siquiera tenemos información descrita de las escuelas.

Ahora identifiquemos los eventos del caso. Podemos ver que donde coinciden los alumnos y los cursos, es en la inscripción; siendo así, la inscripción es un evento.

Las tablas de estado son difíciles de encontrar si no estamos en un escenario de desarrollo de aplicaciones, así que no hay estados.

En resumen, el modelo de datos tendría las siguientes tablas: `alumno`, `curso`, `inscripcion`. Observa cómo están en minúsculas, en singular, y deliberadamente, omito el uso de acentos.

Identificar atributos de las tablas

Ahora, identifiquemos, para cada una de las tablas, sus atributos.

Según la descripción, del alumno sabemos su matrícula, su nombre completo y su fecha de nacimiento. Aunque la descripción no lo dice, la experiencia nos enseña que es muy práctico dividir el nombre completo en primer apellido, segundo apellido y nombre.

`alumno` = `matricula` + `primer_apellido` + `segundo_apellido` + `nombre` + `fecha_nacimiento`.

Del curso sabemos el código, el nombre del curso, el precio, el número máximo de asistentes, y el código equivalente para el convenio académico de la Unión Europea.

`curso` = `codigo_curso` + `nombre_curso` + `precio` + `maximo_asistentes` + `id_ue`.

De las inscripciones, sabemos el folio, la matrícula del alumno que se inscribe, la clave del curso al cual se inscribe, y la fecha y hora de inscripción. Aunque la descripción no lo dice, sabemos que en ocasiones hay exenciones de pago o descuentos, por lo cual el precio del curso no siempre es el precio pagado, así que agregamos también el precio pagado por el curso.

`inscripcion` = **`folio`** + **`matricula`** + **`codigo_curso`** + **`momento_inscripcion`** + **`precio_pagado`**

Observa cómo la experiencia te va dando elementos para que no te tomes todo tan literal. Tú como modelador de la base de datos puedes hacer agregados que consideres pertinentes para que los fenómenos queden debidamente registrados.

Determinar tipos de datos y propiedades especiales

Para concluir el ejercicio, nada más nos falta determinar los tipos de datos correctos para cada dato. Quedan de la siguiente manera.

En la tabla **`alumno`**, tenemos.

a) **`matricula`**, aunque es un número de 6 posiciones, no se realizarán cálculos con dicho número, así que decidimos colocarlo como **`CHAR(6)`**.

b) **`primer_apellido`**, como es un texto que puede ser muy extenso o muy concreto, decidimos que sea variable; además, como los nombres pueden tener símbolos de varios lenguajes, decidimos que sea de caracteres dobles, así que lo establecemos como **`NVARCHAR(33)`**. La longitud la determinamos dividiendo el máximo de 100, entre los tres campos que lo conformarán. Hay varias maneras de manejar eso, pero nos vamos por la más simple.

c) **`segundo_apellido`**, y **`nombre`** también serán **`NVARCHAR(33)`**.

d) **`fecha_nacimiento`** es una fecha de la cual no necesitamos la hora, así que decidimos que sea **`DATE`**.

En la tabla **curso**, tenemos.

a) **codigo_curso**, es un código alfanumérico de máximo 5 caracteres, así que se queda como **CHAR(5)**.

b) **nombre_curso**, es una descripción que puede ser larga o corta, de máximo 100 posiciones; como no se dice nada de que se requieran caracteres dobles, se quedan simples, por lo tanto, queda como **VARCHAR(100)**.

c) El **precio**, al ser valor monetario, queda como **DECIMAL(19,4)**, por convencionalismo.

d) **maximo_asistentes**, al ser entero y al no poder ser mayor a 80, asignar **TINYINT** es más que suficiente.

e) id_ue, es un código alfanumérico de 10 posiciones, escrito en caracteres occidentales, así que será **CHAR(10)**.

En la tabla **inscripcion**, tenemos.

a) **folio**, que es un número entero, positivo y consecutivo, queda como **INT**.

b) **matricula**, se homologa a la matricula que está en alumno, así que también es **CHAR(6)**.

c) **codigo_curso**, se homologa al código que está en curso, y queda como **CHAR(5)**.

d) **momento_inscripcion**, al ser fecha con hora, quedaría como **DATETIME**.

e) Finalmente, **precio_pagado** al ser moneda, también queda como **DECIMAL(19,4)**.

3.3 Concepto de llaves y relaciones

Una *llave*, también conocida como *clave*, es un conjunto de campos que permiten identificar o localizar un determinado contenido dentro de una tabla. El concepto de llave no debe confundirse con el concepto de *índice*, que tiene que ver más con la implementación y desempeño de una base de datos, que con la modelación.

Si la llave está compuesta por un solo campo, se le da el nombre de *llave simple*; la llave está compuesta por dos o más campos, se le da el nombre de *llave compuesta*.

i) Llave primaria

Una *llave primaria* es el conjunto de campos mínimos, suficientes y necesarios para identificar como único un registro dentro de una tabla.

En una tabla, la llave primaria no puede repetir valores, por lo cual se dice que tiene *unicidad*. En el caso de las llaves primarias compuestas, la combinación de todos los atributos que componen la llave deben tener unicidad, aunque los campos por separado no la tengan.

Los campos que forman una llave primaria se llaman *atributos primos*; los campos que no forman parte de la llave primaria se les llama *atributos no primos*.

Un campo tiene atributo de *nulidad*, o como dicen, es *nullable*, cuando puede desconocerse su valor, es decir cuando puede ser nulo. Ningún atributo primo puede tener atributo de nulidad.

Cuando en una tabla se tienen dos o más llaves que podrían funcionar como llave primaria, se tiene que elegir a una como llave primaria, mientras que el resto se mantienen como *llaves candidatas*.

Cuando los atributos primos son características inherentes del sujeto, evento o estado, se dice que es una *llave primaria natural*; si una tabla requiere una llave primaria pero no tiene una llave primaria natural, se tiene que crear una *llave primaria ficticia*, que es una que inventamos, para que sirva como identificador. En el caso de atributos

clasificados, se sugiere una llave de caracteres; en el caso de eventos, se sugiere un número entero consecutivo.

Cuando las llaves primarias ficticias son enteros consecutivos, se les puede aplicar un atributo de *identidad*, o *identity*. Este atributo implica que el valor de la llave debe generarse por sí mismo a partir de un valor inicial, y se debe ir incrementando, aplicando un monto específico y constante, al que se le llama *semilla*, o *seed*.

Es muy común que cuando la llave primaria es compuesta y está formada por muchos campos, se genere también una llave primaria ficticia, para simplificar tareas de captura. Cuando se hace eso, es muy importante tomar en cuenta que la llave compuesta que simplificamos permanece como llave candidata, seguirá teniendo atributo de unicidad, y de todas maneras debe validarse.

Existe la falsa creencia que todas las tablas deben tener una llave primaria, pero eso es falso: hay tablas en las cuales recuperar un registro en particular es totalmente inútil, por ejemplo, tablas de encuestas de mercado anónimas, donde hay muchos registros iguales. Si no se piensa recuperar los registros de manera particular, una llave primaria carece de sentido.

En algunos sistemas, las llaves primarias se implementan a través de una *restricción de llave primaria*, o *primary key constraint*, y se le debe dar un nombre. Generalmente, el nombre es el mismo de la tabla, con el prefijo “pk”.

Las restricciones de llave primaria implican unicidad, por lo cual no es necesario establecerla de manera explícita, porque ya la tiene.

ii) Llave foránea

Una *llave foránea* es la presencia, en una tabla, de la llave primaria de otra tabla.

Cuando se presenta una llave foránea, realmente hablamos de la existencia de los mismos campos en dos tablas; en una, esos campos

constituyen una llave primaria, y no admiten valores repetidos; en la otra, constituyen una llave foránea, y pueden admitir valores repetidos. A los campos que están en una y otra tabla, se le llaman *atributos de coincidencia*.

iii) Relaciones entre tablas

Una llave foránea siempre apunta hacia la tabla donde los atributos de coincidencia son la llave primaria; cuando este fenómeno se presenta, se dice que entre las tablas existe una *relación*.

En una relación entre tablas, la tabla donde los atributos de coincidencia son la llave primaria se le llama *tabla fuerte*, o *strong entity*, y la tabla donde los atributos de coincidencia son la llave foránea se le llama *entidad débil* o *weak entity*.

En algunos sistemas, las llaves foráneas se implementan a través de una *restricción de llave foránea*, o *foregin key constraint*, y se le debe dar un nombre. Generalmente, el nombre es de tipo *snake_notation*, compuesta por "fk", el nombre de la entidad débil, y el nombre de la entidad fuerte.

Cuando hay relación entre tablas, seguramente nos encontraremos con que en la tabla débil y en la tabla fuerte habrá registros con el mismo valor en los atributos de coincidencia, y eso permite que ambas tablas estén conectadas. A los registros que tienen los mismos valores en los atributos de coincidencia, tanto en la tabla fuerte como en la tabla débil, se les llama *registros de coincidencia*.

iv) Concepto de dominio

Se entiende por *dominio* el conjunto de valores válidos para un campo.

El dominio puede ser de tres tipos.

a) **Dominio por tipo de dato:** Es el dominio que se establece por la simple elección del tipo de dato; por ejemplo, al seleccionar un tipo de dato BIT, automáticamente los valores

permitidos para el dato se limitan a 0 y 1. En el caso de los datos de tipo cadena, el dominio de tipo de dato se representa como una limitante de longitud.

b) **Dominio dependiente del modelo**, llamado también **dominio de integridad referencial:** Es el domino que se presenta cuando, habiendo una relación entre tablas, los valores en los atributos de correspondencia en la tabla débil deben estar previamente registrados en la tabla fuerte. Por ejemplo, si en una orden de compra hay un campo llamado `numero_cliente`, que es `INT` y llave foránea que conecta con la tabla `cliente`, no basta con que lo que queremos registrar en la tabla de órdenes de compra sea `INT`: es necesario que el número esté previamente registrado en la tabla `cliente`.

c) **Dominio de regla de negocio:** Es el dominio que se presenta porque la organización así lo decide. Por ejemplo, los hoteles no tienen piso 13, en un avión no hay asientos 13, no se usa el número de empleado 666. Ya sea por superstición, o por la razón que sea, la organización excluye ciertos valores, y eso afecta el dominio.

EJERCICIO 3.2: LLAVES Y RELACIONES

Analiza el modelo de datos que tienes hasta ahora.

Las tablas están definidas como sigue.

`alumno` = `matricula` + `primer_apellido` + `segundo_apellido` + `nombre` + `fecha_nacimiento`

`curso` = `codigo_curso` + `nombre_curso` + `precio` + `maximo_asistentes` + `id_ue`

`inscripcion` = `folio` + `matricula` + `codigo_curso` + `momento_inscripcion` + `precio_pagado`

Responde las siguientes preguntas:

I. ¿Cuántas tablas componen el modelo?
II. ¿Cuántas llaves primarias tiene el modelo?
III. ¿Cuántos atributos primos tiene el modelo?
IV. ¿Cuántas llaves candidatas hay en el modelo?
V. ¿Cuántas relaciones hay en el modelo de datos?
VI. ¿Cuántas llaves foráneas hay en el modelo de datos?

Numeralia de la base de datos

¿Cuántas tablas componen el modelo?

Son 3: **`alumno`**, **`curso`** e **`inscripción`**.

¿Cuántas llaves primarias tiene el modelo?

Son 3, una por cada tabla.

a) Para **`alumno`**, es **`matricula`**.
b) Para **`curso`**, es **`codigo_curso`**.
c) Para **`inscripción`**, es **`folio`**.

¿Cuántos atributos primos hay en el modelo?

Son 3.

`alumno.matricula`, **`curso.codigo_curso`**, **`inscripcion.folio`**

¿Cuántas llaves candidatas hay en el modelo?

Son 2.

a) En la tabla curso, **`id_ue`**, es valor equivalente a **`codigo_curso`**, y podría ser usada como llave primaria.

b) En la tabla inscripción, **`matricula`** + **`curso`** podrían ser una alternativa a la llave primaria ficticia, que es **`folio`**.

Ambas llaves candidatas deben tener atributo de unicidad.

¿Cuántas relaciones hay en el modelo de datos?

Son 2.

La llave primaria de **`curso`** está en **`inscripcion`**, por tanto, hay una relación entre ambas tablas, donde **`curso`** es la tabla fuerte, e **`inscripción`** es la tabla débil.

El atributo de coincidencia para esta relación es **`codigo_curso`**.

La llave primaria de **`alumno`** está en **`inscripción`**, por tanto, hay una relación entre ambas tablas, donde **`alumno`** es la tabla fuerte, e **`inscripción`** es la tabla débil.

El atributo de coincidencia para esta relación es **`matricula`**.

¿Cuántas llaves foráneas hay en el modelo de datos?

Son 2.

La primera está en **`incripcion`**. La llave foránea es **`matricula`**, que tiene correspondencia con la llave primaria de **`alumno`**.

La segunda está en **`incripcion`**. La llave foránea es **`codigo_curso`**, que tiene correspondencia con la llave primaria de **`curso`**.

4 NORMALIZACIÓN DE LA BASE DE DATOS

4.1 Vicios del modelo relacional

Los *vicios del modelo relacional* son condiciones que se presentan en un modelo de datos, que dificultan o complican el adecuado rendimiento y confiablidad de la base de datos, y son 3: Redundancia, Inconsistencia, y Falta de Integridad.

i) Redundancia

La *redundancia* es el almacenamiento de los mismos datos varias veces, dentro de una misma base de datos.

Sólo se permite un tipo de redundancia en el modelo relacional, llamado *redundancia controlada*, que se presenta en la duplicidad de almacenamiento que se tiene en una llave primaria y una llave foránea; esta redundancia es necesaria, ya que es la única forma de poder mantener la relación entre los registros de dos tablas.

La redundancia es problemática en el sentido de que mantenerla es muy laborioso. Si un dato se tiene en varios lugares, cuando se actualice el dato, debe actualizarse en todas partes.

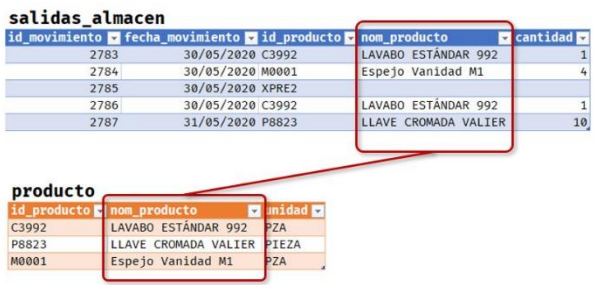


FIGURA 4.1: *Redundancia. Se repiten datos que se pueden recuperar.*

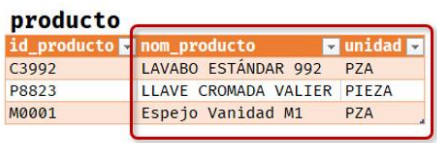
Como puede verse en la figura, en la tabla **salida_almacen** es innecesario colocar el **nom_producto**, pues al saber el **id_producto**, podemos recuperar el nombre. Si llegamos a actualizar algún nombre de producto en **producto**, tendríamos que actualizar también los nombres que están repetidos en **salidas_almacen**, para que la información fuera consistente.

Este vicio se elimina normalizando la base de datos, de tal forma que los datos queden en la tabla a la que más profundamente pertenezcan, y en ninguna otra parte más.

ii) Inconsistencia

La *inconsistencia* es la diferencia de significado o codificación de un mismo dato en diferentes partes de la base de datos.

Si en algún campo se permiten mayúsculas, y luego minúsculas, si se espera un dato numérico y se acepta un dato alfanumérico, si en los catálogos aparece un mismo concepto con dos claves diferentes, todo eso es inconsistencia.



producto

id_producto	nom_producto	unidad
C3992	LAVABO ESTÁNDAR 992	PZA
P8823	LLAVE CROMADA VALIER	PIEZA
M0001	Espejo Vanidad M1	PZA

FIGURA 4.2: *Inconsistencia de mayúsculas y minúsculas, y de claves.*

En la figura, puede apreciarse una doble inconsistencia.

La primera, en el campo **nom_producto**, se alterna mayúsculas y minúsculas, y eso se debe estandarizar. La segunda, en el campo **unidad**, se tiene dos maneras de referir las piezas; si se desea una consulta de todos los productos cuya **unidad** sea pieza, estaremos en un problema: pieza debería tener un código único y consistente.

La inconsistencia se soluciona categorizando adecuadamente la información, creando tablas de atributos clasificados, y haciendo validaciones a nivel campo.

Una definición adecuada de dominios puede ayudarnos mucho a resolver problemas de inconsistencia.

iii) Falta de integridad

La *falta de integridad* se presenta cuando, teniendo dos tablas relacionadas, se elimina en la entidad fuerte un registro que tiene asociados varios registros de coincidencia en la entidad débil.

En este caso, si se eliminan o modifican los datos de la entidad fuerte, deberán eliminarse o modificarse también los registros en la entidad débil.

FIGURA 4.3: *Falta de integridad. Dato apuntando a una referencia que no existe.*

En la figura se puede ver que hay una salida de almacén que refiere a un número de producto que de hecho ya no existe, por lo que se dice que hay falta de integridad, pues no están íntegros los datos para representar adecuadamente los datos.

Este vicio se elimina imponiendo restricciones para que no sea posible eliminar o modificar un registro de una entidad fuerte, si este tiene registros relacionados en la entidad débil, o bien, haciendo que los

registros en la entidad débil tengan la misma suerte que el registro en la entidad fuerte.

4.2 El proceso de normalización

El modelo relacional de bases de datos está basado en el modelo de álgebra relacional y en la teoría de conjuntos. Esto quiere decir que entre más se apeguen las estructuras de datos a los esquemas del modelo relacional, más eficiente será la recuperación y explotación de información.

Una forma de procurar que las estructuras de las bases de datos se apeguen a las reglas del álgebra relacional, es aplicando el proceso denominado *normalización*, que consiste en aplicar reglas específicas que permitan a los datos tener una estructura de cohesión tal que permita llegar de cualquier elemento del modelo a cualquier otro elemento, utilizando relaciones entre los elementos en un ambiente de redundancia controlada.

Los beneficios que se obtienen con la normalización son los siguientes:

a) **Se evita la redundancia innecesaria de datos.** Sólo se permite la redundancia de valores en los atributos llave (primaria - foránea), y sólo con el fin de que sean posibles las relaciones entre elementos.
b) **Se evitan la inconsistencia de los datos.** Al evitarse la redundancia innecesaria, al actualizar un dato, ese dato queda actualizado para cualquier aplicación.
c) **Se evita la falta de integridad de los datos.** Al formar relaciones obligatorias entre los elementos, se evita la eliminación o actualización de en un elemento, que son requeridos por otro elemento.

i) Formas normales básica

Al conjunto de reglas que deben aplicarse al modelo de datos para que sea considerada normalizada, se le llaman *formas normales* (*normal forms*, o *NF*). Aunque hay un gran número de reglas en el álgebra relacional, para efectos de las bases de datos, se consideran básicas las primeras tres formas normales, que son:

a) **1NF (Primera forma normal):** Todos los atributos son atómicos.
b) **2NF (Segunda forma normal):** Todos los atributos no atómicos tienen dependencia funcional completa con la llave primaria.
c) **3NF (Tercera forma normal):** No existen dependencias funcionales transitivas.

Las formas normales son acumulativas, es decir, que cada forma normal exige el cumplimiento de la anterior.

ii) Caso práctico

Imagina que la facturación de una compañía de venta de suministros de cómputo está siendo mantenida en una hoja electrónica de cálculo, y desea ponerse en una base de datos.

Cada vez que un cliente realiza una compra, se registra la información de la factura, del cliente, de los productos que está comprando y las promociones que le fueron otorgadas.

La compañía por lo pronto maneja un límite máximo de 3 promociones por producto, en cada venta.

Una parte de los registros están así:

NumFact	FechaFact	NumClie	Correos	CodProd	DescProd	PR1	PR2	PR3	PrecioVenta	Cant	Umed	DescUMed
2345	01/01/2011	101	jperez@hotmail.com, jpz@gmail.com	3903	Unidad USB 20GB	6M	D15		$ 200.00	2	PZA	Pieza
2345	01/01/2011	101	jperez@hotmail.com, jpz@gmail.com	5632	Monitor HDTV 40"	6M			$ 6,000.00	1	KIT	Kit
2346	02/01/2011	124	ana@live.com	3903	Unidad USB 20GB	6M	D15		$ 200.00	3	PZA	Pieza
2347	02/01/2011	150		6722	Impresora láser HP	D15			$ 3,000.00	2	PZA	Pieza

FIGURA 4.4: *Datos no normalizados.*

Donde:

1. **`NumFact`:** Número de la factura.
2. **`FechaFact`:** Fecha en que se realiza la factura.
3. **`NumClie`:** Número que identifica a un cliente.
4. **`NomClie`**: Nombre del cliente al que se le emite la factura.
5. **`Correos`:** Correos electrónicos del cliente.
6. **`CodProd`:** Código del producto que se está comprando.
7. **`DescProd`:** Descripción del producto.
8. **`PR1`:** Código de promoción aplicable.
9. **`PR2`:** Código de promoción aplicable.
10. **`PR3`:** Código de promoción aplicable.
11. **`PrecioVenta`:** Precio al que se vendió el producto.
12. **`Cant`:** Cantidad de unidades vendidas.
13. **`UMed`:** Unidad de medida del producto vendido.
14. **`DescUMed`:** Descripción de la unidad de medida.

