PUERTO MADERO EDITORIAL

DISEÑO DE BASE DE DATOS:

FUNDAMENTOS TEÓRICOS Y PRÁCTICOS

AUTORES:
Ivonne Elizabeth Rodríguez Flores
Gisel Katerine Bastidas Guacho

EDITADO POR:
Juan Carlos Santillán Lima
Sandra Isabel González Polo

1era Ed.
2026

puertomaderoeditorial.com.ar

La Plata - Argentina

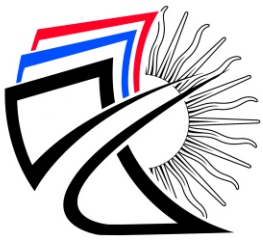
Diseño de base de datos: Fundamentos teóricos y prácticos

Ivonne Elizabeth Rodríguez Flores, Gisel Katerine Bastidas Guacho

ISBN: 978-631-6557-89-6

ISBN: 978-631-6557-89-6

ISBN: 978-631-6557-89-6



PUERTO MADERO EDITORIAL

Diseño de base de datos: Fundamentos teóricos y prácticos

Database Design: Theoretical and Practical Foundations

AUTORES:

Ivonne Elizabeth Rodríguez Flores

Gisel Katerine Bastidas Guacho

ISBN: 978-631-6557-89-6

Rodríguez Flores, Ivonne Elizabeth
Diseño de base de datos : fundamentos teóricos y prácticos / Ivonne Elizabeth Rodríguez Flores ; Gisel Katerine Bastidas Guacho ; Editado por Juan Carlos Santillán Lima ; Sandra Isabel González Polo. - 1a ed. - La Plata : Puerto Madero Editorial Académica, 2026.
Libro digital, PDF/A

Archivo Digital: descarga y online
ISBN 978-631-6557-89-6

1. Ingeniería Informática. I. Santillán Lima, Juan Carlos, ed. II. González Polo, Sandra Isabel, ed. III. Título.
CDD 600

ISBN: 978-631-6557-89-6

Primera Edición, Julio 2026

TÍTULO: Diseño de base de datos: Fundamentos teóricos y prácticos
ISBN: 978-631-6557-89-6
DOI: https://doi.org/10.55204/PMEA.143

Editado por:
Sello editorial: ©Puerto Madero Editorial Académica
Nº de Alta: 933832

Editorial: © Puerto Madero Editorial Académica
CUIL: 20630333971
Calle 45 N491 entre 4 y 5
Dirección de Publicaciones Científicas Puerto Madero Editorial Académica
La Plata, Buenos Aires, Argentina
Teléfono: +54 9 221 314 5902
+54 9 221 531 5142
Código Postal: AR1900

Este libro se sometió a arbitraje bajo el sistema de doble ciego (peer review)

Corrección y diseño:
Puerto Madero Editorial Académica
Diseñador Gráfico: José Luis Santillán Lima

Diseño, Montaje y Producción Editorial:
Puerto Madero Editorial Académica
Diseñador Gráfico: Santillán Lima, José Luis

Director del equipo editorial: Santillán Lima, Juan Carlos

Editor: Santillán Lima, Juan Carlos
González Polo, Sandra Isabel

Hecho en Argentina
Made in Argentina

ISBN: 978-631-6557-89-6

ISBN: 978-631-6557-89-6

AUTORES:

Ivonne Elizabeth Rodríguez Flores

Escuela Superior Politécnica de Chimborazo, Facultad de Informática y Electrónica, Grupo de Investigación de Tecnologías de la Información para la Gestión del Conocimiento (TIGECON), Panamericana Sur Km 1 1/2, EC060155, Riobamba, Chimborazo, Ecuador
irodriguez@espoch.edu.ec
http://orcid.org/0000-0002-7788-2609

Gisel Katerine Bastidas Guacho

Escuela Superior Politécnica de Chimborazo, Facultad de Informática y Electrónica, Carrera de Software, Panamericana Sur Km 1 1/2, EC060155, Riobamba, Chimborazo, Ecuador
gis.bastidas@espoch.edu.ec
https://orcid.org/0000-0002-6070-7193