Se desea pasar la información a una base de datos relacional que esté bien normalizada.

iii) Dependencias funcionales

Antes de entender las reglas de normalización, es necesario entender el concepto de dependencias funcionales.

Una *dependencia funcional* es la circunstancia en la cual un atributo es determinante de otro ($X \rightarrow Y$).

Existen diferentes tipos de dependencia funcional:

a) **Dependencia Funcional Completa:** Es cuando un atributo no primo está determinado por la totalidad de la llave primaria.
b) **Dependencia Funcional Parcial:** Es cuando, existiendo una llave primaria compuesta, un atributo no primo es

determinado por parte de la llave primaria, pero no por su totalidad.

c) **Dependencia Funcional Transitiva:** Es cuando un atributo no primo es determinado por otro atributo no primo.

iv) 1NF (Primera Forma Normal)

Se cumple con la primera forma normal (1NF), sí y sólo sí:

a) Condición 1: La tabla tiene una llave primaria.
b) Condición 2: La llave primaria no contiene valores nulos.
c) Condición 3: Se tienen atributos uniformes a nivel registro.
d) Condición 4: Todos los atributos son atómicos.

Un *atributo atómico* es aquél que contiene un solo dato, y no puede dividirse. La división puede darse de varias maneras: *a)* un atributo contiene varios datos, o *b)* porque un mismo dato puede ponerse sin problemas en varios atributos, dado que en esencia es lo mismo.

Por *atributos uniformes*, nos referimos a que en una tabla todos los registros tienen el mismo número de atributos, es decir, no hay registros con más o con menos atributos que otros; además, los datos contenidos en los atributos tienen el mismo dominio, es decir, no se permite que, para un registro, un dato sea numérico, y para otro registro sea alfabético, por ejemplo.

En el ejemplo, se tiene una llave primaria, que es el número de factura, dado que es lo que queremos registrar (`NumFact`). Se cumple la condición 1. Como se puede ver, no se tienen valores nulos en los atributos primos, por lo que se cumple la condición 2.

Los atributos son uniformes (condición 3), puesto que todos los registros tienen el mismo número de atributos, y la información contenida es consistente en cuanto al dominio de tipo.

Sin embargo, se dan ciertas violaciones respecto a los atributos atómicos.

Por ejemplo, el campo **Correo** contiene más de un dato en su contenido.

Dos datos en un mismo atributo

NumFact	FechaFact	NumClie	NomClie	Correos	CodProd	DescProd	PR1	PR2	PR3	PrecioVenta	Cant
2345	01/01/2011	101	Juan Pérez	jperez@hotmail.com, jpz@gmail.com	3903	Unidad USB 20GB	6M	D15		$ 200.00	2
2345	01/01/2011	101	Juan Pérez	jperez@hotmail.com, jpz@gmail.com	5632	Monitor HDTV 40"	6M			$ 6,000.00	1
2346	02/01/2011	124	Ana Aguilar	ana@live.com	3903	Unidad USB 20GB	6M	D15		$ 200.00	3
2347	02/01/2011	150	Antonio Ortiz		6722	Impresora láser HP	D15			$ 3,000.00	2

FIGURA 4.5: *Campo con datos no atómicos.*

La forma de resolverlo es generando copias del registro. Cada valor distinto contenido en **Correo** constituye la variación entre uno y otro registro. La tabla quedaría como sigue.

NumFact	FechaFact	NumClie	NomClie	Correos	CodProd	DescProd	PR1	PR2	PR3	PrecioVenta	Cant
2345	01/01/2011	101	Juan Pérez	jperez@hotmail.com	3903	Unidad USB 20GB	6M	D15		$ 200.00	2
2345	01/01/2011	101	Juan Pérez	jpz@gmail.com	3903	Unidad USB 20GB	6M	D15		$ 200.00	2
2345	01/01/2011	101	Juan Pérez	jperez@hotmail.com	5632	Monitor HDTV 40"	6M			$ 6,000.00	1
2345	01/01/2011	101	Juan Pérez	jpz@gmail.com	5632	Monitor HDTV 40"	6M			$ 6,000.00	1
2346	02/01/2011	124	Ana Aguilar	ana@live.com	3903	Unidad USB 20GB	6M	D15		$ 200.00	3
2347	02/01/2011	150	Antonio Ortiz		6722	Impresora láser HP	D15			$ 3,000.00	2

FIGURA 4.6: *Tabla con datos atómicos.*

Otra violación es que se coloca el mismo dato en diferentes atributos, que son equivalentes. Cualquier clave de promoción puede ser alimentada en **PR1**, **PR2** y **PR3**, de forma indistinta; incluso, se corre el riesgo de repetir la clave en los tres campos. En ese caso, estamos hablando del mismo dato en tres atributos. Mira cómo se manifiesta este fenómeno.

Atributos que almacenan lo mismo

NumFact	FechaFact	NumClie	NomClie	Correos	CodProd	DescProd	PR1	PR2	PR3	PrecioVenta	Cant
2345	01/01/2011	101	Juan Pérez	jperez@hotmail.com, jpz@gmail.com	3903	Unidad USB 20GB	6M	D15		200.00	2
2345	01/01/2011	101	Juan Pérez	jperez@hotmail.com, jpz@gmail.com	5632	Monitor HDTV 40"	6M			6,000.00	1
2346	02/01/2011	124	Ana Aguilar	ana@live.com	3903	Unidad USB 20GB	6M	D15		200.00	3
2347	02/01/2011	150	Antonio Ortiz		6722	Impresora láser H	D15			3,000.00	2

FIGURA 4.7: *Tabla con múltiples campos equivalentes.*

La forma de resolverlo también es generar copias de registro para cada promoción encontrada y eliminar los atributos innecesarios, dejándose un sólo campo para el dato.

La solución quedaría como sigue:

NumFact	FechaFact	NumClie	NomClie	Correos	CodProd	DescProd	PR1	PR2	PR3	PrecioVenta	Cant
2345	01/01/2011	101	Juan Pérez	jperez@hotmail.com	3903	Unidad USB 20GB	6M			$ 200.00	2
2345	01/01/2011	101	Juan Pérez	jperez@hotmail.com	3903	Unidad USB 20GB	D15			$ 200.00	2
2345	01/01/2011	101	Juan Pérez	jpz@gmail.com	3903	Unidad USB 20GB	6M			$ 200.00	2
2345	01/01/2011	101	Juan Pérez	jpz@gmail.com	3903	Unidad USB 20GB	D15			$ 200.00	2
2345	01/01/2011	101	Juan Pérez	jperez@hotmail.com	5632	Monitor HDTV 40"	6M			$ 6,000.00	1
2345	01/01/2011	101	Juan Pérez	jpz@gmail.com	5632	Monitor HDTV 40"	6M			$ 6,000.00	1
2346	02/01/2011	124	Ana Aguilar	ana@live.com	3903	Unidad USB 20GB	6M			$ 200.00	3
2346	02/01/2011	124	Ana Aguilar	ana@live.com	3903	Unidad USB 20GB	D15			$ 200.00	3
2347	02/01/2011	150	Antonio Ortiz		6722	Impresora láser HP	D15			$ 3,000.00	2

FIGURA 4.8: *Tabla con campos atómicos, y sin múltiples datos equivalentes.*

Obviamente, el resultado de aplicar la 1NF dista mucho de ser ineficiente. Muchos datos se repiten muchas veces sin necesidad, pero es en las siguientes formas normales donde eso se corrige.

v) 2NF (Segunda Forma Normal)

Se está en segunda forma normal (2NF), sí y sólo sí:

a) Condición 1: Se cumple con la primera forma normal (1NF).
b) Condición 2: Si todos los atributos no primos tienen *dependencia funcional completa* con la llave primaria.
c) Condición 3: Si no existen dependencias funcionales parciales (sólo en el caso de tablas con llave primarias compuestas).

Una forma empírica de explorar las dependencias funcionales es observar si, para cada valor de llave, el valor de un atributo no primo permanece constante.

Concentrémonos en la siguiente porción de datos de ejemplo, relacionada con la factura 2345:

NumFact	FechaFact	NumClie	CodProd	DescProd	PrecioVenta	Cant	Umed	DescUMed
2345	01/01/2011	101	3903	Unidad USB 20GB	$ 200.00	2	PZA	Pieza
2345	01/01/2011	101	5632	Monitor HDTV 40"	$ 6,000.00	1	KIT	Kit

FIGURA 4.9: *Dependencia funcional completa.*

Queda claro que cuando **`NumFact`** es 2345, **`FechaFact`** siempre es 01/01/2011, y que **`NumClie`** siempre es 101; podemos decir, que tanto **`FechaFact`** como **`NumClie`** tienen dependencia funcional completa con la llave primaria.

Por otro lado, también puedes comprobar que cuando **`NumFact`** es 2345, los campos **`CodProd`**, **`DescProd`**, **`PrecioVenta`**, **`Cant`**, **`UMed`** y **`DescUMed`** pueden tener diferentes valores, por lo que se dice que no tienen *dependencia funcional completa*.

Cuando eso sucede, los campos que no tienen dependencia funcional completa se ponen en otra tabla; generalmente, deberá incluir la llave primaria de la tabla original, y complementarla con uno o más campos para formar la llave primaria de la nueva tabla.

La solución a este dilema sería como sigue; toma cuenta que esta solución o es todavía la definitiva:

NumFact	FechaFact	NumClie
2345	01/01/2011	101

NumFact	CodProd	DescProd	PrecioVenta	Cant	Umed	DescUMed
2345	***3903***	Unidad USB 20GB	$ 200.00	2	PZA	Pieza
2345	***5632***	Monitor HDTV 40"	$ 6,000.00	1	KIT	Kit

FIGURA 4.10: *División de una tabla, en generales y detalle.*

Fíjate cómo en la primera tabla se queda la llave primaria original, más los campos que tenían dependencia funcional completa. En la segunda tabla, se coloca la llave primaria de la primera tabla, más los campos con los cuales no se tenía dependencia funcional completa. Revisa cómo el atributo primo de la primera tabla, más un atributo de la

segunda, que en conjunto formen una llave primaria compuesta con unicidad.

Cuando hay tablas con llave compuesta, es necesario hacer una verificación adicional. Si algún atributo no primo no está determinado por la *totalidad* de la llave primaria, sino por parte de ella, se dice que hay *dependencia funcional parcial*, y en ese caso, entonces es necesario generar más tablas alternativas, hasta que esa condición se cumple en todo el modelo.

NumFact	CodProd	DescProd	PrecioVenta	Cant	Umed	DescUMed
2345	***3903***	Unidad USB 20GB	$ 200.00	2	PZA	Pieza
2345	***5632***	Monitor HDTV 40"	$ 6,000.00	1	KIT	Kit

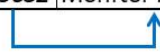


FIGURA 4.11: *Dependencia funcional parcial.*

Cuando una llave tiene una llave primaria simple, no puede haber dependencias funcionales parciales, porque no hay parcialidades en la llave.

En el ejemplo, la descripción de los productos (**DescProd**) depende de manera completa del código de producto (**CodProd**), y no de la totalidad de la llave, por lo cual, se presenta una dependencia funcional parcial y es necesario dividir en más tablas.

En resumen, la segunda forma normal exige que los atributos no primos tengan dependencia funcional completa con la llave primaria, y si la llave primaria es compuesta, se debe verificar que no existen dependencias funcionales parciales.

FACTURA

NumFact	FechaFact	NumClie
2345	01/01/2011	101
2346	02/01/2011	124
2347	02/01/2011	150

DETALLEFACTURA

NumFact	CodProd	PrecioVenta	Cant	Umed	DescUMed
2345	***3903***	$ 200.00	2	PZA	Pieza
2345	***5632***	$ 6,000.00	1	KIT	Kit
2346	***3903***	$ 200.00	3	PZA	Pieza
2347	***6722***	$ 3,000.00	2	PZA	Pieza

CLIENTES

NumClie	NomClie
101	Juan Pérez
124	Ana Aguilar
150	Antonio Ortiz

CORREOSCLIENTE

NumClie	Correos
101	***jperez@hotmail.com***
101	***jpz@gmail.com***
124	***ana@live.com***

PRODUCTOS

CodProd	DescProd
3903	Unidad USB 20GB
5632	Monitor HDTV 40"
3903	Unidad USB 20GB
6722	Impresora láser HP

PROMOCIONESPRODUCTOS

CodProd	PR1
3903	***6M***
3903	***D15***
5632	***6M***
6722	***D15***

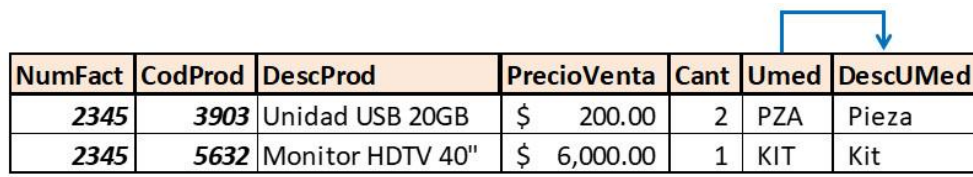
FIGURA 4.12: *Modelo en 2ª Forma normal.*

vi) 3NF (Tercera Forma Normal)

Se está en tercera forma normal (3NF), si y sólo sí:

a) Condición 1: Se cumple con la segunda forma normal (2NF)
b) Condición 2: No existen dependencias funcionales transitivas.

Una dependencia funcional transitiva se presenta cuando un atributo no primo tiene dependencia funcional completa respecto a uno o varios atributos no primos. En nuestro ejemplo, la única dependencia funcional transitiva se da en la tabla que almacena el detalle de la factura.



NumFact	CodProd	DescProd	PrecioVenta	Cant	Umed	DescUMed
2345	*3903*	Unidad USB 20GB	$ 200.00	2	PZA	Pieza
2345	*5632*	Monitor HDTV 40"	$ 6,000.00	1	KIT	Kit

FIGURA 4.13: *Dependencia funcional transitiva.*

En este caso, existe una dependencia funcional transitiva entre `UMed` y `DescUMed`; en ese sentido, se debe decidir cuál de los dos campos es el que operará como llave, y es el campo que permanecerá en la tabla.

Los campos que tienen la dependencia funcional pasarán a formar una nueva tabla, que heredará la llave primaria, quedando como sigue.

DETALLEFACTURA

NumFact	CodProd	PrecioVenta	Cant	Umed
2345	*3903*	$ 200.00	2	PZA
2345	*5632*	$ 6,000.00	1	KIT
2346	*3903*	$ 200.00	3	PZA
2347	*6722*	$ 3,000.00	2	PZA

UNIDADMEDIDA

Umed	DescUMed
PZA	Pieza
KIT	Kit
PZA	Pieza
PZA	Pieza

FIGURA 4.14: *Solución a la dependencia funcional transitiva.*

vii) BCNF (Forma Normal Boyce Codd)

Se le llama *llave candidata* al conjunto de atributos mínimo, suficiente y necesario para identificar como único un registro dentro de una tabla, pero que no es la llave primaria, aunque podría serlo.

Esto se da cuando se tienen dos posibles llaves que podrían actuar como llave primaria. En nuestro ejemplo, la tabla **curso** tiene como llave primaria **codigo_curso**, pero además tiene un campo llamado **id_ue**, que es un código equivalente que podría funcionar como llave primaria.

Se está en BNF cuando:

Condición 1: Se cumple con la tercera forma normal (3NF).

Condición 2: Todos los atributos no primos tienen dependencia funcional completa con las llaves candidatas.

Si un modelo está en BCNF, está indudablemente en 3NF, pero no todo el modelo que está en 3NF, está en BCNF; esto principalmente sucede cuando la llave primaria es compuesta, y la llave candidata es simple, lo que nos lleva a cuestionar si no era más eficiente elegir llave primaria a la candidata.

viii) Desnormalización

El proceso de *desnormalización* implica violar las formas normales con el fin de proporcionarle al uso de la base prestaciones de desempeño o de economía de recursos.

El modelo de base de datos relacional se basa en el álgebra relacional, y la normalización es un proceso que ayuda a que el modelo esté estructurado y pueda utilizarse de una manera eficiente. Esto es, al menos en teoría.

Existen tres formas básicas de desnormalizar:

a) Creación de llave primaria ficticia.
b) Partición horizontal.
c) Partición vertical.

La creación de llave primaria ficticia se da cuando, para no tener una llave primaria compuesta de muchos campos, se genera una llave que no forma parte de los datos inherentes a lo que se quiere registrar: generalmente es una clave numérica consecutiva, y se pone sólo con el objeto de identificar. Esto origina que los campos que originalmente eran la llave primaria se constituyen como una llave candidata, y con ello se generen dependencias funcionales transitivas.

Por otro lado, un modelo de base de datos normalizado no siempre es eficiente dependiendo de la volumetría de los datos. Muchos datos complican la acción de recuperar información, o de cálculo de información.

Al normalizar, las dependencias entre tablas terminan siendo siempre relaciones uno a muchos. Las reglas de normalización ordenan que dos tablas que tienen la misma llave primaria deben juntarse en una sola, por lo cual las relaciones uno a uno, no son válidas.

Pero imagina que hay una tabla de una biblioteca, con millones de registros, en donde se hacen miles de consultas, pero el resultado del 90% de las consultas tienen que ver sólo con los libros más demandados, que son el 5% del total de los libros. ¿Tiene caso barrer los millones de registros cada vez? La respuesta es que, desde el punto de vista del performance sería conveniente tener una tabla con el 5% más utilizado, y en caso de que no se encuentre el libro en esa tabla, se proceda a buscar en una tabla con los mismos atributos y la misma llave, que contenga el resto de los registros. Cuando se divide una misma tabla, dejando una parte de los registros en una tabla y el resto en otra, se le llama *partición*

horizontal, y se provoca un escenario de dos tablas con los mismos campos y las mismas llaves primarias, lo que se opone a la normalización.

Otro uso muy común de las particiones horizontales son la división de datos por periodo de tiempo, es decir, en la tabla de operación se dejan los datos del año en curso, mientras que los datos de los años anteriores se guardan en una tabla de datos históricos, por ejemplo.

También se da el caso que una tabla tenga una gran cantidad de atributos, pero que una gran cantidad de ellos se encuentren sin datos para la mayoría de los registros. En ese caso, se puede dejar una tabla con los atributos que generalmente tienen contenido, y crear otra tabla con la misma llave, que contenga los atributos que generalmente están vacíos o sin valor. A esta división se le llama *partición vertical*.

EJERCICIO 4.1: NORMALIZACIÓN

Tienes la siguiente tabla de datos, y se pide que normalices hasta BCNF.

id_pelicula	id_actor	id_pais	nom_pelicula	estreno	ingresos	nom_actor	nom_pais	genero1	otros_generos
28929	2	3	Revenant	16/12/2015	$532,950,503	Leonardo DiCaprio	Canada	Adventure	Action
28930	3	1	The Big Short	11/12/2015	$133,440,870	Christian Bale	United States	Drama	Thriller
28930	4	1	The Big Short	11/12/2015	$133,440,870	Steve Carell	United States		
28931	5	2	Pepe El Toro	21/08/1953	null	Pedro Infante	México	Drama	
28932	6	1	The professional	14/09/1994	$19,552,639	Gary Oldman	United States	Action	
28932	7	1	The professional	14/09/1994	$19,552,639	Natalie Portman	United States	International	Drama, Crime
28933	3	1	American Psycho	14/04/2000	$34,266,564	Christian Bale	United States	Crime	Thriller, Terror

FIGURA X: *Sin normalizar.*

Tenemos información de películas, los actores que participaron en ellas, y el género de las películas.

Se pide lo siguiente:

I. Normalizar hasta 1NF

II. Normalizar hasta 2NF

III. Normalizar hasta 3NF

IV. Normalizar hasta BCNF

Normalizar hasta 1NF

Primero, se aplican las modificaciones para cumplir con la primera forma normal (1NF).

Se tienen que eliminar los atributos no atómicos.

Te puedes dar cuenta de dos violaciones a esta regla: primero, **`genero1`** y **`otros_generos`** tienen información una misma naturaleza, así que se deben interpretarse como un solo dato; toma en cuenta que ese campo está relacionado con las películas, y que las películas están identificadas por **`id_pelicula`**.

También puedes ver que en **`otros_generos`** contiene listas de valores, que deben estar por separado.

¿Cómo se resuelve?

Se agrupan los campos **`genero1`** y **`otros_generos`**, y se interpretan como un solo dato, al que llamaremos **`nom_genero`**; después, se enumeran todos los valores en los campos, en todos los registros relacionados con una misma película.

Por ejemplo, para la película 28929, *The Revenant*, los valores resultantes para **`nom_genero`** son *Adventure* y *Action*: para la película 28932, *The Professional*, los valores para **`nom_genero`** son *Action*, *International*, *Drama* y *Crime*.