ISBN: 978-631-6557-89-6

ISBN: 978-631-6557-89-6

CONTENIDO

ISBN: 978-631-6557-89-6

ISBN: 978-631-6557-89-6

ISBN: 978-631-6557-89-6

ISBN: 978-631-6557-89-6

ÍNDICE DE FIGURAS

ISBN: 978-631-6557-89-6

ÍNDICE DE TABLAS

ISBN: 978-631-6557-89-6

RESUMEN

El libro *Diseño de bases de datos: fundamentos teóricos y prácticos* ofrece una introducción integral al diseño de bases de datos, abordando de manera progresiva los fundamentos conceptuales, técnicos y metodológicos necesarios para su correcta implementación. A lo largo de cinco capítulos, se presentan los conceptos básicos de las bases de datos y los principales modelos de datos, el modelo relacional con su estructura y restricciones, y los principios fundamentales del diseño de bases de datos orientados a garantizar coherencia, integridad y eficiencia en entornos reales. Asimismo, se desarrolla el proceso de diseño, incluyendo el diseño conceptual mediante el modelo entidad–relación y el diseño lógico, con la transformación de esquemas conceptuales a esquemas relacionales y la aplicación de la normalización como herramienta clave para reducir redundancias y evitar anomalías en los datos. Cada capítulo incluye ejercicios prácticos que refuerzan los contenidos abordados y permiten llevar la teoría a la práctica mediante escenarios de diseño de bases de datos.

PALABRAS CLAVE: Bases de datos, principios de diseño, entidad-relación, modelo relacional, normalización.

ISBN: 978-631-6557-89-6

ABSTRACT

The book *Database Design: Theoretical and practical foundations* offers a comprehensive introduction to database design, progressively addressing the conceptual, technical, and methodological foundations required for its proper implementation. Across five chapters, it presents the basic concepts of databases and the main data models, the relational model with its structure and constraints, and the fundamental principles of database design aimed at ensuring consistency, integrity, and efficiency in real-world environments. The database design process is also developed, including conceptual design using the Entity–Relationship model and logical design, with the transformation of conceptual schemas into relational schemas and the application of normalization as a key tool to reduce redundancy and prevent data anomalies. Each chapter includes practical exercises that reinforce the topics covered and allow theory to be applied through database design scenarios.

Keywords: Databases, design principles, entity-relationship, relational model, normalization.

ISBN: 978-631-6557-89-6

INTRODUCCIÓN

Actualmente, los datos se han convertido en el recurso más valioso para la toma de decisiones, por lo que resulta necesario garantizar su correcta estructuración, almacenamiento, acceso y seguridad. Solo así las organizaciones pueden operar con eficiencia y asegurar la continuidad de sus procesos. Desde esta perspectiva, las bases de datos es una parte importante para el funcionamiento de las organizaciones. Por ello, este libro nace con el propósito de ser una guía clara y práctica para estudiantes, profesionales y entusiastas que buscan comprender y aplicar los principios del diseño de bases de datos.

A lo largo de sus capítulos, este libro aborda los fundamentos teóricos y metodológicos sobre el diseño de base de datos relacionales. Se introduce al lector al modelo Entidad–Relación, una de las herramientas más utilizadas para el modelado conceptual y se detalla el proceso de transformación de un esquema conceptual a un esquema relacional apto para su implementación en un Sistema de Gestión de Base de Datos. Asimismo, se proporciona la fundamentación teórica de la normalización, necesaria para garantizar esquemas relacionales consistentes, eficientes y estructuralmente libres de redundancias.

Dado que una base de datos relacional bien diseñada es, sin duda, garantía del buen funcionamiento de un sistema y facilita su futuro mantenimiento, este libro además ofrece ejercicios orientados a desarrollar competencias que hoy en día son indispensables en el mundo profesional. Finalmente, este texto espera convertirse en un recurso adecuado para reforzar habilidades, fomentar buenas prácticas de diseño y motivar la reflexión sobre la importancia de construir bases de datos relacionales robustas, escalables y coherentes. El diseño de base de datos es un campo en constante evolución, y aunque los principios de diseño relacional permanecen sólidos con el tiempo, su correcta aplicación requiere estudio, comprensión y práctica.

El libro se estructura en cinco capítulos que abordan de manera progresiva los fundamentos teóricos y prácticos del diseño de bases de datos. El primer capítulo introduce los conceptos básicos relacionados con las bases de datos y los modelos de datos,

ISBN: 978-631-6557-89-6

proporcionando el marco conceptual necesario para su comprensión. El segundo capítulo se centra en el modelo relacional, analizando su estructura y las restricciones que garantizan la integridad de los datos. En el tercer capítulo se desarrollan los principios básicos del diseño de bases de datos, considerando una serie de factores técnicos y metodológicos que permiten asegurar la correcta implementación del modelo en un entorno real; además, se presenta el método de diseño definido por el enfoque adoptado y por las fases que conforman el proceso de diseño. El cuarto capítulo está orientado al diseño conceptual mediante el uso del modelo entidad–relación, mientras que el quinto y último capítulo aborda el diseño lógico, incluyendo la transformación de esquemas conceptuales a esquemas relacionales y el proceso de normalización de bases de datos relacionales.