Ahora, por cada valor para **`nom_genero`**, se repiten el resto de los campos. Queda como sigue. No le prestes mucha atención al valor en el campo **`nom_actor`**, puedes poner el nombre de cualquier actor en dicha película.

id_pelicula	id_actor	id_pais	nom_pelicula	estreno	ingresos	nom_actor	nom_pais	nom_genero
28929	2	3	Revenant	16/12/2015	$ 532,950,503	Leonardo DiCaprio	Canada	Adventure
28929	2	3	Revenant	16/12/2015	$ 532,950,503	Leonardo DiCaprio	Canada	Action
28930	3	1	The Big Short	11/12/2015	$ 133,440,870	Christian Bale	United States	Drama
28930	4	1	The Big Short	11/12/2015	$ 133,440,870	Steve Carell	United States	Thriller
28931	5	2	Pepe El Toro	21/08/1953	null	Pedro Infante	México	Drama
28932	6	1	The professional	14/09/1994	$ 19,552,639	Gary Oldman	United States	Action
28932	7	1	The professional	14/09/1994	$ 19,552,639	Natalie Portman	United States	International
28932	7	1	The professional	14/09/1994	$ 19,552,639	Natalie Portman	United States	Drama
28932	7	1	The professional	14/09/1994	$ 19,552,639	Natalie Portman	United States	Crime
28933	3	1	American Psycho	14/04/2000	$ 34,266,564	Christian Bale	United States	Crime
28933	3	1	American Psycho	14/04/2000	$ 34,266,564	Christian Bale	United States	Thriller
28933	3	1	American Psycho	14/04/2000	$ 34,266,564	Christian Bale	United States	Terror

FIGURA 4.15: *Datos en 1NF.*

Normalizar hasta 2NF

Después, se aplican las modificaciones para cumplir con la segunda forma normal (2NF). Esto se logra si los atributos no primos tienen una dependencia funcional completa con una llave primaria. Esto también implica que no existan dependencias funcionales parciales.

Te podrás dar cuenta que, cuando el **id_pelicula** es 28932, **id_pais**, **nom_pelicula**, **estreno**, **ingresos** y **nom_pais**, no cambian, así que se puede aislar una tabla con la siguiente estructura.

id_pelicula	id_pais	nom_pelicula	estreno	ingresos	nom_pais
28929	3	Revenant	16/12/2015	$ 532,950,503	Canada
28929	3	Revenant	16/12/2015	$ 532,950,503	Canada
28930	1	The Big Short	11/12/2015	$ 133,440,870	United States
28930	1	The Big Short	11/12/2015	$ 133,440,870	United States
28931	2	Pepe El Toro	21/08/1953	null	México
28932	1	The professional	14/09/1994	$ 19,552,639	United States
28932	1	The professional	14/09/1994	$ 19,552,639	United States
28932	1	The professional	14/09/1994	$ 19,552,639	United States
28932	1	The professional	14/09/1994	$ 19,552,639	United States
28933	1	American Psycho	14/04/2000	$ 34,266,564	United States
28933	1	American Psycho	14/04/2000	$ 34,266,564	United States
28933	1	American Psycho	14/04/2000	$ 34,266,564	United States

FIGURA 4.16: *película, en 2NF.*

A los campos que quedan en la otra tabla, se les agrega el campo de identificación de la primera tabla, que es **id_pelicula**.

id_pelicula	id_actor	nom_actor	nom_genero
28929	2	Leonardo DiCaprio	Adventure
28929	2	Leonardo DiCaprio	Action
28930	3	Christian Bale	Drama
28930	4	Steve Carell	Thriller
28931	5	Pedro Infante	Drama
28932	6	Gary Oldman	Action
28932	7	Natalie Portman	International
28932	7	Natalie Portman	Drama
28932	7	Natalie Portman	Crime
28933	3	Christian Bale	Crime
28933	3	Christian Bale	Thriller
28933	3	Christian Bale	Terror

FIGURA 4.17: *Datos desnormalizados en 2NF.*

Esta tabla puede almacenar los actores que participan en las películas, pero también puede ser una tabla que almacena las películas y sus géneros. Es un predicamento que se puede resolver de la siguiente manera.

Separa el **`id_pelicula`** con todo lo relacionado con los actores, es decir, **`id_actor`** y **`nom_actor`**.

Separa el **`id_pelicula`** con todo lo relacionado a géneros, es decir, **`nom_genero`**.

id_pelicula	id_actor	nom_actor
28929	2	Leonardo DiCaprio
28929	2	Leonardo DiCaprio
28930	3	Christian Bale
28930	4	Steve Carell
28931	5	Pedro Infante
28932	6	Gary Oldman
28932	7	Natalie Portman
28932	7	Natalie Portman
28932	7	Natalie Portman
28933	3	Christian Bale
28933	3	Christian Bale
28933	3	Christian Bale

id_pelicula	nom_genero
28929	Adventure
28929	Action
28930	Drama
28930	Thriller
28931	Drama
28932	Action
28932	International
28932	Drama
28932	Crime
28933	Crime
28933	Thriller
28933	Terror

FIGURA 4.18: *Separación de información 2NF.*

Como podrás ver, hay registros repetidos. Procedemos a eliminar registros repetidos, y de paso, se ordenan.

id_pelicula	id_actor	nom_actor
28929	2	Leonardo DiCaprio
28930	3	Christian Bale
28930	4	Steve Carell
28931	5	Pedro Infante
28932	6	Gary Oldman
28932	7	Natalie Portman
28933	3	Christian Bale

id_pelicula	nom_genero
28929	Action
28929	Adventure
28930	Drama
28930	Thriller
28931	Drama
28932	Action
28932	Crime
28932	Drama
28932	International
28933	Crime
28933	Terror
28933	Thriller

FIGURA 4.19: *Separación de información 2NF, sin duplicados.*

La primera tabla almacena la información de los actores y las películas, así que sería una buena idea que se llamara **actor_pelicula**, que tendría los campos **id_pelicula** + **id_actor** + **nom_actor**. El mínimo de campos que podríamos usar para identificar como únicos los registros sería **id_pelicula** y **id_actor**, que formarían la llave primaria.

La segunda tabla almacena la información de los géneros y las películas, así que sería una buena idea que se llamara **genero_pelicula**, que tendría los campos **id_pelicula** + **nom_genero**. El mínimo de campos que podríamos usar para identificar como únicos los registros sería **id_pelicula** y **nom_genero**, que formarían la llave primaria.

No me gusta que **nom_genero** sea una etiqueta descriptiva que actúa como atributo primo, así que decido crear una llave ficticia para género. Agrego un campo que se llame **id_genero**, luego ordeno por **nom_genero**, y asigno un consecutivo para cada género, quedando de la siguiente manera.

Quedaría como sigue:

id_pelicula	id_genero	nom_genero
28929	1	Action
28932	1	Action
28929	2	Adventure
28932	3	Crime
28933	3	Crime
28930	4	Drama
28931	4	Drama
28932	4	Drama
28932	5	International
28933	6	Terror
28930	7	Thriller
28933	7	Thriller

FIGURA 4.20: *genero_pelicula, con llave ficticia creada.*

Luego, vuelvo a ordenar por **id_pelicula** y **id_genero**.

Ahora, la llave primaria de genero_pelicula sería **id_pelicula** + **id_genero**.

En resumen, las tablas están como sigue hasta el momento.

`pelicula` = **`*id_pelicula`** + **`id_pais`** + **`nom_pelicula`** + **`estreno`** + **`ingresos`** + **`nom_pais`**

`actor_pelicula` = **`*id_pelicula`** + **`*id_actor`** + **`nom_actor`**

`genero_pelicula` = **`*id_pelicula`** + **`*id_genero`** + **`nom_genero`**

En 2NF se deben evitar las dependencias funcionales parciales, que son las que se pueden presentar cuando, teniendo una llave primaria compuesta, un atributo no primo tiene dependencia funcional completa con parte de la llave primaria, pero no con toda la llave primaria.

En nuestro caso, las tablas con llave primaria compuesta son **`actor_pelicula`** y **`genero_pelicula`**, y es donde podemos buscar dependencias funcionales parciales.

Podemos darnos cuenta de que en **`actor_pelicula`**, se tiene una dependencia funcional parcial, porque **`nom_actor`** depende de **`id_actor`**, y no de toda la llave primaria. Por esa razón, se generan dos tablas: la tabla de origen, sin los campos que tienen dependencia funcional parcial, y una tabla formada por los campos que representan la dependencia funcional parcial.

`actor_pelicula` = **`*id_pelicula`** + **`*id_actor`**

`actor` = **`*id_actor`** + **`nom_actor`**

Podemos darnos cuenta también de que en **`genero_pelicula`**, se tiene una dependencia funcional parcial, porque **`nom_genero`** depende de **`id_genero`**, y no de toda la llave primaria. Por esa razón, se generan dos tablas: la tabla de origen, sin los campos que tienen dependencia funcional parcial, y una tabla formada por los campos que representan la dependencia funcional parcial.

`genero_pelicula` = **`*id_pelicula`** + **`*id_genero`**

`genero` = **`*id_genero`** + **`nom_genero`**

Al final, queda esto, después de la 2NF.

`pelicula` = **`*id_pelicula`** + **`id_pais`** + **`nom_pelicula`** + **`estreno`** + **`ingresos`** + **`nom_pais`**

`actor_pelicula` = **`*id_pelicula`** + **`*id_actor`**

`actor` = **`*id_actor`** + **`nom_actor`**

`genero_pelicula` = **`*id_pelicula`** + **`*id_genero`**

`genero` = **`*id_genero`** + **`nom_genero`**

Quedando 5 tablas, al final de 2NF.

Normalizar hasta 3NF

Después, se aplican las modificaciones para cumplir con la tercera forma normal (3NF). Esto se logra si no hay dependencias funcionales parciales, es decir, si no hay atributos no primos con dependencia funcional completa con otro atributo no primo.

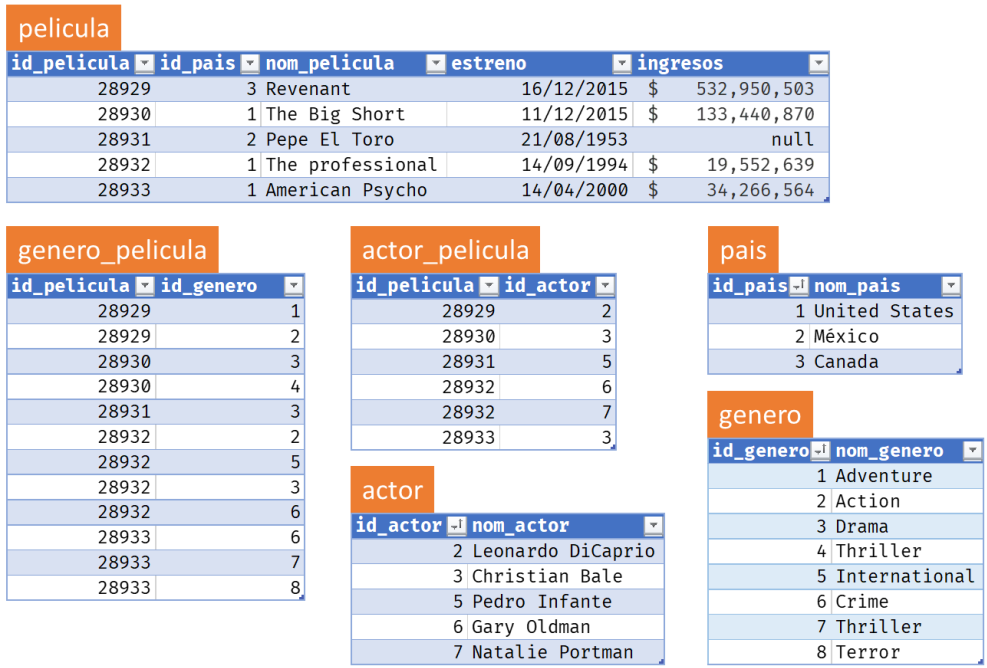
De inicio, las tablas con sólo un atributo no primo, ya están en 3NF, porque no hay manera de que existan dependencias entre campos, pues sólo hay uno.

La única tabla con más de un atributo no primo es **`pelicula`**, y como podemos ver, el atributo no primo llamado **`id_pais`** tiene dependencia funcional con **`nom_pais`**. Para resolverlo, se genera una tabla con los atributos que tienen dependencia, y en la tabla original se deja la llave de la nueva tabla. Quedaría como sigue:

`pelicula` = **`*id_pelicula`** + **`id_pais`** + **`nom_pelicula`** + **`estreno`**+ **`ingresos`**

`pais` = **`*id_pais`** + **`nom_pais`**

La base de datos en 3NF luce así.



pelicula

id_pelicula	id_pais	nom_pelicula	estreno	ingresos
28929	3	Revenant	16/12/2015	$ 532,950,503
28930	1	The Big Short	11/12/2015	$ 133,440,870
28931	2	Pepe El Toro	21/08/1953	null
28932	1	The professional	14/09/1994	$ 19,552,639
28933	1	American Psycho	14/04/2000	$ 34,266,564

genero_pelicula

id_pelicula	id_genero
28929	1
28929	2
28930	3
28930	4
28931	3
28932	2
28932	5
28932	3
28932	6
28933	6
28933	7
28933	8

actor_pelicula

id_pelicula	id_actor
28929	2
28930	3
28931	5
28932	6
28932	7
28933	3

actor

id_actor	nom_actor
2	Leonardo DiCaprio
3	Christian Bale
5	Pedro Infante
6	Gary Oldman
7	Natalie Portman

pais

id_pais	nom_pais
1	United States
2	México
3	Canada

genero

id_genero	nom_genero
1	Adventure
2	Action
3	Drama
4	Thriller
5	International
6	Crime
7	Thriller
8	Terror

FIGURA 4.21: *base normalizada hasta BCNF.*

Normalizar hasta BCNF

Después, se aplican las modificaciones para cumplir con Boyce Code Normal Form (BCNF). Esto se logra si las reglas de normalización aplican para las llaves candidatas, en caso de que existan. En nuestro caso, no hay tablas con llaves candidatas, por lo cual ya se encuentra también en BCNF. Nuestra base está normalizada.

5 DOCUMENTACIÓN DE BASE DE DATOS

5.1 Diccionario de datos

Un *diccionario de datos* es una relación tabulada de campos y sus atributos.

No hay una sola distribución de campos para un diccionario de datos, pues depende para qué se requiere el diccionario, serán los campos pertinentes.

i) Campos básicos

Los campos mínimos que a mi juicio debería tener un diccionario de datos, son los siguientes.

a) **Nombre de la base de datos o catálogo.** Nombre de la base de datos o catálogo al que pertenece el campo.
b) **Nombre de la tabla.** Nombre de la tabla a la que pertenece el campo.
c) **Nombre del campo.** Nombre del campo.
    a. En ocasiones, se puede simplificar la base de datos, la tabla y el nombre de campo, en una sola entrada en el diccionario.
    b. El diccionario deberá estar en orden alfabético, respecto a base de datos, tabla y nombre de campo.

d) **Tipo de dato.** Tipo de dato del campo.
e) **Longitud.** Describe la máxima longitud que puede tener el dato.
    a. Se utiliza primordialmente para los datos de tipo cadena, que siempre lo deben especificar.
    b. Hay tipos de datos que tienen una longitud fija, determinada por el tipo de dato: `bit`, `tinyint`, `smallint`, `int`, `date`, `datetime`, `bool`, entre otros. En estos casos, no se especifica la longitud, porque se asume.

f) **Precisión.** Se utiliza con tipos de datos numéricos con punto flotante (`decimal`, `float`, por ejemplo), y es el número de dígitos que compone todo el número.
   a. En un número 38832.292, la precisión sería 8.

g) **Escala.** Se utiliza con tipos de datos numéricos con punto flotante (`decimal`, `float`, por ejemplo), y es el número de dígitos después del punto decimal.
   a. En un número 38832.292, la precisión sería 3.

h) **Contenido.** Descripción breve del contenido del campo.

ii) Campos particulares

a) **PK.** Sí o No. Indica si el campo es atributo primo.
   a. Si en una tabla sólo hay un campo con este atributo, la llave primaria es simple.
   b. Si en una tabla hay más de un campo con este atributo, la llave primaria es compuesta.

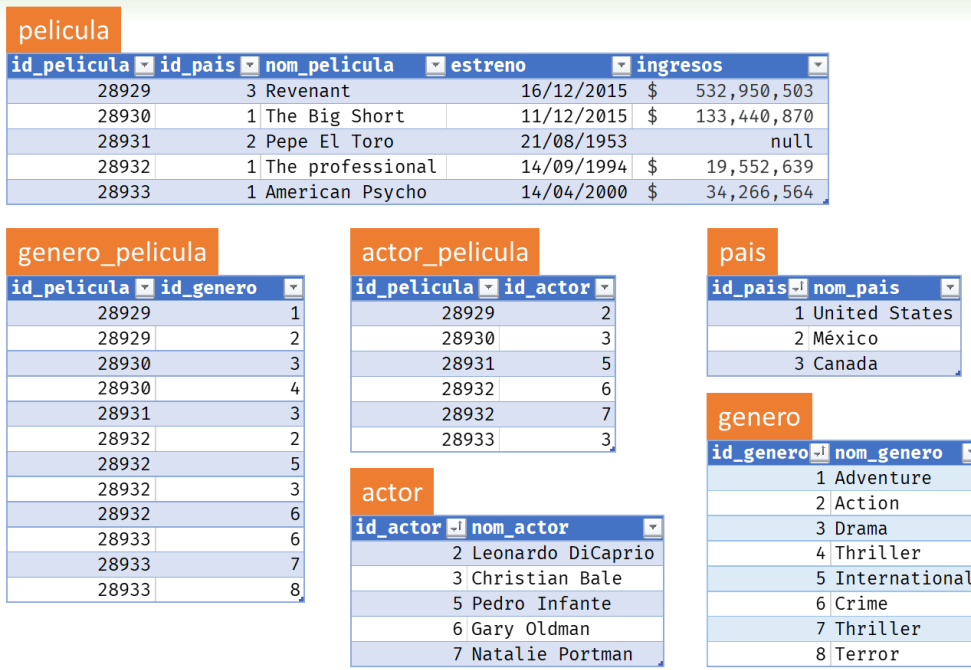
b) **FK.** Sí o No. Indica si el campo participa en una llave foránea.
c) **Null.** Sí o No. Indica si el campo admite ser nulo (*nullable*).
   c. Los atributos primos no pueden ser nulos.
   d. Los atributos que forman llave foránea sólo admiten nulos en caso de que la relación entre las tablas admita opcionalidad.

d) **Unique.** Sí o No. Indica si el campo debe tener un valor único en la tabla, es decir, que no está permitido repetir valores).
e) **UC.** Uniqe compuesto. Indica si el campo, en conjunto con otros campos, dentro del ámbito de la misma tabla, deben tener unicidad. Se presenta en llaves primarias compuestas.
   a. Se coloca una letra A, a todos los campos que en conjunto son únicos.
   b. Como puede haber varios conjuntos de campos con unicidad en una misma tabla, al segundo conjunto se le coloca B, C, y así.

c. En caso de que un campo participe en dos o más conjuntos Unique Compuesto, las letras se separarán las letras de los conjuntos, usando comas. Ejemplo: A, B.

EJERCICIO 5.1: DICCIONARIO DE DATOS

Toma en cuenta el resultado final del proceso de normalización del ejercicio anterior, que dejó el siguiente modelo de datos.



pelicula

id_pelicula	id_pais	nom_pelicula	estreno	ingresos
28929	3	Revenant	16/12/2015	$ 532,950,503
28930	1	The Big Short	11/12/2015	$ 133,440,870
28931	2	Pepe El Toro	21/08/1953	null
28932	1	The professional	14/09/1994	$ 19,552,639
28933	1	American Psycho	14/04/2000	$ 34,266,564

genero_pelicula

id_pelicula	id_genero
28929	1
28929	2
28930	3
28930	4
28931	3
28932	2
28932	5
28932	3
28932	6
28933	6
28933	7
28933	8

actor_pelicula

id_pelicula	id_actor
28929	2
28930	3
28931	5
28932	6
28932	7
28933	3

actor

id_actor	nom_actor
2	Leonardo DiCaprio
3	Christian Bale
5	Pedro Infante
6	Gary Oldman
7	Natalie Portman

pais

id_pais	nom_pais
1	United States
2	México
3	Canada

genero

id_genero	nom_genero
1	Adventure
2	Action
3	Drama
4	Thriller
5	International
6	Crime
7	Thriller
8	Terror

FIGURA 5.1: *Base normalizada hasta BCNF.*

Documenta el diccionario de datos.

Diccionario de datos

Antes de documentar el diccionario de datos, valdría mucho la pena hacer las siguientes precisiones y agregados respecto.

a) Asumamos que la base de datos no registrará más de 256 géneros, ni más de 256 países, pero es posible que registre más de 256 actores y más de 256 películas.

b) Asumamos que todas las descripciones que aparecen tienen una desviación estándar en su longitud de más de 2, por lo que deberán ser de longitud variable.

BD	Tabla	Campo	Tipo	Longitud	Presición	Escala	PK	FK	Null	Unique	UC	Contenido
BD	actor	id_actor	INT				Sí	No	No	Sí		Número de identificador del actor.
BD	actor	nom_actor	VARCHAR	50			No	No	No	No		Nombre artístico del actor.
BD	actor_pelicula	id_actor	INT				Sí	Sí	No	No	A	Número de identificador del actor que participó en una película.
BD	actor_pelicula	id_pelicula	INT				Sí	Sí	No	No	A	Número de identificación de la película en la que actuó un actor.
BD	genero	id_genero	TINYINT				Sí	No	No	Sí		Número de identificación del género cinematográfico.
BD	genero	nom_genero	VARCHAR	50			No	No	No	No		Nombre del género cinematográfico.
BD	genero_pelicula	id_genero	TINYINT				Sí	Sí	No	Sí	A	Número de identificación del género al que pertenece una película.
BD	genero_pelicula	id_pelicula	INT				Sí	Sí	No	No	A	Número de identificación de la película que pertenece a un determinado
BD	pais	id_pais	TINYINT				Sí	No	No	Sí		Número de identificación del país.
BD	pais	nom_pais	VARCHAR	50			No	No	No	No		Nombre del país.
BD	pelicula	id_pelicula	INT				Sí	No	No	Sí		Número de identificación de la película.
BD	pelicula	estreno	DATE				No	No	No	No		Fecha de estreno de la película.
BD	pelicula	id_pais	TINYINT				No	Sí	No	No		Identificador del país donde se produjo la película.
BD	pelicula	ingresos	DECIMAL		19	4	No	No	Sí	No		Últimos ingresos totales reportados.
BD	pelicula	nom_pelicula	NVARCHAR	100			No	No	No	No		Nombre de la película, en su idioma original.

FIGURA 5.2: *Diccionario de datos con campos básicos y particulares.*

Toma en cuenta las siguientes particularidades.

a) Sólo se especifica como nullable aquellos campos que tenemos evidencia de que puede no saberse la información.

b) En el caso de las llaves compuestas, mira como la unicidad se da en el conjunto de campos, y no en los campos individuales.

c) La tabla se ordenó siguiendo estas reglas:

   a. Base de datos, ascendente.

   b. Nombre de la tabla, ascendente.

   c. PK, descendente.

   d. Nombre de campo, ascendente.

5.2 Diagrama de Entidad Relación

El *Diagrama de Entidad Relación* (*ERD / Entity Relationship Diagram*) es una representación gráfica que muestra las relaciones existentes entre tablas de un mismo modelo de datos, además de su cardinalidad y opcionalidad.

La ventaja de disponer de un DER es que en todo momento puedes visualizar la estructura de datos de tu modelo. En general, la secuencia para su creación es la siguiente:

a) Representar las tablas (rectángulos).
b) Representar las relaciones (líneas conectoras de ángulos rectos).
c) Representar la cardinalidad y opcionalidad.

i) Representación de tablas

Las tablas se representan en forma de rectángulos, y tienen dos niveles de detalle: general, donde sólo se muestra el nombre de la tabla; y

detallada, donde se muestra además información de los campos, sus tipos de dato, y su participación en las llaves.

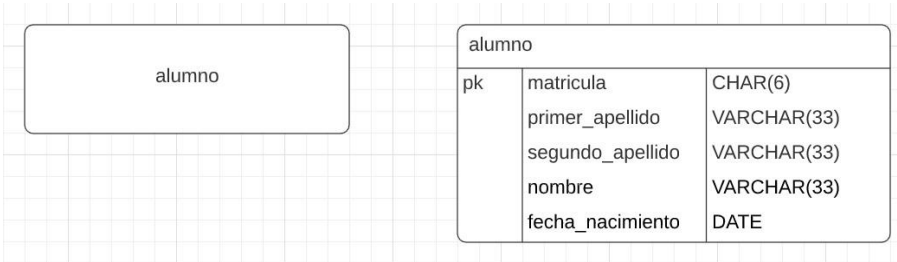


FIGURA 5.3: *Tabla general y detallada. Hockey Notation. Elaborado en LucidChart.*

i) Representar las relaciones

Las *relaciones* entre las tablas se representan gráficamente como líneas conectoras de ángulos rectos. Lo ideal es que conecten de atributo de coincidencia a atributo de coincidencia.

¿Cómo se puede saber qué tablas están relacionadas con otras? La estrategia es ver la llave primaria de una tabla, y buscar si los atributos primos de dicha tabla están presentes en otra tabla, como llaves foráneas. Si hay coincidencias, entonces las tablas están relacionadas.

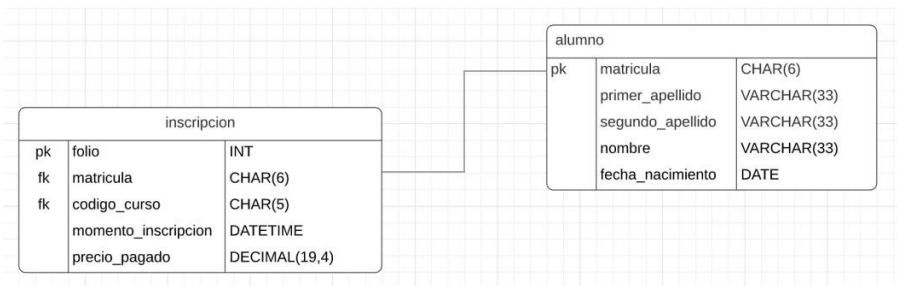


FIGURA 5.4: *Relación entre tablas.*

Observa en la figura X, cómo la llave primaria de `alumno`, en este caso `matricula`, está presente como llave foránea en la tabla `inscripcion`; por esa razón, se traza una línea de ángulos rectos que conecta los atributos de coincidencia, es decir `alumno.matricula` y `inscripcion.matricula`.

ii) Representación de cardinalidad y opcionalidad

Saber que dos tablas están relacionadas es importante, aunque también debemos saber *qué tipo de relación es*.

Debemos entender por *cardinalidad*, la correspondencia de registros de coincidencia que hay en una relación entre tablas.

La notación gráfica que más me gusta para representar un diagrama de entidad relación, es el *ERD de Hockey*, también conocida como *Crow's Foot*, o pata de cuervo. En dicha notación, la relación entre tablas se ilustra a través de líneas rectángulas, y la cardinalidad, a través de líneas que indican el tipo de cardinalidad.

La *opcionalidad*, por otra parte, representa la posibilidad de ausencia de registros de coincidencia que hay en una relación entre tablas. En la notación de Hockey, se ilustra como un pequeño círculo, que indica cero.

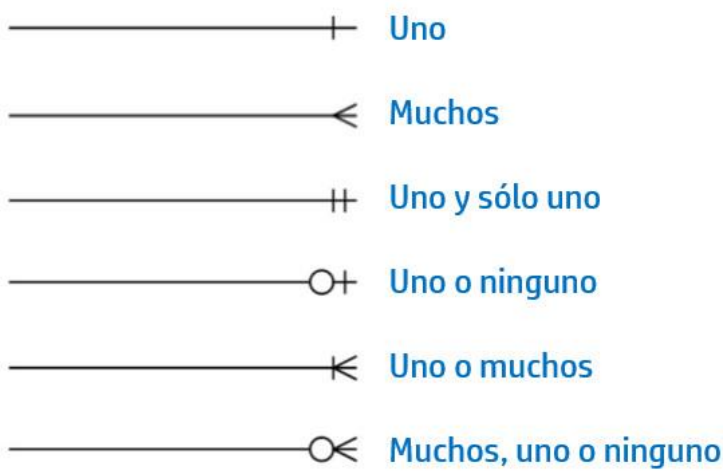


FIGURA 5.5: *Símbolos para ilustrar la cardinalidad y opcionalidad, en Notación Hockey.*

La cardinalidad se expresa en los extremos de la línea. Cuando tenemos una llave primaria conectándose con una llave foránea, estamos frente a una *relación de uno a muchos*, es decir, uno de un lado, a muchos

del otro; es importante distinguir las formas especiales, porque al decir uno, podemos decir uno, uno y sólo uno, y uno o ninguno; por otro lado, tenemos muchos, uno o muchos, o muchos o ninguno. Toma en cuenta que «muchos» puede ser uno nada más.

Si a la cardinalidad le agregas un cero, quiere decir que existe la posibilidad que no haya registros de coincidencia. Toma en cuenta que la no opcionalidad *implica obligatoriedad*.

Analiza lo siguiente:

a) Observa la relación existente entre `alumno` e `inscripcion`.
b) Sabemos que están relacionadas, porque ambas tienen el atributo de coincidencia `matricula`.
c) Dado que el atributo de coincidencia es llave primaria en `alumno`, entonces alumno es la tabla fuerte (*strong table*), mientras que `inscripcion` es la tabla débil (*weak table*).
d) Hay una regla: a la tabla fuerte le toca la cardinalidad de uno, y a la tabla débil, la cardinalidad de muchos. Estamos ante una relación de uno a muchos que es la más típica.
e) ¿Esta es la relación que necesitamos? A fin de cuentas, es una relación uno a muchos, donde uno está del lado de la tabla fuerte, y el muchos del lado de la tabla débil.

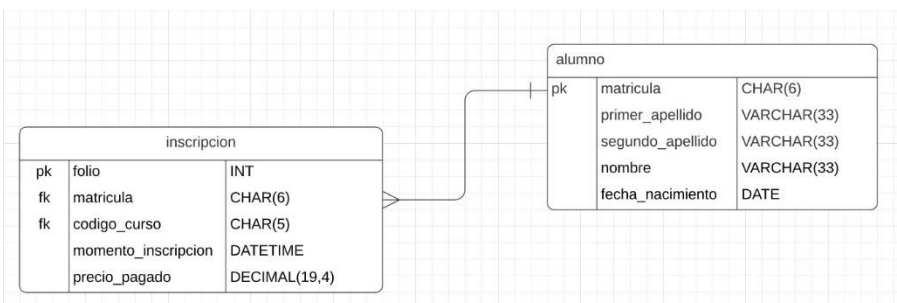


FIGURA 5.6: *Relación uno a muchos, sin opcionalidad.*

f) Esa relación está equivocada. Recuerda que la ausencia de opcionalidad implica obligatoriedad. La representación indica que hay obligatoriedad en ambos sentidos.

g) La forma de validar esto es la siguiente. Empecemos preguntando, al agregar un alumno a la tabla **`alumno`**, ¿cuántos registros de coincidencia deben existir en **`inscripcion`**? La verdad es que ninguno: un alumno, aunque debe inscribirse a un curso tarde que temprano, tiene plazo para hacerlo, así que es totalmente normal que el alumno no se haya inscrito a ningún curso, y, por tanto, no tenga registros de coincidencia. Por lo tanto, la relación debe tener opcionalidad hacia el lado de **`inscripcion`**. El registro en alumno no lo requiere.

h) Al agregar una inscripción en **`inscripcion`**, ¿cuántos registros de coincidencia deben existir en **`alumno`**? La verdad es que toda inscripción debe señalar necesariamente a un alumno. De hecho, si no se dice qué alumno se inscribe, la inscripción no tiene sentido, así que no hay opcionalidad, es decir, hay obligatoriedad.

i) Es necesario entonces revisar la semántica, para poder definir con precisión la cardinalidad correcta.

j) El trazado correcto es el siguiente:

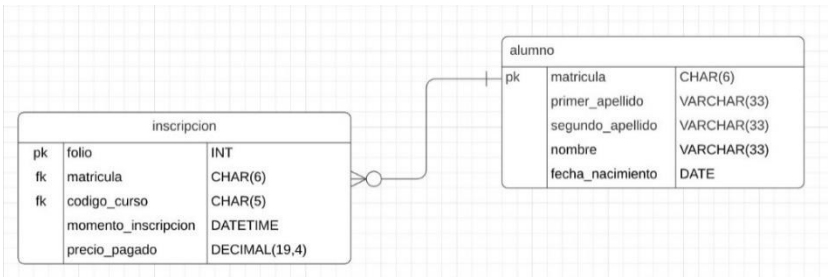


FIGURA 5.7: *Relación uno a muchos con opcionalidad. Elaborado con LucidChart.*

iii) Relaciones especiales

Hay varias relaciones especiales que merecen atención aparte.

a) Relaciones uno a uno.
b) Relaciones muchos a muchos.
c) Relaciones recursivas.

Una *relación uno a uno* se da entre tablas que poseen la misma llave primaria, en donde una es extensión de la otra.

En ocasiones, los modelos de datos aplican una técnica llamada *partición vertical*, es decir, parten una tabla en dos tablas por cuestiones de desempeño de la base de datos.

Imagínate que tienes una tabla donde almacenas información de productos; ahora imagina que, en contadas ocasiones, el producto es de importación, y tienes necesidad de guardar información relacionada con permisos y temas legales. Tener campos utilizados ocasionalmente, provoca problemas de manejo y desempeño.

Lo que se hace generalmente, es que se generan dos tablas con la misma llave primaria, donde una es la extensión de la otra. Si se presenta la situación donde se requiera actualizar los campos relacionados con la importación, pues se repite la llave primaria en la segunda tabla, y se llenan los datos complementarios, que estarán disponibles a través de la llave.

Aquí es cuando se genera una relación uno a uno, donde uno de los extremos tiene opcionalidad. Desde el punto de vista de modelado, una relación uno a uno no tiene justificación, pero dese el punto de vista de implementación, puede tenerlo.

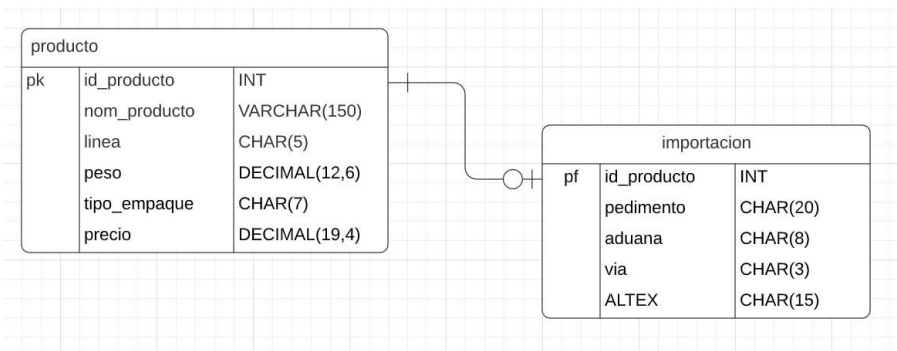


FIGURA 5.8: *Relación uno a uno. Elaborado en LucidChart.*

Una *relación muchos a muchos* se da cuando, semánticamente, una tabla puede tener muchos registros de coincidencia en ambas tablas relacionadas. Imagina que quieres relacionar actores con películas. Obviamente, un actor puede participar en muchas películas, pero en una película participan muchos actores.

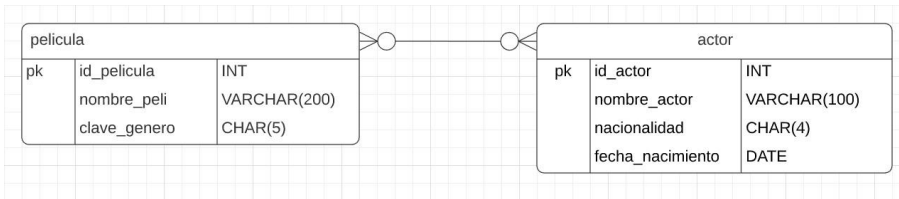


FIGURA 5.9: *Relación muchos a muchos, en modelación. Elaborado en LucidChart.*

La forma en que se maneja esta situación es creando una *tabla intermedia*, también llamada *tabla de relación*, que contendrá las llaves foráneas de las tablas que representaban la relación original. En la implementación, una relación muchos a muchos entre dos tablas, pasa a ser una relación de uno a muchos con opcionalidad, de dos tablas fuertes, a una tabla intermedia débil, que contiene las llaves primarias de las otras tablas. Las tablas de relación son consideradas tablas de *estado*: no son ni sujetos, ni eventos.

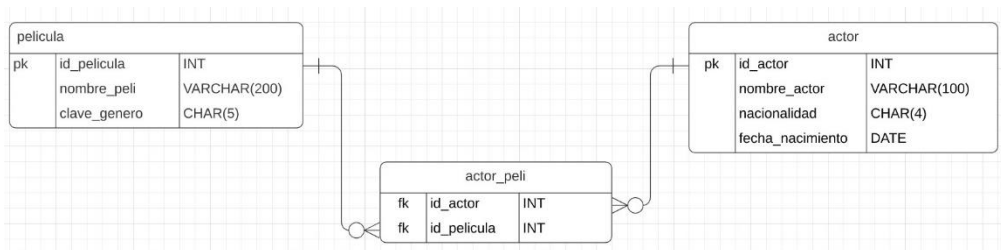


FIGURA 5.10: *Relación muchos a muchos, en implementación. Elaborado en LucidChart.*

Si nos fijamos bien, `inscripcion` actúa como tabla de relación, entre `alumno` y `curso`, porque contiene la llave primaria de ambas tablas. Como puede comprobarse, un alumno puede tomar muchos cursos, y los cursos pueden ser tomados por muchos alumnos.

Desde el punto de modelado, la relación muchos a muchos es válida, pero en implementación está contraindicada: tiene que usarse una tabla de relación.

Finalmente, se tienen las *relaciones recursivas*, en donde una tabla tiene relación consigo misma. El ejemplo típico es el de la tabla de empleados cuya llave primaria es el número de empleado; en la misma tabla, hay un campo que almacena el número de empleado del superior jerárquico; obviamente, dicha información está en la misma tabla, así que la tabla establece una relación uno a muchos consigo misma.

FIGURA 5.11: *Relación recursiva. Elaborado en LucidChart.*

EJERCICIO 5.2: DIAGRAMA DE ENTIDAD RELACIÓN

Toma en cuenta el diccionario de datos de nuestro modelo de ejemplo.

BD	Tabla	Campo	Tipo	Longitud	Presición	Escala	PK	FK	Null	Unique	UC	Contenido
BD	actor	id_actor	INT				Sí	No	No	Sí		Número de identificador del actor.
BD	actor	nom_actor	VARCHAR	50			No	No	No	No		Nombre artístico del actor.
BD	actor_pelicula	id_actor	INT				Sí	Sí	No	No	A	Número de identificador del actor que participó en una película.
BD	actor_pelicula	id_pelicula	INT				Sí	Sí	No	No	A	Número de identificación de la película en la que actuó un actor.
BD	genero	id_genero	TINYINT				Sí	No	No	Sí		Número de identificación del género cinematográfico.
BD	genero	nom_genero	VARCHAR	50			No	No	No	No		Nombre del género cinematográfico.

BD	genero_pelicula	id_genero	TINYINT				Sí	Sí	No	Sí	A	Número de identificación del género al que pertenece una película.
BD	genero_pelicula	id_pelicula	INT				Sí	Sí	No	No	A	Número de identificación de la película que pertenece a un determinado
BD	pais	id_pais	TINYINT				Sí	No	No	Sí		Número de identificación del país.
BD	pais	nom_pais	VARCHAR	50			No	No	No	No		Nombre del país.
BD	pelicula	id_pelicula	INT				Sí	No	No	Sí		Número de identificación de la película.
BD	pelicula	estreno	DATE				No	No	No	No		Fecha de estreno de la película.
BD	pelicula	id_pais	TINYINT				No	Sí	No	No		Identificador del país donde se produjo la película.
BD	pelicula	ingresos	DECIMAL		19	4	No	No	Sí	No		Últimos ingresos totales reportados.
BD	pelicula	nom_pelicula	NVARCHAR	100			No	No	No	No		Nombre de la película, en su idioma original.

FIGURA 5.12: *Diccionario de datos con campos básicos y particulares.*

A partir de esa tabla, elabora el diagrama de entidad relación.

Distribución concéntrica preliminar

Se deben representar las tablas, con sus campos y atributos, procurando que queden al centro las tablas principales, sean sujetos y eventos, en segunda instancia, las tablas de relación, y al final, las tablas fuertes, o de catálogo.

De forma abstracta, en nuestro ejemplo, sería más o menos así:

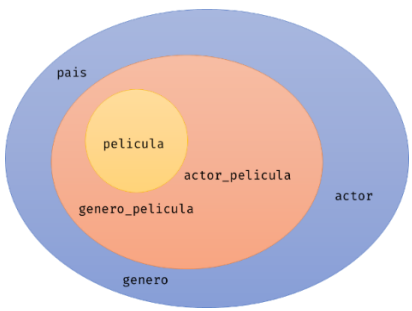


FIGURA 5.13: *Distribución concéntrica de las tablas, en un DER.*

Representar las tablas

De acuerdo con el diccionario de datos, quedaría de la siguiente manera. No olvides que posteriormente podrás distribuir las tablas como mejor te parezca, por lo pronto, comienza ilustrando, buscando la semejanza con la distribución concéntrica.

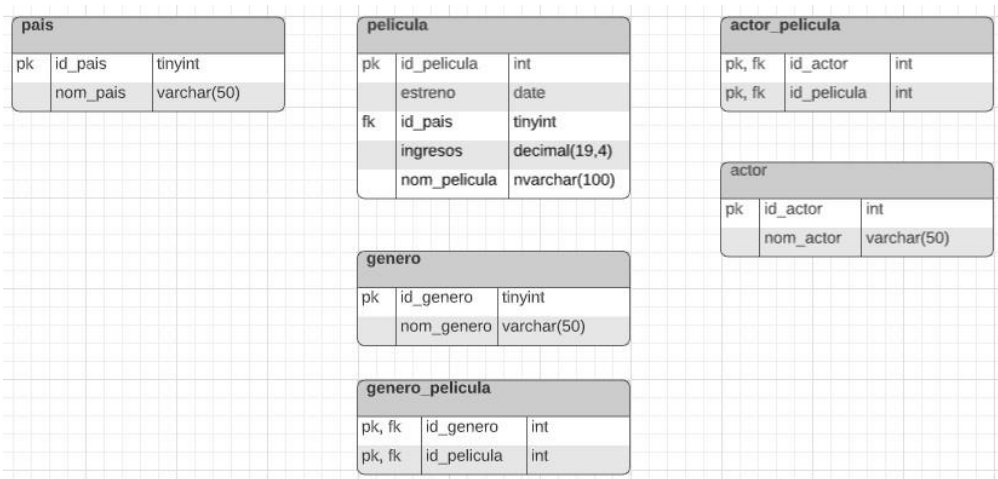


FIGURA 5.14: *Representación de tablas usando LucidChart.*

Representar las relaciones y la cardinalidad

Se debe buscar la existencia de la llave primaria de una tabla, en otra tabla, y extender una relación entre las tablas.

Tomemos por ejemplo **`pelicula`** y **`pais`**.