ISBN: 978-631-6557-89-6

CAPÍTULO 1

1 INTRODUCCIÓN A LAS BASES DE DATOS

1.1 Conceptos básicos

1.1.1 Relación entre dato e información

Es importante identificar la relación que existe entre los términos dato e información. Para lo cual, en la Tabla 1-1 se muestran las definiciones tomadas desde cuatro diferentes perspectivas y tipos de fuentes, que son: la IAIDQ, ISO/IEC 2382-1 y la Jerarquía DIKW.

- La IAIDQ (International Association for Information and Data Quality): es una sociedad sin fines de lucro conformada por personas dedicadas a mejorar la calidad de los datos e información. Las definiciones consideradas en el glosario autorizado de la IAIDQ corresponden a los autores: Martin Eppler y Larry English.
- ISO/IEC 2382-1:1993, *Tecnología de la información - Vocabulario - Parte 1: Términos fundamentales*. Los términos y definiciones de esta norma internacional son utilizados por las normas técnicas nacionales vigentes para tecnología de la información de diferentes países de la Región.
- Jerarquía DIKW (por sus siglas en inglés: Data, Information, Knowledge, Wisdom). Es un enfoque que surge en los dominios tanto de la gestión del conocimiento como de las ciencias de la información.

Una vez revisadas las definiciones desde diferentes perspectivas, es indiscutible la existencia de una relación jerárquica entre el dato e información; pero también, se reconoce que el dato se ubica como raíz o punto de partida que da lugar a la información y al conocimiento. El dato (del latín data, plural de datum) significa simplemente “hechos”, entidades independientes sin evaluar. Los datos pueden ser numéricos, alfabéticos, imágenes u otra representación de hechos. Así mismo, la relación entre dato e información está dada en función que, los datos son la materia prima (en crudo), que pueden producirse y presentarse en cualquier forma y que no tienen un significado por sí mismos. Y que, la

ISBN: 978-631-6557-89-6

información se define en términos del dato más el contexto por el cual pueda ser interpretado. La información está contenida en descripciones, y proporciona respuestas a las preguntas.

Tabla 1-1. Definiciones de dato, información y conocimiento

Fuente	Dato	Información
IAIDQ (IAIDQ, 2016)	(1) Los símbolos, números, u otra representación de hechos (2) Es la materia prima para producir la información cuando se pone en un contexto que le dé sentido (Larry English) Números o entradas crudos no relacionados, por ejemplo, en una base de datos; Formas crudas de representaciones transaccionales. (Martin Eppler)	(1) Los datos en su contexto, es decir, el significado dado a los datos o la interpretación de los datos basados en su contexto; (2) el producto terminado como resultado del procesamiento, presentación e interpretación de datos (Larry English) La información puede definirse como todas las entradas que las personas procesan para obtener comprensión. Es una diferencia (una distinción) que hace una diferencia, una respuesta a una pregunta. Un conjunto de datos relacionados que forman un mensaje. (Martin Eppler)
ISO/IEC 2382-1 (ISO/IEC 2382-1, 1993)	Representación re-interpretable de la información de una manera formal, adecuada para la comunicación, interpretación o procesamiento. *Nota 1*: Los datos pueden ser procesados por Humanos o por medios automáticos	Conocimiento relativo a objetos, como hechos, eventos, cosas, procesos o ideas, incluyendo los conceptos, que dentro de un determinado contexto tiene un significado particular
Jerarquía DIKW (Rowley, 2007)	Son símbolos que representan las propiedades de objetos y eventos. Russell Ackoff (1989).	Son datos que se procesan para ser útiles. La información está contenida en descripciones, y proporciona respuestas a las preguntas: "quién", "qué", "dónde" y "cuándo".

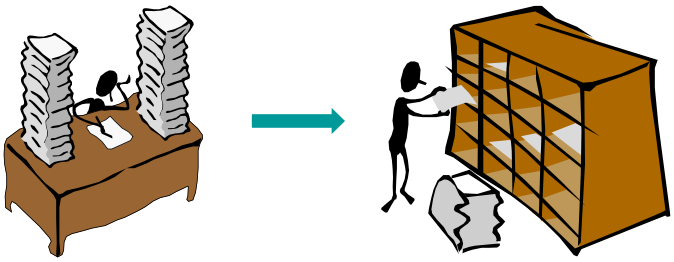
Fuente: (Rodriguez, 2020)

1.1.2 Base de datos

Tareas básicas, como búsquedas y ordenar la información se vuelven complejas y tomaría gran cantidad de tiempo. Razones valiosas para el surgimiento de las Bases de datos.