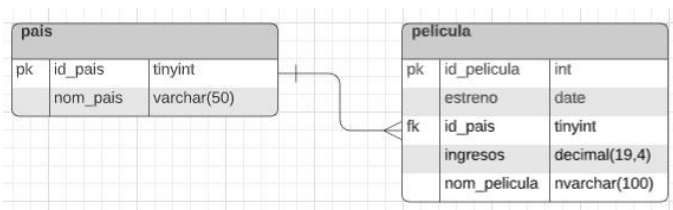


FIGURA 5.15: *Relación uno a muchos, entre país y película.*

Dado que la llave primaria de **`pais`** está en **`pelicula`**, se establece una relación entre las tablas, es decir, se traza una línea de ángulos rectos entre ellas.

El atributo de coincidencia es **`id_pais`**, por lo que se sugiere que la línea vaya de **`pais.id_pais`**, a **`pelicula.id_pais`**.

Algunos editores son lo suficientemente inteligentes para reconocer en qué tabla el atributo de coincidencia es la llave primaria, y colocan la cardinalidad «uno» en la tabla fuerte, y «muchos», en la tabla débil.

La relación, como está expresada en este momento, indica que cada puede ser producida en uno y solo un país, y cada país, necesariamente, debe tener muchas (más de una) películas producidas.

Esto es un poco inexacto, porque al capturar un país, no necesariamente tengo que capturar en ese momento una película producida en dicho país.

Representar la opcionalidad

La opcionalidad existe cuando, en una relación entre tablas, una tabla fuerte no necesariamente debe tener registros de coincidencia en la tabla débil, como es el caso de la relación entre **`pais`** y **`película`**.

La opcionalidad se representa a través de un círculo, que indica que la relación puede tener cero coincidencias, quedando de forma correcta de la siguiente manera.

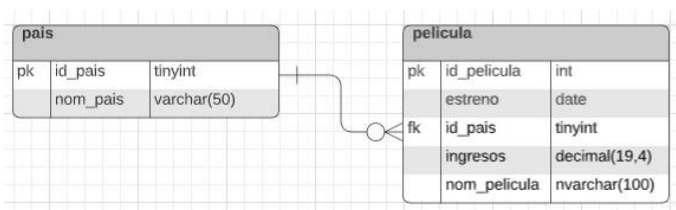


FIGURA 5.16: *Relación uno a muchos, entre país y película, con opcionalidad.*

Colocando las relaciones, cardinalidades y opcionalidades, el modelo quedaría como sigue. Observa cómo se distribuyeron las tablas, para evitar líneas cruzadas. Eso a veces es imposible, pero se debe minimizar al máximo.

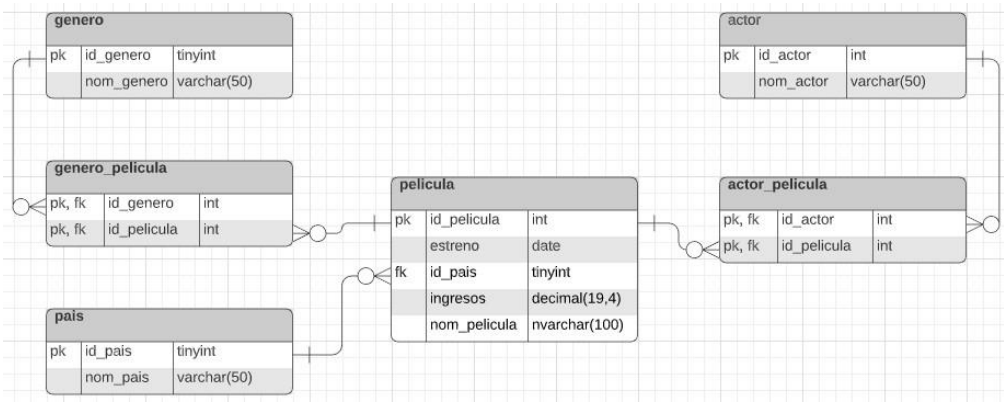


FIGURA 5.17: *Diagrama de entidad relación, con cardinalidades y opcionalidades.*

La opcionalidad es esencial, pues permite una carga de datos más sencilla que cuando no se tiene.

6 STRUCTURED QUERY LANGUAGE (SQL)

6.1 SQL Estándar

Tarde que temprano enfrentarás la situación de tener que especificar un almacén de datos que deseas construir, o bien, tendrás que especificar una consulta que te permita recuperar ciertos datos que necesitas.

Para eso, requerirás indicarles a tus herramientas qué es lo que quieres, y la forma más estandarizada para que lo hagas es usando el lenguaje SQL, que te permitirá crear y manipular estructuras de datos e información, independientemente de la herramienta que estés usando. SQL vendría siendo el *Esperanto* del manejo de datos.

El *lenguaje SQL*, o *Structured Query Language*, es un lenguaje estándar para almacenar, manipular y recuperar datos almacenados en bases de datos.

Muchos de los sistemas manejadores de bases de datos soportan el lenguaje SQL, aunque tienen sus propios dialectos y mejoras.

Aprender modelación de base de datos relacionales y el lenguaje de base de datos SQL te servirá para el manejo de muchos productos, como **SQL Server**, **Oracle**, **MySQL**, **Access**, **Informix**, entre muchos otros. También son importantes para comenzar a utilizar herramientas de inteligencia de negocios, lenguaje R, datos en Excel o en Google Sheets.

Como podrás apreciar, me refiero al lenguaje como “S-Q-L”, o “es-qiu-el”, en inglés; también encontrarás quien le llama “siquel”. Lo correcto es SQL, porque se trata de un acrónimo, y no lo digo yo, lo dice el estándar ISO.

Es un *lenguaje declarativo*, en el sentido que le especificas qué quieres que haga, pero no le puedes decir cómo quieres que lo haga.

El lenguaje tiene dos grupos de estatutos principales.

1. **DDL**, o **Data Definition Language**: Son los estatutos que permiten establecer la estructura de la base de datos y sus objetos. Los principales comandos son:
    a. **CREATE**. Permite crear elementos.
    b. **ALTER**. Permite modificar elementos ya creados.
    c. **DROP**. Permiten eliminar elementos ya creados.

2. **DML**, o **Data Manipulation Language**. Son los estatutos que permiten agregar, modificar o eliminar los datos en la base de datos. Los principales comandos son:
    a. **SELECT**. Permite recuperar datos almacenados en la base de datos.
    b. **INSERT**. Permite agregar registros a las tablas de la base de datos.
    c. **UPDATE**. Permite modificar los datos almacenados en los registros de la base de datos.
    d. **DELETE**. Permite eliminar registros almacenados en la base de datos.

En este libro, los estatutos SQL se mostrarán usando el lenguaje estándar, y de forma básica. Este es un libro de modelación, y la idea no es que aprendas a programar SQL, sino que puedas implementa de manera real el modelo de datos que has aprendido a diseñar, y que puedas interactuar con la base de datos para comprobar que el modelo es funcional.

El código de este libro podrás encontrarlo en GitHub:

```
https://github.com/FelipeRamirezPhD/AprendaModelacionBD
```

6.2 Trabajando con bases de datos

Para crear una base de datos usando el lenguaje SQL, lo único que debes hacer es utilizar el estatuto `CREATE DATABASE`, usando la siguiente sintaxis:

```
CREATE DATABASE nombre_bd;
```

El nombre de la base de datos puede ser cualquiera, siempre y cuando sea significativo. Como puedes ver, utilicé sólo letras minúsculas, y *snake_notation*, que es la notación en donde se separan las palabras con línea baja.

Una vez que se tiene creada la base de datos, requerimos poner en uso la base de datos, antes de trabajar con ella. Esto se logra con el estatuto `USE`.

```
USE nombre_bd;
```

Para eliminar una base de datos usando el lenguaje SQL, lo único que debes hacer es utilizar el estatuto `DROP DATABASE`, usando la siguiente sintaxis:

```
DROP DATABASE nombre_bd;
```

Hay una variante interesante, que intenta eliminar la base de datos sólo en caso de que exista.

```
DROP DATABASE IF EXISTS nombre_bd;
```

EJERCICIO 6.1: CREACIÓN DE UNA BASE DE DATOS

Escribe el código SQL para generar la base de datos y las tablas del modelo de datos de ejemplo. La base de datos se llamará **`filmes`**.

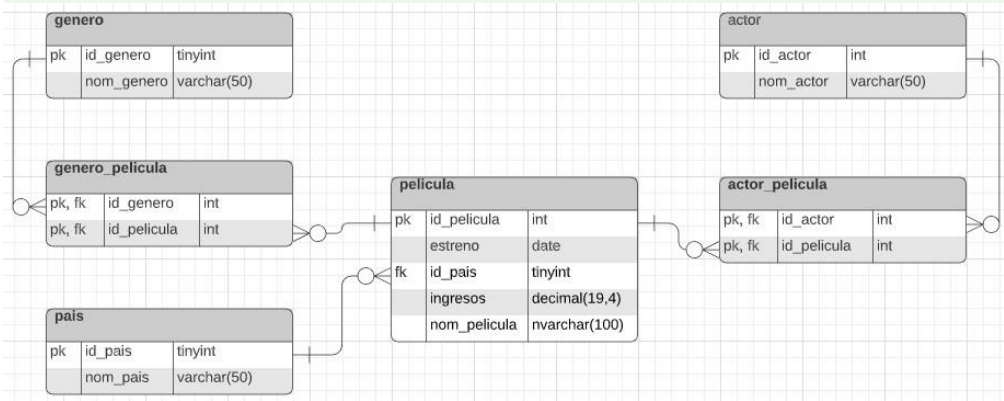


FIGURA 6.1: *Diagrama de entidad relación de la base de datos filmes.*

Se asume una base de datos en MySQL. Si la base de datos existe, debe eliminarse antes.

FILMES_DB.SQL

```
-- Si la BD existe, la elimina.
drop database if exists filmes;
-- Crea la base de datos.
create database filmes;
```

6.3 Trabajando con tablas

Para crear una tabla usando el lenguaje SQL, lo único que debes hacer es utilizar el estatuto `CREATE TABLE`, usando la siguiente sintaxis:

```
CREATE TABLE nombre_tabla (
      campo1 tipo_dato,
      campo2 tipo_dato,
      campo3 tipo_dato,
      …
);
```

El nombre de la tabla se sugiere que esté en singular, usando *snake_notation*, en minúsculas. Lo mismo aplica a los nombres de campo. Como puedes ver, la especificación de los campos requiere colocar el nombre que le darás al campo, más el tipo de dato que le corresponde. Los campos se separan por comas, y en su conjunto se les llama *lista de campos*.

Por omisión, todos los campos se generan admitiendo ser nulos; como podemos apreciar en nuestro diccionario de datos, eso no es lo más común. Se recomienda que, al momento de crear la tabla, se le especifique a los campos que no admitan ser nulos, el estatuto `NOT NULL`, usando la siguiente sintaxis:

```
CREATE TABLE nombre_tabla (
      campo1 tipo_dato NOT NULL,
      campo2 tipo_dato,
      campo3 tipo_dato NOT NULL,
      …
);
```

En este código, solo `campo2` admitiría ser nulo.

Si en algún momento ya se tiene una tabla creada, y se requiere agregar un campo adicional, o remover un campo existente se puede utilizar el estatuto `ALTER TABLE`.

Para agregar un campo, se utiliza la siguiente sintaxis.

```
ALTER TABLE nombre_tabla
ADD campo tipo_dato;
```

Para eliminar un campo, se utiliza la siguiente sintaxis.

```
ALTER TABLE nombre_tabla
DROP COLUMN columna;
```

Para eliminar tablas en lenguaje SQL, lo único que debes hacer es utilizar el estatuto `DROP TABLE`, usando la siguiente sintaxis:

```
DROP TABLE nombre_tabla;
```

El estatuto admite también la variante de eliminar la tabla siempre y cuando exista.

```
DROP TABLE IF EXISTS nombre_tabla;
```

Hay ocasiones en las cuales, lo que se desea es que la tabla esté vacía, es decir, que los registros de borren pero que la estructura permanezca. en esos casos, se debe utilizar el estatuto `TRUNCATE TABLE`.

```
TRUNCATE TABLE nombre_tabla;
```

EJERCICIO 6.2: CODIFICACIÓN DE TABLAS

Escribe el código SQL para generar las tablas del modelo de datos de ejemplo, en la base de datos filmes. Elimina las tablas, si es que existen.

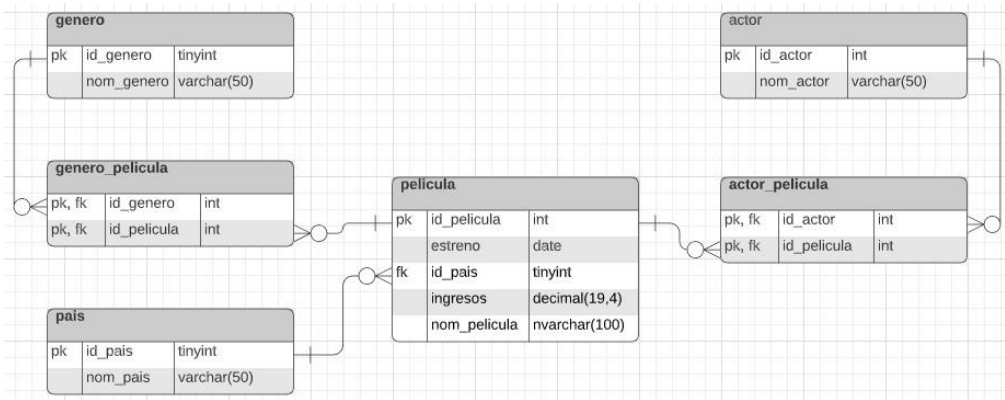


FIGURA 6.2: *Diagrama de entidad relación, con cardinalidades y opcionalidades.*

Se asume una base de datos en MySQL.

FILMES_TABLAS.SQL

```
-- Pone en uso la BD filmes.
use filmes;

-- Elimina las tablas del modelo, si existen.
drop table if exists actor;
drop table if exists actor_pelicula;
drop table if exists genero;
drop table if exists genero_pelicula;
drop table if exists pais;
drop table if exists pelicula;

-- Se especificará de manera explícita qué campos
-- no admiten ser nulos.

-- Crea la tabla actor.
create table actor(
    id_actor int NOT NULL,
    nom_actor varchar(50) NOT NULL
);
```

```
-- Crea la tabla actor_pelicula.
create table actor_pelicula(
    id_actor int NOT NULL,
    id_pelicula int NOT NULL
);

-- Crea la tabla genero.
create table genero(
    id_genero tinyint NOT NULL,
    nom_genero varchar(50) NOT NULL
);

-- Crea la tabla actor_pelicula.
create table genero_pelicula(
    id_genero tinyint NOT NULL,
    id_pelicula int NOT NULL
);

-- Crea la tabla pais.
create table pais(
    id_pais tinyint NOT NULL,
    nom_pais varchar(50) NOT NULL
);

-- Crea la tabla pelicula.
-- En esta tabla tenemos dos campos que pueden ser nulos
-- estreno, porque puede ser que la película ya exista, pero
-- no se haya estrenado aún, e ingresos, porque puede ser
-- que aún no los tenga, o que no se hayan reportado.
create table pelicula(
    id_pelicula int NOT NULL,
    estreno date,
    id_pais tinyint NOT NULL,
    ingresos decimal(19,4),
    nom_pelicula nvarchar(100) NOT NULL
);
```

6.4 Restricciones de llave primaria (*primary key*)

Para crear una llave primaria, lo primero que debes asegurarte es que los atributos primos no sean nulos. Esto se logra usando `ALTER TABLE` con esta sintaxis.

```
ALTER TABLE nombre_tabla
MODIFY nombre_campo tipo_dato NOT NULL;
```

Después de asegurarte que los atributos primos no pueden ser nulos, entonces puedes establecer las restricciones de llave primaria, usando `ALTER TABLE` con las siguientes sintaxis.

En caso de que la llave primaria sea simple.

```
ALTER TABLE nombre_tabla
ADD PRIMARY KEY (campo);
```

En caso de que la llave primaria sea compuesta.

```
ALTER TABLE nombre_tabla
ADD CONSTRAINT nombre_restricción
PRIMARY KEY (campo1 [, campo2, …]);
```

Se sugiere que `nombre_restricción` tenga la siguiente estructura: `pk_nombre_tabla`. La sintaxis para llave primaria compuesta funciona también con un campo, pero el nombre que se le ponga a la restricción es ignorado.

Si por alguna razón requieres redefinir la llave primaria, primero tienes que eliminar la que tienes, y volverla a crear. Para remover la llave primaria, tendrías que aplicar `ALTER TABLE` con la siguiente sintaxis.

MYSQL:

```
ALTER TABLE nombre_tabla
DROP PRIMARY KEY;
```

SQL SERVER/ORACLE/MS ACCESS:

```
ALTER TABLE nombre_tabla
DROP CONSTRAINT nombre_restricción;
```

EJERCICIO 6.3: RESTRICCIONES DE LLAVE PRIMARIA

Escribe el código SQL para generar las llaves primarias de todo el modelo de datos de ejemplo. Toma como base la siguiente tabla.

BD	Tabla	Campo
BD	actor	id_actor
BD	actor_pelicula	id_actor
BD	actor_pelicula	id_pelicula
BD	genero	id_genero
BD	genero_pelicula	id_genero
BD	genero_pelicula	id_pelicula
BD	pais	id_pais
BD	pelicula	id_pelicula

FIGURA 6.3: *Tabla de atributos primos del modelo.*

Se asume una base de datos en MySQL. Recuerda la sintaxis básica:

```
alter table tabla
add constraint pk_tabla
primary key (lista_de_atributos_primos);
```

El código quedaría así. Analízalo.

```
-- Pone en uso la BD filmes.
use filmes;

-- Creación de llaves primarias.
-- Se intenta asignar el nombre a la restricción
-- en todos los casos. Es posible que ciertas
-- versiones o manejadores ignoren el nombre
-- pero las llaves se generarán.
alter table actor
add constraint pk_actor
primary key (id_actor);

alter table actor_pelicula
add constraint pk_actor_pelicula
primary key (id_actor, id_pelicula);

alter table genero
add constraint pk_genero
primary key (id_genero);

alter table genero_pelicula
add constraint pk_genero_pelicula
primary key (id_genero, id_pelicula);

alter table pais
add constraint pk_pais
primary key (id_pais);

alter table pelicula
add constraint pk_pelicula
primary key (id_pelicula);
```

6.5 Restricciones de llave foránea (*foreign key*)

Una vez que tienes creadas las llaves primarias, puedes proceder a la creación de las llaves foráneas. Acuérdate que las llaves foráneas se definen en la tabla débil. Para crear una restricción de llave foránea, tendrías que aplicar `ALTER TABLE` con la siguiente sintaxis.

```
ALTER TABLE tabla_debil
ADD CONSTRAINT nombre_restricción
FOREIGN KEY (atributo_coincidencia1 [, atributo_coincidencia2, …] )
REFERENCES table_fuerte(atributo_coincidencia1 [,
atributo_coincidencia2, …]);
```

Se sugiere que `nombre_restricción` tenga la siguiente estructura: `fk_tabla_debil_tabla_fuerte`.

Una vez que tienes establecida una llave foránea apuntando a una tabla, puede decirse que ya estableciste relación ente las tablas.

Si por alguna razón requieres redefinir la llave foránea, primero tienes que eliminar la que tienes, y volverla a crear. Para remover una llave foránea, tendrías que aplicar `ALTER TABLE` con la siguiente sintaxis.

MySQL:

```
ALTER TABLE nombre_tabla
DROP FOREGIN KEY nombre_restricción;
```

SQL Server/Oracle/MS Access:

```
ALTER TABLE nombre_tabla
DROP CONSTRAINT nombre_restricción;
```

Es posible que en algún momento tengas la necesidad de cambiar de llave primaria. En ese caso, deberás promover una llave primaria para que cumpla con la tarea.

EJERCICIO 6.4: RESTRICCIONES DE LLAVE FORÁNEA

Escribe el código SQL para generar las llaves foráneas de todo el modelo de datos de ejemplo. Toma como base la siguiente tabla.

BD	Tabla	Campo	Tabla relacionada
BD	actor_pelicula	id_actor	actor
BD	actor_pelicula	id_pelicula	pelicula
BD	genero_pelicula	id_genero	genero
BD	genero_pelicula	id_pelicula	pelicula
BD	pelicula	id_pais	pais

FIGURA 6.4: *Tabla de llaves foráneas del modelo.*

Se asume una base de datos en MySQL. Recuerda la sintaxis básica:

```
alter table tabla_débil
add constraint fk_tabla_debil_tabla_fuerte
foreign key (lista_de_atributos_de_coincidencia)
references tabla_fuerte (lista_de_atributos_de_coincidencia);
```

El código quedaría así. Analízalo.

FILMES_TABLAS.SQL

```
-- Pone en uso la BD filmes.
use filmes;

-- Se generan las restricciones de llave foránea
alter table actor_pelicula
add constraint fk_actor_pelicula_actor
foreign key (id_actor)
references actor (id_actor);
```

```
alter table actor_pelicula
add constraint fk_actor_pelicula_pelicula
foreign key (id_pelicula)
references pelicula (id_pelicula);

alter table genero_pelicula
add constraint fk_genero_pelicula_genero
foreign key (id_genero)
references genero (id_genero);

alter table genero_pelicula
add constraint fk_genero_pelicula_pelicula
foreign key (id_pelicula)
references pelicula (id_pelicula);

alter table pelicula
add constraint fk_pelicula_pais
foreign key (id_pais)
references pais (id_pais);
```

6.6 Restricciones de unicidad (*unique*)

Esto obliga a que las llaves candidatas tengan, al inicio, una restricción que garantice su unicidad. Esto se logra usando `ALTER TABLE` con la siguiente sintaxis.

```
ALTER TABLE tabla_debil
ADD CONSTRAINT nombre_restricción
UNIQUE (campo1 [, campo2, …] );
```

Se sugiere que `nombre_restricción` tenga la siguiente estructura: `uc_campo`.

Si por alguna razón requieres remover una restricción de unicidad, tendrías que aplicar `ALTER TABLE` con la siguiente sintaxis.

MYSQL:
```
ALTER TABLE nombre_tabla
DROP INDEX nombre_restricción;
```

SQL SERVER / ORACLE O MS ACCESS:
```
ALTER TABLE nombre_tabla
DROP CONSTRAINT nombre_restricción;
```

6.7 Restricciones de valor por omisión (*default*)

Hay dos restricciones de valor que puede asignársele a los campos de una tabla: de valor por omisión (*default*), y de identidad, o *identity*.

La restricción de valor por omisión, o *default*, permite que, cuando no se proporciona un valor para un campo al momento de agregar un registro, se asuma un valor predeterminado.

Establecer una restricción de valor por omisión se logra usando `ALTER TABLE` con esta sintaxis.

MYSQL:
```
ALTER TABLE nombre_tabla
ALTER nombre_campo SET DEFAULT valor_por_omisión;
```

SQL SERVER:
```
ALTER TABLE nombre_tabla
ADD CONSTRAINT nombre_restricción
DEFAULT valor_por_omisión FOR nombre_campo;
```

MS ACCESS:
```
ALTER TABLE nombre_tabla
ALTER COLUMN nombre_campo SET DEFAULT valor_por_omisión;
```

ORACLE:
```
ALTER TABLE nombre_tabla
MODIFY nombre_campo DEFAULT valor_por_omisión;
```

Se sugiere que `nombre_restricción` tenga la siguiente estructura: `df_campo`.

Si por alguna razón se desea eliminar una restricción de valor por omisión, se puede hacer utilizando `ALTER TABLE` con la siguiente sintaxis.

MYSQL

```
ALTER TABLE nombre_tabla
ALTER nombre_campo DROP DEFAULT;
```

SQL SERVER / ORACLE / MS ACCESS

```
ALTER TABLE nombre_tabla
ALTER COLUMN nombre_campo DROP DEFAULT;
```

6.8 Restricciones de valor de auto incremento (*identity*)

La restricción de identidad, o *identity*, permite que un campo adquiera un valor numérico consecutivo, a partir de un número base, incrementándose una determinada cantidad cada vez. Al incremento se le llama *semilla*, o *seed*.

Establecer una restricción de identidad se logra al momento de crear la tabla usando `CREATE TABLE` añadiendo ciertas especificaciones al momento en que se definen los campos.

Cuando se define un campo como auto incrementable, al momento de agregar registros, no se debe proporcionar el valor por omisión, porque se asume que será proporcionado por el sistema.

MYSQL:

```
…
nombre_campo1 INT NOT NULL AUTO_INCREMENT;
…
```

SQL SERVER:

```
…
nombre_campo1 INT IDENTIY(número_inicial, semilla) PRIMARY KEY;
…
```

MS ACCESS:

```
…
nombre_campo1 AUTOINCREMENT PRIMARY KEY;
…
```

En Oracle, no se asocia el incremento a nivel campo. Se genera una secuencia, que es un contador de características globales, y se dispone del valor a discreción

.

ORACLE:

```
CREATE SEQUENCE nombre_secuencia
MINVALUE 1
START WITH 1
INCREMENT BY 1
CACHE 10;
```

Y cuando se desee utilizar el siguiente valor, se utiliza `nombre_secuencia.nextval`.

6.9 Métodos abreviados de creación de objetos

La codificación que se ha realizado hasta el momento ha sido muy precisa, y en los entornos de trabajo suele aplicarse de esa manera, porque es muy auditable: Crear los objetos y las restricciones una a una, permite revisar secuencias e identificar errores.

Si lo que quieres es simplificar el código al máximo, puedes seguir los siguientes consejos.

PRIMERO: Especifica a nivel campo todas las restricciones que puedas, al momento de crear la tabla.

Puedes agregar las siguientes:

`NOT NULL`

`UNIQUE`

`DEFAULT`

`IDENTITY`

SEGUNDO: Define las llaves primaria y foránea en la misma creación de la tabla.

La sintaxis podría ser de la siguiente manera.

```
CRATE TABLE tabla (
      campo1 tipo_dato NOT NULL AUTO_INCREMENT,
      campo2 tipo_dato NOT NULL UNIQUE,
      campo3 tipo_dato NOT NULL DEFAULT valor,
      PRIMARY KEY (campo1),
      FOREIGN KEY (campo2) REFERENCES table_fuerte(campo2)
);
```

EJERCICIO 6.5: CODIFICACIÓN ABREVIADA

Escribe el código para crear en la base de datos filmes una tabla llamada `película_especial` que es exactamente como película, pero con los siguientes supuestos:

a) `id_pelicula` es `int`, es la llave primaria, no acepta nulos, y es auto incrementable.

b) `estreno` es `date`, y admite nulos.

c) `id_pais` es `tinyint`, no admite nulos, tiene valor default 1, y es llave foránea que relaciona con `pais`, usando `id_pais`.

d) `ingresos` es `decimal (19,4)`, y admite nulos.

e) `nom_pelicula` es `varchar` de 100, no admite nulos y es valor único (no permite valores repetidos).

```
-- Usar la BD filmes.
use filmes;

-- Elimina la tabla, si existe.
drop table if exists pelicula_especial;

-- Genera la tabla, con todas las restricciones
-- al momento de creación.
create table pelicula_especial (
    id_pelicula int NOT NULL auto_increment,
    estreno date,
    id_pais tinyint NOT NULL DEFAULT 1,
    ingresos decimal(19,4),
    nom_pelicula nvarchar(100) NOT NULL unique,
    primary key (id_pelicula),
    foreign key (id_pais) references pais (id_pais)
);
```

7 MANIPULACIÓN BÁSICA DE DATOS

Todo el código y ejemplos vistos hasta este momento son estatutos que modifican la estructura de la base de datos, a lo que se conoce como *DDL* (*Data Definition Language*).

Ya que tenemos nuestra base de datos creada, toca el turno de almacenar datos y manipularlos, usando los estatutos conocidos como **DML**, o **Data Manipulation Language**, que son los que te permiten agregar, modificar o eliminar los datos en la base de datos.

Los principales estatutos DML son los siguientes:

1. **SELECT**. Permite recuperar datos almacenados en la base de datos.
2. **INSERT**. Permite agregar registros a las tablas de la base de datos.
3. **UPDATE**. Permite modificar los datos almacenados en los registros de la base de datos.
4. **DELETE**. Permite eliminar registros almacenados en la base de datos.

Te explico cada uno de los estatutos con ejemplos, para que puedas comprobar las capacidades del modelo de datos que has creado.

7.1 Agregado de registros, usando INSERT

Lo primero que hay que hacer es agregar registros. Esto tiene una explicación lógica: requerimos tener datos para explorar el resto de los estatutos DML, así que agregar datos es lo primero que debe suceder.

El estatuto `INSERT INTO` es el que te permite agregar registros a una tabla. Internamente, este estatuto hace lo siguiente: genera un registro temporal, una sala de espera para los datos, por así decirlo; después,

acepta un conjunto de datos, verifica que los tipos sean los correctos, y que todas las reglas y restricciones se respetan, y si no hay problema con los datos, los traslada de forma definitiva a la base de datos para que tengan un almacenamiento permanente.

Si quieres agregar un registro proporcionando un valor a cada campo de la tabla, sin omitir ninguno, puedes utilizar esta sintaxis:

```
INSERT INTO tabla VALUES (ValorCampo1, ValorCampo2,…);
```

En esta sintaxis, los valores proporcionados se especifican siguiendo el orden en que se encuentran los campos en la estructura de la tabla.

Si quieres agregar un registro proporcionando valor a ciertos campos de la tabla, pero no a todos, debes utilizar esta sintaxis:

```
INSERT INTO tabla (Campo1, Campo2,…)
VALUES (ValorCampo1, ValorCampo2,…);
```

En esta sintaxis, los valores proporcionados se especifican siguiendo el orden en que se encuentran en la especificación de campos a actualizar (primer paréntesis de la sintaxis).

Esta sintaxis es requerida en aquellos casos que se tienen campos auto incrementables, o cuando por alguna razón sólo se desea agregar valores en ciertos campos, de inicio. También puede ser utilizada cuando se desea actualizar todos los campos: esta sintaxis funciona para todos los casos; de hecho, es común ver que sólo se utiliza esta sintaxis, debido a que, si por alguna razón extraña decides cambiar el orden de los campos en la estructura de la tabla, tus programas seguirán funcionando.

Cuando utilizas `INSERT INTO`, el registro se agrega satisfactoriamente siempre y cuando los valores a agregar correspondan a los tipos de datos definidos en la estructura, si se respetan

satisfactoriamente las restricciones `UNIQUE`, `NOT NULL`, `PRIMARY KEY` y `FOREIGN KEY`; si algo anda mal, se producirá un error.

Aunque por definición `INSERT INTO` sirve para agregar registros a una tabla, realmente agrega un registro a la base de datos. ¿Qué quiero decir con eso? Pues que no solo se tienen que respetar las restricciones de la tabla, sino del modelo de datos. Te pongo un ejemplo: si tienes dos tablas relacionadas, y quieres agregar un registro en la tabla débil, lo más probable es que el valor que quieras poner en los campos de coincidencia debe estar previamente registrados en la tabla fuerte, o provocarás una violación a la integridad referencial. Esto quiere decir que el agregado de registros tiene un orden. Si a una base de datos vacía le cargas datos, debes, comenzar por las tablas fuertes, y luego con las débiles.

Al insertar información, toma en cuenta estas reglas:

a) Los datos numéricos van sin paréntesis.
b) Los datos de tipo cadena van entre comilla simple sin inclinación.
c) Los datos de tipo fecha se especifican en un formato `YYYY-mm-dd`, entre comillas.

Estas reglas pueden variar, dependiendo la configuración del motor de base de datos.

EJERCICIO 7.1: CARGANDO INFORMACIÓN A LA BASE DE DATOS

Escribe el código para cargar información en la base de datos.

Asume que se utiliza MySQL. Revisa los comentarios.

```
-- Usar BD filmes.
use filmes;

-- Se agregan primero los registros a las tablas fuertes
-- o de catálogo. Esto garantiza que en las relaciones
-- entre tablas, las restricciones de llave foránea
-- no indiquen falta de integridad referencial.
-- Primero, las que son fuertes, en todas las relaciones.
-- Segundo, las que son fuertes y débiles en algunas relaciones.
-- Tercero, las que son débiles en todas las relaciones.

-- Tabla genero.
-- Es tabla fuerte en todas las relaciones.
-- Agregado de registro cuando se proporcionan todos los campos
-- (método abreviado, no recomendado).

insert into genero values (1,'Action');

-- Método estándar, que funciona siempre, y que no se afecta
-- en caso de cambios de posición en los campos (recomendado).

insert into genero (id_genero, nom_genero)
values (2,'Adventure');
insert into genero (id_genero, nom_genero)
values (3,'Crime');
insert into genero (id_genero, nom_genero)
values (4,'Drama');
insert into genero (id_genero, nom_genero)
values (5,'International');
insert into genero (id_genero, nom_genero)
values (6,'Terror');
insert into genero (id_genero, nom_genero)
values (7,'Thriller');
```

```
-- Tabla actor.
-- Es tabla fuerte en todas las relaciones.

insert into actor (id_actor, nom_actor)
values (2,'Leonardo DiCaprio');
insert into actor (id_actor, nom_actor)
values (3,'Christian Bale');
insert into actor (id_actor, nom_actor)
values (4,'Steve Carell');
insert into actor (id_actor, nom_actor)
values (5,'Pedro Infante');
insert into actor (id_actor, nom_actor)
values (6,'Gary Oldman');
insert into actor (id_actor, nom_actor)
values (7,'Natalie Portman');

-- Tabla pais.
-- Es tabla fuerte en todas las relaciones.

insert into pais (id_pais, nom_pais)
values (1,'United States');
insert into pais (id_pais, nom_pais)
values (2,'México');
insert into pais (id_pais, nom_pais)
values (3,'Canada');

-- Tabla pelicula.
-- Es tabla fuerte en algunas relaciones, y es débil
-- en algunas relaciones.

insert into pelicula (id_pelicula, estreno, id_pais,
ingresos, nom_pelicula)
values (28929,'2015-12-16',3,532950503,'Revenant');
insert into pelicula (id_pelicula, estreno, id_pais,
ingresos, nom_pelicula)
values (28930,'2015-12-11',1,133440870,'The Big Short');
insert into pelicula (id_pelicula, estreno, id_pais,
ingresos, nom_pelicula)
values (28931,'1953-08-21',2,null,'Pepe El Toro');
insert into pelicula (id_pelicula, estreno, id_pais,
ingresos, nom_pelicula)
values (28932,'1994-09-14',1,19552639,'The professional');
insert into pelicula (id_pelicula, estreno, id_pais,
ingresos, nom_pelicula)
values (28933,'2000-04-14',1,34266564,'American Psycho');
```

```
-- Tabla actor película.
-- Es tabla débil en todas las relaciones.

insert into actor_pelicula (id_actor, id_pelicula)
values (2,28929);
insert into actor_pelicula (id_actor, id_pelicula)
values (3,28930);
insert into actor_pelicula (id_actor, id_pelicula)
values (4,28930);
insert into actor_pelicula (id_actor, id_pelicula)
values (5,28931);
insert into actor_pelicula (id_actor, id_pelicula)
values (6,28932);
insert into actor_pelicula (id_actor, id_pelicula)
values (7,28932);
insert into actor_pelicula (id_actor, id_pelicula)
values (3,28933);

-- Tabla genero pelicula.
-- Es tabla débil en todas las relaciones.

insert into genero_pelicula (id_genero, id_pelicula)
values (1,28929);
insert into genero_pelicula (id_genero, id_pelicula)
values (2,28929);
insert into genero_pelicula (id_genero, id_pelicula)
values (4,28930);
insert into genero_pelicula (id_genero, id_pelicula)
values (7,28930);
insert into genero_pelicula (id_genero, id_pelicula)
values (4,28931);
insert into genero_pelicula (id_genero, id_pelicula)
values (1,28932);
insert into genero_pelicula (id_genero, id_pelicula)
values (3,28932);
insert into genero_pelicula (id_genero, id_pelicula)
values (4,28932);
insert into genero_pelicula (id_genero, id_pelicula)
values (5,28932);
insert into genero_pelicula (id_genero, id_pelicula)
values (3,28933);
insert into genero_pelicula (id_genero, id_pelicula)
values (6,28933);
insert into genero_pelicula (id_genero, id_pelicula)
values (7,28933);
```

```
-- Estas dos líneas causan error.
-- Esta línea trata de agregar un registro que ya existe.
-- Como no puede haber dos registros con la misma llave
-- primaria, se produce un error.

insert into genero (id_genero, nom_genero)
values (2,'Adventure');

-- Esta línea trata de agregar un registro en genero_pelicula,
-- pero el género 20 no está en la tabla genero, y se
-- produce una violación de integridad referencial.
-- Sólo se pueden agregar valores que existen
-- en las respectivas tablas fuertes en la relación.

insert into genero_pelicula (id_genero, id_pelicula)
values (20,28933);
```

7.1 Expresiones lógicas

Una *expresión lógica*, también llamado *condición*, *condición lógica* o *predicados*, es una expresión que resuelve falso o verdadero, como producto de aplicar operadores de comparación y valores.

Algunos operadores de comparación clásicos son:

Operador	Nombre
`>`	Mayor que
`<`	Menor que
`>=`	Mayor o igual que
`<=`	Menor o igual que
`=`	Igual a
`<>`	Diferente a
`NOT NULL`	No es nulo.
`BETWEEN`	Valor entre uno y otro valor.
`LIKE`	Comparación de cadenas basada en patrones. Al formar patrones, % indica “el resto”, y _ indica una sola posición.

Cuando la expresión lógica sólo se compone de una condición, se dice que es una *condición simple*; cuando la expresión lógica se compone de 2 o más condiciones, se dice que es una *condición compuesta*.

Las condiciones complejas utilizan *operadores lógicos*, que permiten resolver condiciones en grupo.

Los operadores lógicos típicos son los siguientes:

Operador	Nombre
`AND`	**Conjuntivo.** Con una condición que sea falsa, todo es falso.
`OR`	**Disyuntivo.** Con una condición que sea verdadera, todo es verdadero.
`NOT`	**Negación.** Cambia falso en verdadero, y verdadero en falso.

Las expresiones se resuelven por pares, de izquierda derecha; si se desea modificar ese comportamiento, se pueden utilizar los paréntesis, para obligar a que las condiciones se resuelvan de determinada manera.

Estos ejemplos pueden ilustrarte el uso de los operadores.

```
id_producto = '28892'
```

Retorna **`verdadero`** sólo si **`id_producto`** es 28892.

```
sueldo >= 25000
```

Retorna **`verdadero`** sólo si **`sueldo`** es mayor a 25000.

```
tipo_producto = 25 AND precio >= 5000
```

Retorna verdadero sólo si **tipo_producto** es 25 y **precio** es mayor o igual a 5000; con una de las dos condiciones que no se cumpla, retornará **falso**.

```
tipo_producto = 25 OR precio >= 5000
```

Retorna verdadero si **tipo_producto** es 25 o si **precio** es mayor o igual a 5000; con una de las dos condiciones que se cumpla, retornará **verdadero**.

```
fecha_ingreso NOT NULL
```

Retorna verdadero si **fecha_ingreso** no es nulo.

```
NOT id_producto = '28892'
```

Retorna **verdadero** sólo si **id_producto** no es 28892.

```
tipo_producto = 25 AND precio >= 5000 OR estado = 1
```

Retorna **verdadero** si **tipo_producto** es 25 y **precio** es mayor o igual a 5000; o bien, si **estado** es 1.

```
tipo_producto = 25 AND (precio >= 5000 OR estado = 1)
```

Retorna **verdadero** si **tipo_producto** es 25 y si **precio** es mayor o igual a 5000 o **estado** sea 1.

```
precio BETWEEN 2000 AND 5000
```

Retorna **verdadero** si **precio** es mayor o igual a 2000 o menor o igual a 5000.

```
nom_ciudad LIKE 'AU%'
```

Retorna **verdadero** si **nom_ciudad** inicia con «AU».

```
nom_ciudad LIKE '%ON'
```

Retorna **verdadero** si **nom_ciudad** termina con «ON».

```
estado LIKE '___'
```

Retorna **verdadero** si **estado** es una cadena de 3 posiciones.

```
nom_país LIKE 'M_XICO'
```

Retorna **verdadero** si **nom_país** es «M XICO», pero la segunda letra puede ser cualquiera. Este recurso es muy útil cuando se desean incluir condiciones que pueden tener o no tener acento.

Las expresiones lógicas son muy comunes, a través de la cláusula **WHERE**, que se utiliza para indicar a qué registros les aplican las afectaciones de una instrucción: si la condición resuelve por verdadero, los registros son afectados.

7.2 Modificando de registros, usando UPDATE

El estatuto `UPDATE` es el estatuto que te permite modificar los datos registrados en la base de datos. Al igual que `INSERT INTO`, acepta un conjunto de datos, verifica que los tipos sean los correctos, y que todas las reglas y restricciones se respetan, y si no hay problema con los datos, los traslada de forma definitiva a la base de datos para que tengan un almacenamiento permanente.

Al estatuto `UPDATE` se le tiene que indicar a qué tabla se le desean actualizar los datos, se tienen que especificar asignaciones de valores a nivel campo, y en algunos casos, una condición que actúa como filtro.

Se le llama *actualización global* cuando se desea la modificación de los valores de los campos, sin omitir ningún registro. La sintaxis es la siguiente:

```
UPDATE tabla SET Campo1=NuevoValor1, Campo2=NuevoValor2,…;
```

Con esta sintaxis, todos los registros tendrán en el `Campo1`, el valor `NuevoValor1`.

Se le llama *actualización específica* cuando se desea la modificación de los valores, siempre y cuando se cumpla con una expresión lógica. La sintaxis es la siguiente:

```
UPDATE tabla SET Campo1=NuevoValor, Campo2=NuevoValor,…
WHERE ExpresiónLógica;
```

Con esta sintaxis, los registros tendrán en el `Campo1`, el valor `NuevoValor1`, únicamente si se cumple la `ExpresiónLógica`. En bases de datos, un error clásico es omitir la cláusula `WHERE` en un `UPDATE`. Verifícalo siempre, porque si no lo haces, los cambios pueden ser irreversibles.

Es muy común modificar campos que no participan en la llave primaria, ni que son llaves foráneas. Modificar llaves foráneas es inusual,

pero posible, mientras que modificar atributos primos es extremadamente inusual.

Al modificar información, toma en cuenta estas reglas:

d) Los datos numéricos van sin paréntesis.
e) Los datos de tipo cadena van entre comilla simple sin inclinación.
f) Los datos de tipo fecha se especifican en un formato `YYYY-mm-dd`, entre comillas.

Estas reglas pueden variar, dependiendo la configuración del motor de base de datos.

EJERCICIO 7.2: MODIFICANDO DATOS

Escribe el código para actualizar las descripciones de la base de datos a mayúsculas. Apóyate en la función `UPPER()`.