ISBN: 978-631-6557-89-6

Figura 1-1. El manejo de volúmenes de información en el proceso manual



Entre las diversas definiciones dadas a una base de datos, se han destacado las siguientes:

- Una colección compartida de datos relacionados lógicamente y su descripción, diseñada para satisfacer las necesidades de información de una organización (Connolly & Begg, 2015).
- Una base de datos es una colección de datos lógicamente coherente con algún tipo de significado inherente, y representa algún aspecto del mundo real (Elmasri & Navathe, 2011).
- Una colección de datos administrada por un sistema gestor de base de datos (SGBD, o sus siglas en inglés DBMS) (Silberschatz et al., 2014).

La tecnología de las bases de datos nace a principios de la década de los 60. Una base de datos se distingue por las características (Huawei Technologies Co., 2022) que se indican a continuación:

- Almacenamiento a largo plazo. El mecanismo de almacenamiento debe ser fiable y posibilitar la recuperación en caso de fallos, evitando pérdidas.
- Organización. Los datos deben organizarse, describirse y almacenarse de acuerdo con un modelo de datos primario.
- Acceso multiusuario: Los datos de la base de datos son compartidos y utilizados por múltiples usuarios simultáneamente.

ISBN: 978-631-6557-89-6

1.1.3 Data Base Management System - DBMS

Son diferentes las traducciones que se da a Data Base Management System (DBMS), por ejemplo: Sistema de Manejo de Bases de datos, Sistema Gestor de Bases de datos (SGBD) y Sistema de Organización de Bases de datos. De igual forma, son diversas las definiciones que se encuentran sobre un DBMS, de las cuales, se han destacado las siguientes:

- El DBMS es el software que permite a los usuarios definir, crear, mantener y controlar el acceso a la base de datos. Y, es el software que interactúa con los programas de aplicación de los usuarios y la base de datos (Connolly & Begg, 2015).
- El DBMS es el software de propósito general que facilita la definición, construcción, manipulación y compartición de las bases de datos entre varios usuarios y aplicaciones (Elmasri & Navathe, 2011).

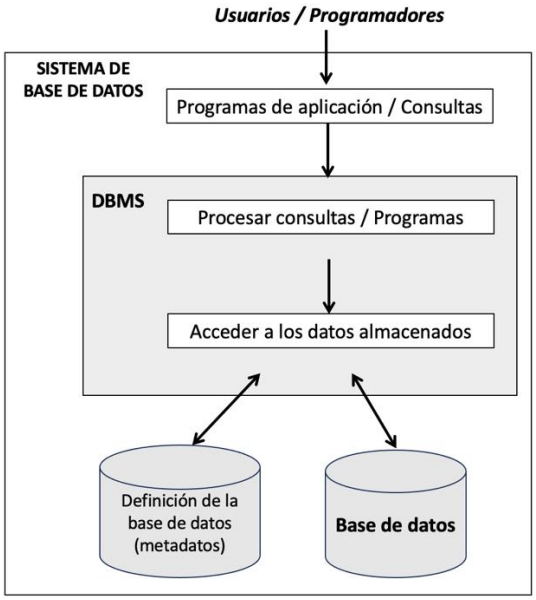
1.1.4 Sistema de base de datos

Según Connolly & Begg (2015), una *aplicación de base de datos* es un programa que interactúa con la base de datos en algún momento de su ejecución. Y, un *sistema de base de datos* es el conjunto de *programas de aplicación* que interactúan con la base de datos, junto con el DBMS y la propia base de datos. Por otro lado, Silberschatz (2014) define al *sistema de base de datos* como una colección de datos interrelacionados y un conjunto de programas que permiten a los usuarios tener acceso a esos datos y modificarlos.

En resumen, el *sistema de base de datos* es un sistema computarizado, que resulta de la combinación entre la base de datos y el DBMS (Date, 2004; Elmasri & Navathe, 2011). Estos conceptos, se ilustran en la Figura 1-2.

ISBN: 978-631-6557-89-6

Figura 1-2. Entorno de un sistema de bases de datos



Fuente: (Elmasri & Navathe, 2011)

1.1.5 Arquitectura de tres niveles ANSI-SPARC

Al compartirse una base de datos, cada usuario puede requerir una diferente vista de los datos almacenados, y para satisfacer esta necesidad, muchos DBMS se basan en la arquitectura de tres niveles propuesta en la década de los 70 por ANSI-SPARC (American National Standards Institute - Standards Planning and Requirements Committee), la cual provee las bases para comprender la funcionalidad de un DBMS.

La arquitectura de tres niveles ANSI-SPARC promueve la independencia de los datos, tanto lógica como física, y comprende tres niveles de abstracción, que son: externo, conceptual y el interno; los cuales se describen en la Figura 1-3.