También modifica el campo ingresos de la tabla `pelicula`, asignando cero cuando sea nulo.

Posteriormente, actualiza el campo `ingresos` de la tabla `pelicula`, asignando el valor de $230,000,000.00, y cambiando «PEPE EL TORO» por «PEPE EL TORO (B/N)», cuando `id_pelicula` sea 28931.

Asume que se utiliza MySQL. Revisa los comentarios.

```
-- Usar BD filmes.
use filmes;

-- Pasar todas las descripciones a mayúsculas.
-- Son actualizaciones globales, y por tanto, no tienen
-- cláusula WHERE.
update actor set nom_actor = upper(nom_actor);
update genero set nom_genero = upper(nom_genero);
update pais set nom_pais = upper(nom_pais);
update pelicula set nom_pelicula = upper(nom_pelicula);
```

```
-- Actualiza pelicula.ingresos con cero, cuando sea nulo
update pelicula set ingresos=0 where ingresos is null;

-- Actualiza pelicula.ingresos con $230,000.000.00,
-- y pelicula.nom_pelicula con PEPE EL TORO (B/N), cuando
-- id_pelicula sea 28931.
update pelicula set ingresos=230000000.00,
nom_pelicula='PEPE EL TORO (B/N)'
where id_pelicula=28931;
```

7.3 Eliminando registros, usando DELETE

El estatuto **`DELETE FROM`** es el estatuto que te permite eliminar registros de la base de datos.

Al estatuto **`DELETE FROM`** se le tiene que indicar de qué tabla se desean eliminar registros, y en algunos casos, una condición que actúa como filtro.

Se le llama *eliminación global* cuando se desea la eliminación de todos los registros, sin omitir ninguno. La sintaxis es la siguiente:

```
DELETE FROM tabla;
```

Con esta sintaxis, todos los registros de la tabla se eliminarán.

Se le llama *eliminación específica* cuando se desea la eliminación de los registros, siempre y cuando cumplan con una condición. La sintaxis es la siguiente:

```
DELETE FROM tabla WHERE ExpresiónLógica;
```

Con esta sintaxis, los registros que cumplan con la **`ExpresiónLógica`** serán eliminados.

EJERCICIO 7.3: MODIFICANDO DATOS

Escribe el código para eliminar todos los registros de **`pelicula_especial`**.

Agrega el género comedia en **`genero`** con el **`id_genero`** 9, «COMEDIA», y luego elimínalo.

Intenta eliminar un registro de **`pais`**, y comprueba que no es posible si tiene registros asociados en **`pelicula`**.

Asume que se utiliza MySQL. Revisa los comentarios.

```
-- Usar BD filmes.
use filmes;

-- Elimina todos los registros de una tabla.
delete from pelicula_especial;

-- Agrega un registro, para comprobar la eliminación.
insert into genero values(9,'COMEDIA');
-- Muestra el resultado.
select * from genero where id_genero=9;

-- Elimina el registro que acabas de crear.
delete from genero where id_genero=9;
-- Muestra el resultado.
select * from genero where id_genero=9;

-- Intenta eliminar un registro que tiene registros de coincidencia
-- asosciados. Genera error.
delete from pais where id_pais = 1;
```

7.4 Consultando información usando SELECT

El estatuto `SELECT` es el estatuto que te permite realizar consultas. Una *consulta* (*query*) es una recuperación de información a partir de la base de datos, que puede ser una consulta escalar, si recupera un solo dato, o una consulta tabular, que retorna arreglos bidimensionales de datos.

La sintaxis básica de SELECT es la siguiente:

```
SELECT lista_campos FROM tabla;
```

Si la lista de campos es un `*`, quiere decir que son todos los campos de la tabla.

```
SELECT * FROM tabla;
```

Si la lista de campos es selectiva, solo aparecerán los campos que se señalen. La sintaxis es la siguiente:

```
SELECT Campo1, Campo2,… FROM tabla;
```

Si la lista de campos es selectiva y además no queremos que el resultado de la consulta revele el nombre real del campo, se puede agregar un alias, usando la cláusula `as`. La sintaxis es la siguiente:

```
SELECT Campo1 as 'Alias', Campo2 as 'Alias',… FROM tabla;
```

Si deseamos aplicar un filtro a la consulta podemos hacerlo con la cláusula `WHERE`. La sintaxis es la siguiente.

```
SELECT lista_campos FROM tabla
WHERE ExpresiónLógica;
```

Si deseamos aplicar ordenamiento a la consulta podemos hacerlo con la cláusula `ORDER BY`. La sintaxis es la siguiente.

```
SELECT lista_campos FROM tabla
ORDER BY Campo [asc/desc];
```

Si deseamos mostrar campos de más de una tabla, podemos aprovechar la relación entre tablas que definimos en el modelo, usando la cláusula `INNER JOIN`. La sintaxis es la siguiente.

```
SELECT lista_campos
FROM tabla_debil
INNER JOIN table_fuerte
ON table_debil.atributo_coinc = tabla_fuerte.atributo_coinc;
```

En caso de que desees realizar más de una relación, puedes agregar una después de la otra.

El código puede llegar a ser muy complejo, pero puede simplificarse colocando alias para las tablas. La sintaxis quedaría así.

```
SELECT lista_campos
FROM tabla_debil td
INNER JOIN table_fuerte tf
ON td.atributo_coinc = tf.atributo_coinc;
```

Todos los elementos de una consulta pueden convivir entre sí, es decir, que puedes mezclar `INNER JOIN` con `WHERE` con `ORDER BY` siempre y cuando respetes ese orden.

EJERCICIO 7.4: MODIFICANDO DATOS

Desarrolla las siguientes consultas, y observa los resultados.

Asume que se utiliza MySQL. Revisa los comentarios.

```
-- Usar BD filmes.
use filmes;

-- Consulta básica.
-- Muestra todos los campos y todos los registros
-- de país.
select * from pais;
```

	id_pais	nom_pais
▶	1	UNITED STATES
	2	MÉXICO
	3	CANADA
*	NULL	NULL

```
-- Consulta con campos selectivos.
-- Muestra el nombre de película, fecha de estreno,
-- e ingesos totales.
select nom_pelicula, estreno, ingresos
from pelicula;
```

	nom_pelicula	estreno	ingresos
▶	REVENANT	2015-12-16	532950503.0000
	THE BIG SHORT	2015-12-11	133440870.0000
	PEPE EL TORO (B/N)	1953-08-21	230000000.0000
	THE PROFESSIONAL	1994-09-14	19552639.0000
	AMERICAN PSYCHO	2000-04-14	34266564.0000

```
-- Consulta con campos selectivos y alias de campo.
-- Muestra el nombre de pelicula, fecha de estreno e ingresos
-- sin revelar los nombres reales de los campos.
select nom_pelicula as 'Nombre', estreno as 'Fecha de estreno',
ingresos as 'Ingresos totales' from pelicula;
```

Nombre	Fecha de estreno	Ingresos totales
REVENANT	2015-12-16	532950503.0000
THE BIG SHORT	2015-12-11	133440870.0000
PEPE EL TORO (B/N)	1953-08-21	230000000.0000
THE PROFESSIONAL	1994-09-14	19552639.0000
AMERICAN PSYCHO	2000-04-14	34266564.0000

```
-- Consulta con filtro.
-- Mostrar el nombre de la película y la fecha de estreno
-- de aquellas películas cuyos ingresos sean superiores
-- a los 200 millones.
select nom_pelicula, estreno from pelicula
where ingresos>200000000;
```

nom_pelicula	estreno
REVENANT	2015-12-16
PEPE EL TORO (B/N)	1953-08-21

```
-- Consulta con datos ordenados.
-- Mostrar el nombre de la pelicula, y los ingresos, ordenado
-- por los ingresos, en orden descendente.
select nom_pelicula, ingresos from pelicula
order by ingresos desc;
```

nom_pelicula	ingresos
REVENANT	532950503.0000
PEPE EL TORO (B/N)	230000000.0000
THE BIG SHORT	133440870.0000
AMERICAN PSYCHO	34266564.0000
THE PROFESSIONAL	19552639.0000

```
-- Consulta con datos ordenados por más de un campo.
-- Mostrar el nombre de la pelicula y el id del país
-- Ordenando de forma descendente por id del país, y
-- ascendente por el nombre de la película.
select nom_pelicula, id_pais from pelicula
order by id_pais desc, nom_pelicula asc;
```

	nom_pelicula	id_pais
▶	REVENANT	3
	PEPE EL TORO (B/N)	2
	AMERICAN PSYCHO	1
	THE BIG SHORT	1
	THE PROFESSIONAL	1

```
-- Consulta de más de una tabla, relacionadas.
-- Muestra el nombre de película y de país,
-- considera que el nombre del país no está
-- en la tabla pelicula, pero puede estar
-- disponible a través del valor de id_pais.
select pelicula.nom_pelicula, pais.nom_pais
from pelicula
inner join pais on pelicula.id_pais = pais.id_pais;
```

	nom_pelicula	nom_pais
▶	REVENANT	CANADA
	THE BIG SHORT	UNITED STATES
	PEPE EL TORO (B/N)	MÉXICO
	THE PROFESSIONAL	UNITED STATES
	AMERICAN PSYCHO	UNITED STATES

```
-- Consulta de múltiples tablas relacionadas.
-- Muestra el nombre de película y del género,
-- considera que el nombre de la película y del género
-- no están en la tabla de relación, pero puenden ser
-- obtenidas a partir de las llaves, relacionando
-- con las tablas fuertes.
select pelicula.nom_pelicula, genero.nom_genero
from genero_pelicula
inner join pelicula on
genero_pelicula.id_pelicula=pelicula.id_pelicula
inner join genero on genero_pelicula.id_genero=genero.id_genero;
```

nom_pelicula	nom_genero
REVENANT	ACTION
REVENANT	ADVENTURE
THE BIG SHORT	DRAMA
THE BIG SHORT	THRILLER
PEPE EL TORO (B/N)	DRAMA
THE PROFESSIONAL	ACTION
THE PROFESSIONAL	CRIME
THE PROFESSIONAL	DRAMA
THE PROFESSIONAL	INTERNATIONAL
AMERICAN PSYCHO	CRIME
AMERICAN PSYCHO	TERROR
AMERICAN PSYCHO	THRILLER

```
-- El mismo query que el anterior, con alias de tabla.
select pl.nom_pelicula, gn.nom_genero
from genero_pelicula gp
inner join pelicula pl on gp.id_pelicula=pl.id_pelicula
inner join genero gn on gp.id_genero=gn.id_genero;
```

```
-- Consulta con todos los elementos juntos.
-- Consulta selectiva de campos
-- con alias de campo
-- con alias de tabla
-- con doble inner join
-- con filtro de registros
-- el país NO debe ser 1
-- ordenado por nombre de género, ascendente
select
      pl.nom_pelicula as 'Pelicula' ,
      gn.nom_genero as 'Género cinematográfico'
from genero_pelicula gp
inner join pelicula pl
on gp.id_pelicula=pl.id_pelicula
inner join genero gn
on gp.id_genero=gn.id_genero
where not pl.id_pais=1
order by gn.nom_genero asc;
```

	Pelicula	Género cinematográfico
▶	REVENANT	ACTION
	REVENANT	ADVENTURE
	PEPE EL TORO (B/N)	DRAMA

8 EJERCICIO INTEGRADOR

Una forma de probar los nuevos conocimientos que has adquirido es haciendo un modelo de datos que permita soportar análisis de inteligencia de negocios.

Si logras hacer este ejercicio por ti mismo, creo que estarás bastante preparado para enfrentar proyectos de modelación de complejidad que podría calificarse como media – alta.

Como podrás darte cuenta, el ejercicio no está del todo resuelto: la idea es que lo resuelvas tú de manera fina, y que compruebes las habilidades que has adquirido.

Te relataré el ejercicio como si fuera una historia en la cual tú serás quien llene los espacios en blanco.

8.1 Entiende a tu cliente

Tener una red de contactos profesional para hacer *networking* es siempre una buena idea.

En lo personal, yo formé mi propia red de contactos en **LinkedIn** con un grupo de amigos que son especialistas en diferentes cosas. Cuando hay una oportunidad de trabajo, le pasamos el dato a quien mejor está calificado para atenderlo, y todos ganamos.

Un amigo mío me pasó el contacto del nuevo director comercial de una empresa llamada **Shoe Land**, que según me dice, tiene una necesidad de implementar un tablero de datos que le facilite la toma de decisiones.

Acuerdo una cita con el director comercial, para hacer el levantamiento de requerimientos. Antes de acudir a la cita, hago mi

trabajo de investigación: reviso en internet el web site de la compañía, para saber a qué se dedica, qué hacen, y no llegar en blanco.

Resulta que es una cadena de zapaterías que inició operaciones hace 15 años, en el Estado de Guanajuato, en México; gracias a sus diseños y calidad, tuvo una expansión bastante grande, lo que permitió poner sucursales de venta al público en México y en Estados Unidos.

Por lo que pude investigar, la empresa comercializa calzado de diferentes tipos: deportivo, de trabajo, casual y formal; gracias a que ha manejado su estrategia de marca adecuadamente, de tal manera que sus modelos tienen un nombre reconocido comercialmente entre sus clientes, a nivel internacional. La mayor parte de sus ingresos son gracias a las ventas en Estados Unidos, donde los zapatos son muy apreciados por sus acabados artesanales. Los ingresos anuales de la compañía son del orden de los $75 millones de pesos.

Entender a qué se dedica el cliente y cuál es la magnitud de sus operaciones puede darte una idea del volumen de datos que maneja, y te da la oportunidad de prepararte un poco para no llegar a la primera cita con el cliente sin saber nada de su negocio y de la industria en la que participa.

El web site, aunque contiene buena información, se ve anticuado; tienen buena presencia en Facebook, pero no en Instagram ni en Twitter. Concluyo que la tecnología no ha sido una prioridad, lo que también me dice que el cambio de director comercial puede tener su origen en una renovación de la administración y que quieren modernizarse.

8.2 Analiza los requerimientos

Me reúno con el nuevo director comercial, y a la reunión también acude una persona del área de desarrollo de sistemas; la persona de desarrollo me platica que tienen un sistema desarrollado *ad-hoc* para la empresa, que controla todas las operaciones, a través de un portal web, donde las sucursales pueden registrar las ventas y las operaciones. La

base de datos del sistema está en **Microsoft SQL Server**, pero no está preparada para que se ejecuten operaciones de inteligencia de negocios en ella.

La idea es que se diseñe una base de datos que será alimentada de datos a través de un proceso nocturno, fuera de horario de operaciones. Se espera que la base de datos esté implementada en **MySQL**, en un servidor independiente.

Me interesa esa información, desde luego, pero me interesa más saber qué es lo que quiere el director comercial, así que le pregunto qué es lo que requiere.

El director comercial me explica que le llama la atención que no exista un *tablero de datos* (*dashboard*) que le permita monitorear el comportamiento de las ventas, y que quiere uno.

Lo que el nuevo director desea es contar con una herramienta de inteligencia de negocios que le permita lo siguiente:

a) Analizar los montos de ventas.
b) Analizar las unidades vendidas.
c) Analizar la utilidad obtenida.
d) Analizar las prácticas de descuentos.

La información debe tener las siguientes características.

a) Los montos deben estar en pesos mexicanos, en todos los casos.
b) La información se analiza, antes de impuestos.
c) Para el tema de las utilidades, sólo se deben considerar los costos de producción en fábrica, dejándose fuera otros costos, como los de traslado o de almacenaje.

La finalidad última de la base de datos es proveer de datos a un tablero de datos, y como todo tablero de datos, el ordenamiento y filtrado

de los datos por diferentes criterios es parte importante de la herramienta. Los datos deben poder segmentarse atendiendo los siguientes criterios.

a) Temporalidad (*año*, *mes*).
b) Tipo de calzado (*Deportivo*, *Trabajo*, *Casual*, *Formal*, y así).
c) Moneda de la operación (*Pesos* Mexicanos, *Dólares* Americanos).
d) Geográfica (*Ciudad*, *Estado*, *País*).
e) Modelo (*Verona*, *AGILE*, *Excelsior*, y así).
f) Tamaño (*22MX*, *22USA*, *22.5MX*, *22.5USA*, y así).
g) Sucursal (*Roma Plaza*, *Inter Plaza*, *Galerías Monterrey*, *River Walk*, y así).
h) Tipo de sucursal (*Estándar*, *Premium*, *Outlet*, etc.)

Me piden que diseñe e implemente una base de datos que soporte un tablero de datos; el *dashboard* será desarrollado en **Microsoft Power BI** y la idea es que, para todas las tareas de inteligencia de negocios, se trabaje con una base de datos diferente a las de producción de la compañía; el área de desarrollo de la empresa se encargará de elaborar la interfaz de datos para poblar la base que yo desarrolle. Sólo me debo preocupar de contar, al final, con un modelo de datos que cubra todos los requerimientos de información y segmentación que el nuevo director comercial requiere para informarse y tomar decisiones.

Sólo hay dos supuestos bajo los cuales el modelo de datos para el tablero no sería posible de implementar:

a) La información que se necesita no está disponible en los sistemas de la organización.
b) Obtener la información requerida es demasiado costoso y no se justifica financieramente.

Revisamos rápido los requerimientos, la información que necesito para mi base, y al parecer no hay problemas con lo que queremos hacer.

La reunión es en la mañana. Esa misma tarde elaboro un documento descriptivo, simple pero claro, llamado «*Entendimiento de requerimientos*», en donde describo los requerimientos tal y como yo los he entendido.

Hacer un documento de entendimiento de requerimientos y validarlo con el cliente es esencial, puesto que si entiendes mal los requerimientos aplicarás esfuerzos en algo que nadie pidió, e incluso, puede ser que tengas que retrabajar, y eso siempre es una pérdida irremediable de tiempo. Es como construir una casa, sin validar antes el diseño y los planos con el cliente: una vez que comienzas a construir, las modificaciones serán difíciles y costosas de realizar.

Le mando el documento al cliente de la información, para que me confirme si entendí bien todo lo que necesita; dos días después, me dicen que sí está bien, y me autorizan el proyecto, así que les mando mi propuesta económica, y exijo firmar un acuerdo de confidencialidad, que los protege a ellos, y a mí me autoriza a conocer sus sistemas y si información, sin incurrir en un delito informático.

Una semana después me mandan la orden de compra, firmamos el acuerdo de confidencialidad, y me pongo a trabajar sobre el modelo.

8.3 Enfócate en el fenómeno

Una vez que tienes claro los requerimientos y que los has validado con el cliente final de la información, marca tu distancia respecto a los requerimientos y enfócate en los fenómenos que necesitas observar para poder satisfacer los requerimientos.

Evita caer en la tentación de atender sólo lo que te dice la persona con la que hablas, pues puede tener una visión sesgada de la realidad; es probable que esa persona se vaya, que cambie sus requerimientos conforme conozca más de la operación, o que cambie de rol dentro de la misma organización; ya sea que la persona modifique sus necesidades, o

que llegue otra persona, seguramente habrán nuevos requerimientos respecto a la forma o la cantidad, pero el fenómeno seguirá siendo el mismo. No aplica eso de que "haz lo que te piden": piensa como un médico, haz un diagnóstico acertado, receta lo correcto. Nadie llega con el médico y le dice "quiero tener esta enfermedad, y quiero que me recete esto, deme la receta y ya". Tú cliente dice *qué quiere*, pero tú eres el especialista que estará encargado de definir el *cómo* más apropiado.

En el caso del ejemplo, nos podemos dar cuenta que se desea hacer una herramienta de análisis de ventas y utilidades.

Debemos preguntarnos entonces ¿qué fenómenos en la organización me permitirían conocer la información de ventas y utilidades? La respuesta es clara: la venta de calzado en sucursal es el fenómeno que nos interesa, junto con los costos de producción.

La siguiente pregunta es, ¿hay algún registro existente en la organización que sea evidencia de ese fenómeno? La evidencia puede ser un sistema de información existente, un documento, e incluso leyes que especifican información que es necesario tener a la mano.

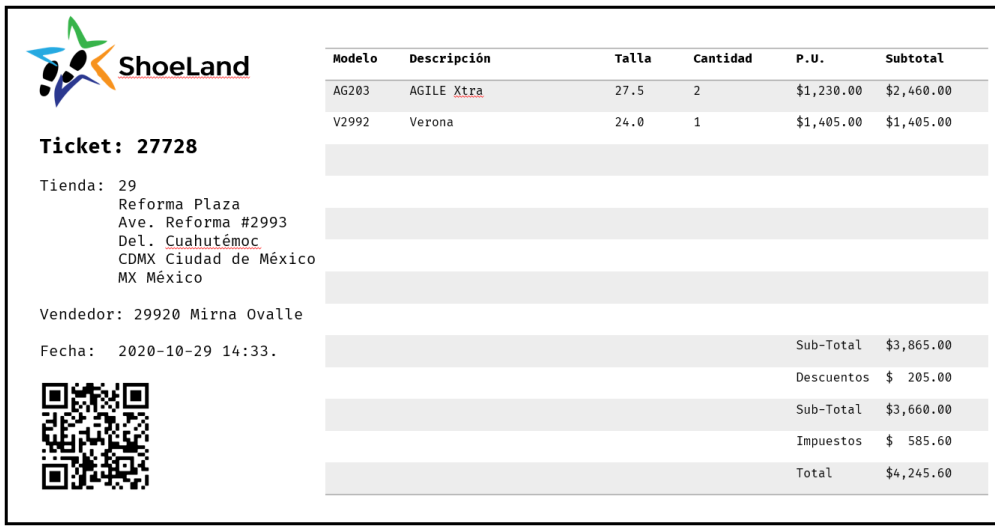
En nuestro ejemplo, la información que mejor representa el fenómeno de la venta en sucursales es el *ticket de venta*, y el sistema de producción.