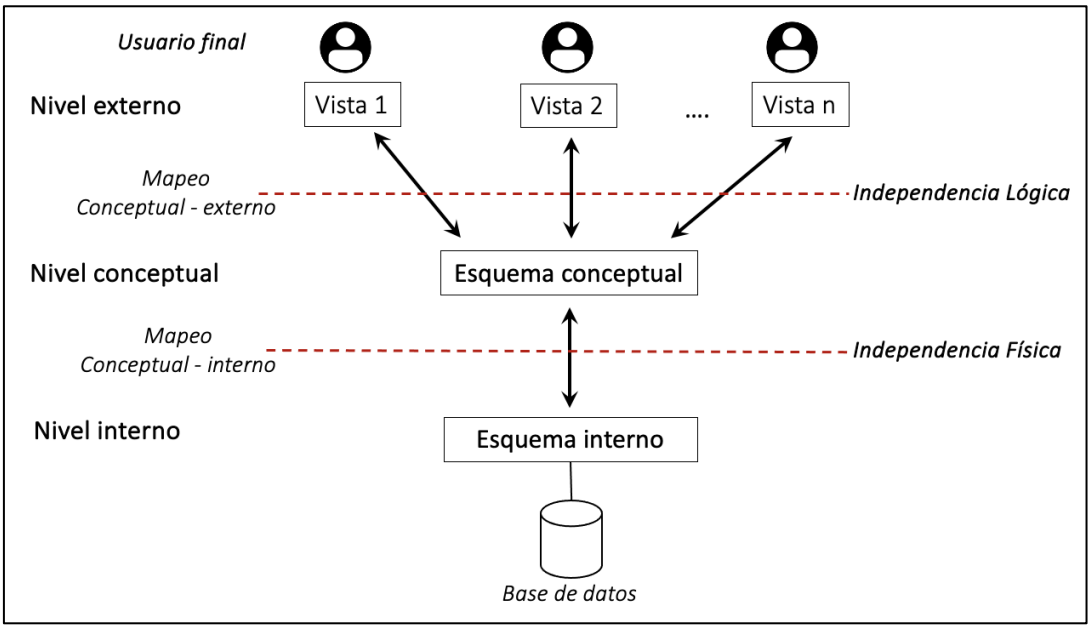
Nivel externo: Se encarga de la forma en que los usuarios perciben los datos. Puede haber múltiples vistas dependiendo del rol del usuario.

Nivel interno: Se encarga del cómo los datos son almacenados en el sistema

Nivel conceptual: También conocido como nivel lógico. El nivel conceptual proporciona tanto el mapeo como la independencia entre los niveles externo e interno.

ISBN: 978-631-6557-89-6

Figura 1-3. Arquitectura de tres niveles ANSI-SPARC



Fuente: (Connolly & Begg, 2015; Date, 2004; Elmasri & Navathe, 2011)

La arquitectura de tres niveles al promover la independencia de datos, significa que, los niveles superiores no se ven afectados por los cambios en los niveles inferiores. Se definen dos tipos de independencia de datos: lógica y física (Connolly & Begg, 2015).

- **Independencia lógica:** cambios en el nivel conceptual no afectan las vistas externas.
- **Independencia física:** cambios en el almacenamiento físico o nivel interno no afectan al nivel conceptual (modelo lógico).

El mapeo a dos etapas de la arquitectura ANSI-SPARC puede ser ineficiente, pero proporciona mayor independencia de datos. Sin embargo, para un mapeo más eficiente, también es posible el mapeo directo entre los esquemas externos al esquema interno, omitiendo el esquema conceptual; pero se reduce la independencia de los datos. De manera que, cada vez que cambia el esquema interno, el esquema externo y cualquier programa de aplicación dependiente también podría cambiar. Como parte de la funcionalidad de un DBMS, éste es el responsable del mapeo entre los diferentes niveles. El DBMS debe confirmar que cada esquema externo se puede derivar del esquema conceptual, y debe

ISBN: 978-631-6557-89-6

utilizar la información del esquema conceptual para establecer un mapeo entre cada esquema externo y el esquema interno (Connolly & Begg, 2015).

La arquitectura de tres niveles ANSI-SPARC se ha constituido en un marco teórico en el desarrollo de tecnologías de bases de datos, así como también, sirve de referencia cuando un sistema de bases de datos debe proporcionar a los usuarios una visión abstracta de los datos. De la misma forma, se ha aplicado en el diseño de las bases de datos, se inicia con una descripción abstracta y global de los requerimientos de información, los cuales se obtienen desde el dominio del mundo real.

1.1.6 Abstracción de los datos, Esquemas e Instancias

Los sistemas de bases de datos ofrecen a los usuarios una visión *abstracta de los datos*. Es decir, oculta ciertos detalles del modo en que se almacenan y mantienen los datos. La abstracción de los datos por niveles simplifica la interacción de los usuarios con el sistema (Silberschatz et al., 2014). Los niveles de abstracción están representados en la arquitectura ANSI-SPARC.

Es importante distinguir entre la *descripción* de la base de datos y la *misma base de datos*. La descripción general de la base de datos se denomina *esquema de la base de datos* que se especifica durante el proceso del diseño. Así mismo, se puede contar con tres tipos esquemas conforme con los niveles de abstracción de la arquitectura de tres niveles ilustrada en la Figura 1-3. En el nivel más alto se tiene uno o más esquemas externos (llamados subesquemas) que corresponden a diferentes vistas de los datos. En el nivel conceptual está el esquema conceptual (solo uno por base de datos) que describe la estructura de la base de datos, se concentra en describir las entidades, atributos y las relaciones entre éstos con las respectivas restricciones. Y, en el nivel más bajo de abstracción se encuentra el esquema interno, el cual contiene la definición completa de la base de datos, es decir, registros almacenados, métodos de representación, campos e índices utilizados (Connolly & Begg, 2015).

El esquema es especificado durante el proceso de diseño y no se espera cambios frecuentes. Aunque, los datos de la base de datos si cambian frecuentemente, por ejemplo, cuando se insertan, modifican o eliminan. Estos datos de la base de datos en un punto en

ISBN: 978-631-6557-89-6

el tiempo (como una fotografía de los datos en un instante del tiempo) es llamado *instancia de la base de datos* (Connolly & Begg, 2015).

Los esquemas se visualizan mediante convenciones, por ejemplo, en forma de diagramas, diagrama relacional, diagrama de clases UML, diagramas Entidad Relación, entre otros. Otra forma de expresar un esquema es con lenguaje DDL de SQL. Y, también se utiliza el lenguaje de descripción formal (modelo relacional), expresado con notación matemática o simbólica, donde define el esquema en términos de relaciones, atributos, dominios y claves, ver Ejemplo 1-1.

Ejemplo 1-1. Representación de un esquema de base de datos relacional

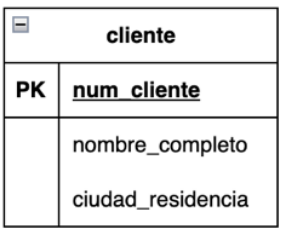
- **Notación simbólica - formal**

  cliente (num_cliente, nombre_completo, ciudad_residencia)

- **DDL - SQL**

```
CREATE TABLE Cliente (
   num_cliente INT PRIMARY KEY,
   nombre_completo VARCHAR(100),
   ciudad_residencia VARCHAR(100) );
```

- **Diagrama**



1.2 Modelo de datos

El Modelo de datos es una colección de conceptos para describir los datos, sus relaciones, su semántica y sus limitaciones; de tal forma que, con la abstracción facilita la interpretación del mundo real y su representación en forma de datos, en un sistema informático (Connolly & Begg, 2015; Silberschatz et al., 2014).

Es importante diferenciar entre el Modelo de datos y el Modelo de la base de datos, ya que ambos suelen tratarse de la misma forma; para este propósito, la Tabla 1-2 presenta

ISBN: 978-631-6557-89-6

los criterios de comparación que permiten diferenciar entre un modelo de datos y un modelo de base de datos.

Tabla 1-2. Comparación entre Modelo de datos y Modelo de base de datos

Criterio	Modelo de datos	Modelo de base de datos
Definición	Teoría para la organización y representación de los datos Se constituye en el Paradigma sobre el cual trabaja un DBMS	Representación de datos obtenida del proceso de diseño para implementarse en un DBMS Un modelo de base de datos se fundamenta en un modelo de datos.
	¿Cuál es la lógica o paradigma de organización y representación de los datos?	¿Qué lógica o paradigma (marco teórico) seguir para diseñar y crear una base de datos?
Otros nombres	- Modelo de datos General - Modelo de datos Teórico - Modelo de datos Paradigma	- Modelo de base de datos - Modelo específico de dominio - Modelo de diseño o Esquema de base de datos
Alcance	Teórico / Paradigma Estructura de datos	Desarrollo / Diseño (práctico) Implementación de la estructura en un DBMS
Ejemplos de modelos	- Modelo Relacional - Orientado a Objetos - Semiestructurado (XML, JSON)	Proceso de diseño: Conceptual: Entidad -Relación , UML Lógico: Tablas, atributos, claves Físico: índices, particiones
¿Cuál es su relación?	Proporciona el marco teórico (Paradigma) para diseñar una base de datos	Implementación de una base de datos (poner en práctica) conforme a un paradigma para un dominio específico de negocio (una realidad en particular) en un DBMS.
Ejemplo aplicado	Modelo Relacional: Tablas con filas y columnas, restricciones de integridad (claves), algebra relacional	Modelo lógico: *Base de datos para registro de ventas de productos* **tablas** (*producto, cliente, venta, detalle_venta*) **atributos** *cliente (id_cliente, nombre, direccion)* *producto (id_producto, nombre, descripcion)* *venta (num_venta, fecha_id_cliente)* *detalle_venta (num_venta, id_producto, cantidad)*