8.4 La documentación como fuente

Lo sistemas de información y los documentos transaccionales son una gran ayuda para comenzar a identificar la información relevante, así como la estructura de datos.

Cada descripción corresponde a una etiqueta, cada número es una clave o un monto, y comienza un ejercicio de imaginación donde vamos suponiendo datos, tipos y longitudes.

El típico ticket de ShoeLand, es así.



ShoeLand

Ticket: 27728

Tienda: 29
Reforma Plaza
Ave. Reforma #2993
Del. Cuahutémoc
CDMX Ciudad de México
MX México

Vendedor: 29920 Mirna Ovalle

Fecha: 2020-10-29 14:33.

Modelo	Descripción	Talla	Cantidad	P.U.	Subtotal
AG203	AGILE Xtra	27.5	2	$1,230.00	$2,460.00
V2992	Verona	24.0	1	$1,405.00	$1,405.00
				Sub-Total	$3,865.00
				Descuentos	$ 205.00
				Sub-Total	$3,660.00
				Impuestos	$ 585.60
				Total	$4,245.60

A partir del ticket, podemos inferir algunas cosas que sin duda alguna deberemos corroborar, pero como principio es bueno. Algunas conclusiones son las siguientes.

a) El número de ticket es numérico.
b) El ticket es expedido en una sucursal.
c) La sucursal tiene un nombre.
d) La sucursal tiene un domicilio.
e) La sucursal está en una ciudad, y sabemos que la ciudad está en un estado, y el estado en un país.
f) El ticket es realizado por un vendedor, cuyo código de identificación es numérico.
g) El ticket se realiza en una fecha y una hora específica.
h) Un ticket puede amparar la compra de más de un modelo (detalle del ticket).
i) El modelo puede tener diferentes tallas, que están codificadas.
j) La clave de los modelos es alfanumérica.
k) Se tiene la descripción del modelo.

l) Cada detalle de ticket contiene el precio unitario de modelo en la fecha de la operación.
m) Cada detalle de ticket especifica las unidades compradas.
n) La cantidad multiplicada por las unidades da el subtotal.
o) Hay descuentos.
p) Aunque los descuentos se muestran consolidados, dados los requerimientos, deben ser por detalle de ticket.
q) Los impuestos existen, pero para el análisis no importan.
r) El gran total no importa, para los requerimientos.

Cuando ganas experiencia modelando datos, un simple documento va abriendo muchas posibilidades respecto a las tablas que se pueden generar.

El análisis de monto y las unidades de ventas se puede realizar, desde el momento que tenemos el monto de venta de cada detalle del ticket, y la cantidad vendida; el ticket confirma que se manejan descuentos, pero tendremos que consultar en otro lugar el tema de los descuentos.

El ticket no ofrece información de utilidades. Sabemos que las utilidades se calculan restando el costo de ventas al precio de venta. Como no tenemos el costo de ventas, tampoco podemos tener las utilidades, que deberemos conseguir en otro lado.

Respecto a las segmentaciones, la fecha en el ticket nos permite manejar la segmentación temporal; también podemos manejar con la información del ticket la segmentación por sucursal, lo que de cierta manera también nos permite la segmentación geográfica, pues al saber en qué sucursal fue, sabemos en qué ciudad, estado y país fue; la segmentación por tamaño también es posible. Saber el país nos permite saber qué tipo de moneda es, y al mismo tiempo, podemos saber cuál es el tipo de cambio a pesos en la fecha de operación.

8.5 Obtén la información faltante

¿Qué datos nos faltan para cumplir todos los requerimientos?

Hasta el momento, no sabemos cómo se da la clasificación por tipo de calzado; tampoco sabemos cómo se determina el tipo de sucursal.

No sabemos cómo se registra el costo de producción, cómo se calculan o determinan los descuentos, ni cómo se determina la utilidad.

Esa información no podemos sacarla del ticket, así que lo preguntamos al personal de la organización. Las respuestas que nos dan son las siguientes:

Cada modelo pertenece a un tipo de calzado (*Deportivo*, *Trabajo*, *Casual*, *Formal*, y así), por lo cual, podemos concluir que el tipo de calzado es un atributo de cada modelo, y conocer el modelo implica que también conocemos de alguna manera el tipo de calzado.

Respecto al tipo de sucursal, resulta que hay por el momento 3 (*Estándar*, *Premium*, *Outlet*); el tipo de sucursal determina qué tan elegantes y espaciosas son las sucursales, qué tantos empleados hay para atender a los clientes, y en qué zonas económicas están ubicadas; esto tiene algunas repercusiones: hay ciertos modelos que sólo se venden en ciertos tipos de sucursal; por ejemplo, hay un modelo de tenis muy exclusivo, llamado *Tanger*, que es bastante caro, y que sólo está disponible en las sucursales *Premium*, pues está demostrado que el tipo de cliente que va a las sucursales *Estándar* no tiene el poder adquisitivo para comprarlos; desde luego, hay algunos modelos clásicos, como *Verona*, que está a la venta en todos los tipos de sucursal, sin embargo, aplica una diferencia importante: comprar *Verona* en una sucursal *Premium* es más caro que comprarlo en una sucursal *Estándar*.

Como ya se complica un poco el tema del precio del producto, hago la pregunta clave: ¿Qué determina el precio? La respuesta que me dan es la siguiente: el precio está determinado por el modelo, en conjunto con el tipo de sucursal, y el país de venta.

Pregunto, para no asumirlo, si el tamaño del calzado determina de alguna manera el precio. Me dicen que el precio del calzado es igual para el modelo, sin importar el tamaño del calzado.

Donde hay precio, hay costo, así que pregunto también si el tamaño determina de alguna manera el costo. Me dicen que ahí sí hay variaciones: dependiendo del modelo y del tamaño, el costo es diferente, debido a la diferencia en el consumo de materiales.

Tomo nota entonces de lo siguiente:

a) Algunos modelos están disponibles únicamente en cierto tipo de sucursal, lo que me llevará a crear en algún momento una tabla de relación entre tipo de sucursal y modelo.
b) El precio está determinado por el modelo, el tipo de sucursal y país, lo que me va a llevará a crear en algún momento una tabla de relación entre modelo, tipo de sucursal y país, donde se indique el precio.
c) El costo depende del modelo y el tamaño, lo que me llevará a crear en algún momento una tabla de relación entre modelo y tamaño, donde se indique el costo de producción.
d) Cada atributo que participa en las tablas de relación, deberá estar como atributos clasificados, o como parte de una tabla con llave primaria.

Me queda claro que el catálogo de precios puede variar en cualquier instante, así que, en el detalle de ticket, tendré que almacenar de alguna manera el precio base del artículo comprado, a la fecha de la compra. Si es una compra en el extranjero, deberé consultar el tipo de cambio del día, para almacenar también el precio expresado en pesos, a la fecha.

El costo de producción será actualizado diariamente en el catálogo de productos (está expresado en pesos), a través de una interfaz con el sistema de información que controla el tema de producción; el costo de producción es el mismo sin importar en qué sucursal o país se realice la venta.

Como el costo de producción puede cambiar de un día a otro, en el detalle de ticket también se debe almacenar el costo de producción que se tenía al momento de la operación.

Siendo así, el detalle de ticket va tomando forma: requiero el número de ticket al que pertenece, el modelo que se está comprando, el tamaño del calzado, la cantidad que se está comprando, el precio base que tiene, el precio al que se vende realmente, el precio base convertido a pesos, y el precio al que se vende realmente convertido a pesos, y el costo de producción a la fecha de la operación. Con ello ya podemos calcular la utilidad que se tiene en cada operación, con datos a la fecha de la operación.

Como no tengo claro cuál pueda ser la llave primaria del detalle de ticket, pregunto si en un mismo ticket, se pueden comprar dos veces el mismo producto. Me dicen que sí, siempre y cuando el tamaño sea diferente.

Revisamos que todos los requerimientos de información pueden ser resueltos, y ahora sí, podemos proceder al modelo de datos.

8.6 Identificación de sujetos, eventos y estados

Para identificar sujetos, eventos y estados, hay que analizar desde el punto de vista conceptual todo lo que sabemos. Algunos de los conceptos terminarán siendo tablas, mientras que otros, terminarán siendo atributos, o incluso atributos clasificados.

Los atributos clasificados (catálogos) que identifico, son los siguientes:

a) País
b) Estado
c) Ciudad
d) Tipo de sucursal
e) Tamaño
f) Tipo de calzado

Además de los atributos clasificados, que son considerados sujetos, podemos identificar otros sujetos adicionales:

a) Sucursal
b) Vendedor
c) Modelo

Los eventos que identifico son los siguientes:

a) Ticket
b) Detalle de ticket

Los estados que identifico son los siguientes:

a) Moneda
b) Sucursal – Modelo
c) Modelo - Tipo de sucursal – País
d) Modelo + Tamaño

Si encuentras algunos atributos que sobren o falten, modifica el modelo.

8.7 Identificación de atributos

Los atributos clasificados (catálogos) que identifico, son los siguientes:

País (**id** + **nombre**)

Estado (**id** + **nombre** + referencia a **país**)

Ciudad (**id** + **nombre** + referencia a **estado**)

Tipo de sucursal (**id** + **nombre**)

`Tamaño` (`id` + `nombre`)

`Tipo de calzado` (`id` + `nombre`)

Además de los atributos clasificados, que son considerados sujetos, podemos identificar otros sujetos adicionales:

`Sucursal` (`id` + `nombre` + referencia a `ciudad`)

`Vendedor` (`id` + `nombre` + referencia a `sucursal`)

`Modelo` (`id` + `nombre` + descripción en `español` + descripción en `inglés`)

Los eventos que identifico son los siguientes:

`Ticket` (`id` + `fecha` + `vendedor` + `sucursal`)

`Detalle de ticket` (`id` de ticket + `modelo` + `tamaño` + `cantidad` + `precio base` + `precio venta` + `precio base convertido` + `precio venta convertido` + `costo` de producción)

Los estados que identifico son los siguientes:

`Moneda` (`id` + `nombre` + `tipo de cambio` del día)

`Sucursal – Modelo` (`sucursal` + `modelo`)

`Modelo - Tipo de sucursal – País` (`Modelo` + `Tipo de sucursal` + `País` + `Precio base`).

`Modelo - Tamaño` (`Modelo` + `Tamaño` + `Costo de producción`)

Si encuentras algunos atributos que sobren o falten, modifica el modelo.

8.8 Determinación de tipos de datos

Para cada tabla identificada, y una vez identificados los atributos, establece cuáles son los tipos de datos adecuados para los campos.

Cuida la consistencia de tipos, sobre todo en las llaves primarias y foráneas.

Algunos consejos importantes serían los siguientes:

a) Las cantidades monetarias, defínelas como `decimal (19,4)`.
b) En el caso del ejemplo, la fecha lleva hora, así que debes usar `datetime`.
c) La mayoría de las descripciones tendrán una desviación estándar de longitud, superior a 2 posiciones, por lo que se sugiere el uso de datos tipo `VAR`.

8.9 Normalización

Normaliza el modelo, hasta BCNF.

Dado que el modelo es completamente nuevo (no estamos partiendo de unos datos ya existentes), lo más seguro es que todo lo que tenemos ya esté en 1NF, e incluso en 2NF.

Verifica que todas las reglas se cumplen, hasta BCNF.

a) Revisa que los datos son atómicos.
b) Revisa que todas las dependencias funcionales son completas.
c) Revisa que no haya dependencias funcionales parciales.
d) Revisa que no haya dependencias funcionales transitivas.
e) Revisa que, en caso de que existan llaves candidatas, éstas también están normalizadas.

8.10 Elaboración de Diccionario de datos

Elabora al máximo detalle el diccionario de datos con información específica. Verifica que tenga la siguiente información:

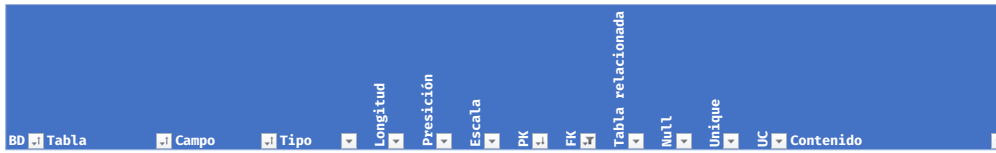


FIGURA 8.1: *Columnas de diccionario de datos detallado.*

Te recomiendo que lo hagas en una hoja electrónica de cálculo, donde puedas aplicar filtros y ordenamientos de forma sencilla, para que te sirva como herramienta de trabajo.

8.11 Elaboración del Diagrama de Entidad Relación

Elabora el diagrama de entidad relación.

Sugiero la notación de Hockey, en estilo *Crow – Foot*.

La herramienta en línea **Lucidchart** es muy sencilla de utilizar, y te permitirá realizar tu diagrama de forma gratuita. Otro software que puede ayudar mucho es **Microsoft Visio**, en su edición professional, que además te permite generar código SQL estándar, a partir del diagrama.

8.12 Codificación de la estructura de datos (DDL)

Codifica diferentes scripts para la creación de la base de datos:

a) Script para la creación de la base de datos.
b) Script para la creación de las tablas (incluyendo NULL y UNIQUE).
c) Script para la creación de restricciones de las llaves primarias.

d) Script para la creación de restricciones de las llaves foráneas.

Comprueba tu código en una implementación de base de datos real. Te sugiero la instalación de **XAMPP**, que trae **MySQL**; puedes instalar también **MySQL Workbench**, que te facilitará mucho la exploración de lo que haces.

8.13 Datos de comprobación (DML).

Codifica los scripts de carga de datos ficticios de tu base de datos; intenta también con código de eliminación y modificación de datos, para que revises si el modelo maneja adecuadamente las restricciones.

Es probable que tengas que definir un orden de carga de datos, de acuerdo con el modelo.

Intenta elaborar un script donde puedas generar un archivo maestro de consulta, que permita realizar operaciones de inteligencia de negocios. Se me ocurre que hagas un `select` que te permite ver lo siguiente:

a) País (descripción)
b) Estado (descripción)
c) Ciudad (descripción)
d) Sucursal (descripción)
e) Tipo de sucursal (descripción)
f) Modelo (ID)
g) Nombre del modelo (descripción)
h) Tipo de calzado (descripción)
i) Moneda (descripción)
j) Cantidad (Valor entero)
k) Precio base en pesos (Valor monetario)
l) Precio de venta en pesos (Valor monetario)
m) Costo de producción en pesos (Valor monetario)

n) Venta base (Valor monetario, calculado = `j * k`)
o) Venta real (Valor monetario, calculado = `j * l`)
p) Descuento (Valor monetario, calculado = `n - o`)
q) Utilidad (Valor monetario, calculado = `o - (j * m)`)

Si el modelo está bien hecho, podrás generar esta consulta maestra sin problemas, ya que las relaciones entre tablas permitirán llegar de un lugar a otro dentro del modelo.

Estos scripts seguramente se los tendrás que proporcionar a la gente de la organización, para que elabore bien las interfaces de carga de datos a tu base.

Este libro se termina aquí. Ojalá que haya sido de utilidad para ti.

Gracias por leerlo.

9 ÍNDICE

<

< · 111
<= · 111
<> · 111

=

= · 111

>

> · 111
>= · 111

1

1NF · 49, 50

2

2NF · 49, 53

3

3NF · 49, 56

A

actualización específica · 115
actualización global · 115
ALTER · 86
ALTER TABLE · 89, 93, 96
ambientación · 25
análisis · 16
AND · 112
ASCII · 31
atributo atómico · 51
atributos · 26, 27
atributos clasificados · 15, 27
atributos de coincidencia · 41
atributos no primos · 39
atributos primos · 39
atributos uniformes · 51
autodescriptiva · 25

B

base de datos · 25
BCNF · 57
BI · 24
BINARY · 32
Binary Large Objects · 32
BIT · 29
BLOB · 32
BOOL · 32
booleano · 23
business intelligence · 24

C

campos · 26, 27
cardinalidad · 75
Categoría · 24

Ch

CHAR · 32
char code · 31

C

clave · 39
colección · 25
columnas · 26, 27
conceptos · 17
condición · 111
condición compuesta · 112
condición lógica · 111
condición simple · 112
Conjuntivo · 112
consulta · 119
CREATE · 86
CREATE DATABASE · 87
CREATE TABLE · 89
Crow's Foot · 75

D

dashboard · 129
Data Definition Language · 86, 105
Data Manipulation Language · 86
DATE · 30
DATETIME · 31
dato · 13
datos · 26
datos de tipo cadena · 31
DDL · 86, 105
DECIMAL · 30
default · 99
DELETE · 86
DELETE FROM · 117
dependencia funcional completa · 54
Dependencia Funcional Completa · 50
dependencia funcional parcial · 54
Dependencia Funcional Parcial · 50
Dependencia Funcional Transitiva · 50
Descripción · 24
desnormalización · 58
Diagrama de Entidad Relación · 73
diccionario de datos · 69
Distribución concéntrica · 81
Disyuntivo · 112
DML · 86
dominio · 28, 41
dominio de integridad referencial · 42
Dominio de regla de negocio · 42
Dominio dependiente del modelo · 42
Dominio por tipo de dato · 42
DROP · 86
DROP DATABASE · 87
DROP TABLE · 90

E

eliminación específica · 117
eliminación global · 117
Entendimiento de requerimientos · 131
entidad débil · 41
entidades · 26
Entity Relationship Diagram · 73
ERD · 73
ERD de Hockey · 75
escala · 30
estado · 79
Estado · 24
estados · 15
Estados · 27
estructura · 25
Etiqueta · 23
eventos · 15
Eventos · 27
expresión lógica · 111

F

falta de integridad · 47

Fecha · 23
fenómeno · 11
filas · 26, 33
FLOAT · 30
foregin key constraint · 41
foreign key · 96
Forma Normal Boyce Codd · 57
formas normales · 48

I

idea · 16
identidad · 40
Identificación · 24
identity · 40, 99, 100
inconsistencia · 46
índice · 39
información · 14
INNER JOIN · 120
INSERT · 86
INSERT INTO · 105
INT · 29
inteligencia de negocios · 24

L

lenguaje declarativo · 85
lenguaje R · 9
lenguaje SQL · 85
LIKE · 111
lista de campos · 89

Ll

llave · 39
llave candidata · 57
llave compuesta · 39
llave foránea · 40
llave primaria · 39
llave primaria ficti · 39
llave primaria natural · 39
llave simple · 39
llaves candidatas · 39

L

Lógico · 23
longitud · 31

M

metadatos · 25
Métodos abreviados de creación de objetos · 101
Microsoft Dynamics · 20

N

NCHAR · 32
Negación · 112
networking · 127
NF · 48
normal forms · 48
normalización · 48
NOT · 112
NOT NULL · 89, 111
nulidad · 28, 39
nullable · 39
NUMERIC · 30
Numéricos · 23
NVARCHAR · 32

O

opcionalidad · 75
OR · 112
Oracle Netsuite · 20
ORDER BY · 120

P

partición vertical · 78
pensamientos · 17
Power BI · 9
precisión · 30
predicados · 111

primary key · 93
primary key constraint · 40
Primera forma normal · 49
Primera Forma Normal · 50
proposición · 16

Q

query · 119

R

redundancia · 45
redundancia controlada · 45
registros · 26, 33
registros de coincidencia · 41
relación · 41
relación de uno a muchos · 75
relación muchos a muchos · 79
relación uno a uno · 78
relaciones · 74
relaciones recursivas · 80
restricción de llave foránea · 41
restricción de llave primaria · 40
Restricciones de llave foránea · 96
Restricciones de llave primaria · 93
Restricciones de unicidad · 98
Restricciones de valor de auto incremento · 100
Restricciones de valor por omisión · 99

S

SAP Business One · 20
seed · 40, 100
Segmento · 24
Segunda forma normal · 49
Segunda Forma Normal · 53
SELECT · 86, 119
semilla · 40, 100
Serie · 23
síntesis · 17
SMALLINT · 29
snake_notation · 33
string · 31
strong entity · 41
strong table · 76
Structured Query Language · 85
sujetos · 15
Sujetos · 27

T

tabla de relación · 79
tabla fuerte · 41
tabla intermedia · 79
tablas · 26
Tableau · 9
tablero de datos · 129
Temporalidad · 24
Tercera forma normal · 49
Tercera Forma Normal · 56
TINYINT · 29
tipos de cadena de ancho fijo · 31
tipos de cadena de ancho variable · 31
tipos de caracter doble · 31
tipos de caracter simple · 31
tipos de datos · 23, 28
tipos de datos de fecha · 30
Tipos de datos misceláneos · 32
tipos de datos numéricos · 28
tipos de datos numéricos decimales · 30
tipos de datos numéricos enteros · 29
TRUNCATE TABLE · 90
tuplas · 26

U

unicidad · 28, 39
UNICODE · 31
unique · 98
UPDATE · 86, 115
UPPER() · 116
USE · 87

V

Valor · 24

VAR types · 31
VARBINARY · 32
VARCHAR · 32
vicios del modelo relacional · 45

W

weak entity · 41
weak table · 76
WHERE · 114, 120

APRENDA EDICIONES

SAN FÉLIX 5432, DESPACHO “D”

VISTA SOL, GUADALUPE, NUEVO LEÓN, MÉXICO

FECHA DE IMPRESIÓN: 10/JULIO/2020