ISBN: 978-631-6557-89-6

		clave primaria (PK): *PK_cliente (id_cliente)* *PK_producto (id_producto)* *PK_venta (num_venta)* *PK_detalle_venta (num_venta, id_producto)* **clave foránea (FK)** *FK_venta (id_cliente)* *FK_detalle_venta1 (num_venta)* *FK_detalle_venta2 (id_producto)*

Elaborado por los Autores

De la misma manera, hay términos relacionados que requieren atención, tales como: modelado de datos y modelar datos (ver Tabla 1-3).

Tabla 1-3. Relación entre Modelamiento, Modelar y Modelo de datos

	Modelado de datos	**Modelar datos**	**Modelo de base de datos**
¿Qué es?	Proceso / Metodología	Acción	Resultado
Ejemplo	Proceso por niveles de abstracción: conceptual, lógico, físico	Diagrama Entidad Relación (DER)	Esquema relacional
Enfoque metodológico	Metodología	Técnica	Artefacto / Modelo

Elaborado por los Autores

- **Modelado de datos**: Es el *proceso* o *metodología* que guía la creación de un modelo de base de datos
- **Modelar datos**: Es la *acción* de crear un modelo de base de datos, implica organizar y representar los datos; es decir, actividades de diseño. Y, estás actividades de modelar se incluyen en un proceso de modelado de datos.

1.2.1 Categorías de los Modelos de datos

Conforme con los tipos de conceptos que se utilizan para describir la estructura de la base de datos (Elmasri & Navathe, 2011), los modelos de datos se los puede agrupar por modelos lógicos y físicos. Además de éstos, también se ha considerado el grupo de los modelos semánticos; no obstante, en este libro no se consideran los modelos físicos.

ISBN: 978-631-6557-89-6

Modelos físicos: Describen los detalles de cómo se almacenan los datos en el computador. Estos modelos de bajo nivel se corresponden al nivel interno según la arquitectura ANSI-SPARC, por tanto, no están pensados para el usuario final (Elmasri & Navathe, 2011).

Modelos lógicos: Conforme con la arquitectura ANSI-SPARC, los modelos lógicos **describen los datos** en los niveles: conceptual y externo (Connolly & Begg, 2015; Elmasri & Navathe, 2011). La descripción se centra en qué datos se almacenan y qué relaciones existen entre estos (Silberschatz et al., 2014). Entre los modelos lógicos más reconocidos tenemos al relacional.

Modelos semánticos: No especifican la forma de almacenamiento (parte estática) ni las operaciones para la manipulación de los datos (parte dinámica). Su propósito es capturar y representar el significado de los datos (semántica) desde un mundo real, utilizando conceptos abstractos. Hay modelos semánticos que se utilizan para el diseño de base de datos, por ejemplo, a partir del modelo Entidad Relación se deriva a un modelo lógico como el relacional (Connolly & Begg, 2015; Elmasri & Navathe, 2011). Así mismo, tenemos modelos de datos semánticos del tipo ontológico utilizados especialmente en el contexto de la web semántica y el intercambio de conocimiento estructurado, por ejemplo el modelo RDF (Lemahieu et al., 2018).

Los modelos de datos lógicos y semánticos, a su vez, se pueden categorizar desde dos perspectivas: por su estructura y por niveles de abstracción para diseño de la base de datos.

En la Tabla 1-4 se presentan un cuadro que categoriza reconocidos modelos de datos según su estructura (Aguilar Vera et al., 2023; Martinez-Mosquera et al., 2020; Tripathi, 2025). Por otro lado, la Tabla 1-5 muestra los modelos categorizados desde una perspectiva por niveles para diseño de base de datos (DB-Engines, n.d.; Shin et al., 2017; Silberschatz et al., 2014).

ISBN: 978-631-6557-89-6

Tabla 1-4. Modelos de datos categorizados por su estructura

Grupo	Categoría estructural	Modelo de datos	Ejemplos	Uso
Lógico	Basado en registros	Jerárquico	IBM IMS	DBMS
		Red	IDMS	DBMS
		Relacional	SQLServer PostgresSQL MySQL	DBMS
	Basado en Objetos	Orientado a Objetos	ObjectDB db4o	DBMS
	Semiestructurados	Documentos	MongoDB CouchDB	DBMS
		XML	BaseX eXist-db	DBMS
	No estructurados	Clave-Valor	Redis Riak	DBMS
		Grafos	Neo4j ArangoDB	DBMS
		Columnar	Cassandra Hbase	DBMS
Lógico / Semántico	Ontológico basado en grafos	RDF	Virtuoso GraphDB	DBMS
Semántico	Abstracción de conceptos	Entidad Relación	Diseño de base de datos	
	Basado en Objetos	Orientado a Objetos conceptual	Diseño de base de datos	

Elaborado por los Autores

De acuerdo con la historia de la tecnología de bases de datos, entre los primeros modelos de datos se encuentran los modelos lógicos basados en registros, denominados así porque el registro se constituye en la unidad estructural fundamental en la organización la base de datos. De estos modelos, el relacional es el que ha alcanzado gran relevancia, así se evidencia en el ranking de la plataforma DB-Engine (DB-Engines, n.d.), en la cual los primeros lugares son ocupados por DBMS relacionales (RDBMS).

ISBN: 978-631-6557-89-6

Tabla 1-5. Modelos de datos por niveles de abstracción para diseño de base de datos

Grupo	Categoría por Nivel de abstracción (diseño de base de datos)	Modelo de datos	Ejemplos
Lógico	**Lógico**	Jerárquico	IBM IMS
		Red	IDMS
		Relacional	SQLServer PostgresSQL MySQL
		Orientado a Objetos	ObjectDB db4o
		Documental - JSON	MongoDB CouchDB
		Documental - XML	BaseX eXist-db
		Clave-Valor	Redis Riak
		Grafos	Neo4j ArangoDB
		Columnar	Cassandra Hbase
Lógico / Semántico	**Conceptual extendido**	RDF	Virtuoso GraphDB
Semántico	**Conceptual / Lógico**	Entidad Relación	DER
	Conceptual	Orientado a Objetos conceptual	UML

Elaborado por los Autores

1.2.2 Clasificación de los DBMS

Un DBMS desde la perspectiva estructural, se fundamenta en un modelo lógico de datos, aunque han tenido una constante evolución, al punto que, actualmente muchos de los DBMSs tienen la característica de multimodelo. Sin embargo, siguen identificándose con un modelo de datos primario.

Hay varios criterios para clasificar los DBMSs, entre ellos tenemos los propuestos por (Elmasri & Navathe, 2011), que son: según el modelo de datos, por el número de usuarios y por el número de sitios sobre los cuales está distribuida la base de datos.

a) El *modelo de datos* en el que se basa el DBMS. Las Tablas anteriores (1-4 y 1-5) presentan DBMSs categorizados por modelos de datos. Así mismo, la plataforma

ISBN: 978-631-6557-89-6

de ranking DB-Engines[1] publica de forma mensual la popularidad de los DBMSs; los cuales, también son clasificados según el modelo de datos. Otra forma frecuente de clasificar a los DBMSs toma como referencia al modelo relacional, derivándose tres clases principales: SQL, NoSQL y NewSQL. La Tabla 1-6 resume las características generales de estas clases (Acharya et al., 2019; Garba & Abubakar, 2020; Jatana et al., 2012; Veerappampalayam et al., 2025), además, organiza los ejemplos de DBMSs mencionados en las Tablas 1-5, de acuerdo con su pertenencia al tipo SQL, NoSQL y NewSQL.

Tabla 1-6. Clasificación del DBMS en SQL, NoSQL y NewSQL

Clase DBMS	Características	Modelo de datos	DBMS
SQL	***Esquema***: Fijo ***Lenguaje:*** SQL ***Soporte ACID:*** SI ***Escalabilidad:*** Vertical ***Uso frecuente:*** Soluciones específicas de dominio. Se especializa en soluciones de negocio críticas y sectoriales, principalmente, donde prevalece el procesamiento transaccional y los datos son estructurados. ***Ventajas:*** - Garantiza la integridad y consistencia de los datos - Lenguaje SQL estándar - Soporte de transacciones - Madurez y estabilidad ***Desventajas:*** - Menos flexibles para datos no estructurados - Dificultad para la escalabilidad horizontal - Poco flexible para cambios en el esquema	Relacional	SQLServer PostgresSQL MySQL
NoSQL	***Esquema:*** Flexible (cada modelo tiene un esquema diferente) ***Lenguaje:*** Varía por cada modelo - Documental: MongoDB usa el MQL (Mongo Query Language), basado en JSON - Clave-Valor: Redis usa comandos propios	Orientado a Objetos	ObjectDB db4o
		Documental - JSON	MongoDB CouchDB

[1] https://db-engines.com/en/ranking

ISBN: 978-631-6557-89-6

<table>
<tr><td rowspan="6"></td><td rowspan="6">- Columnar: Cassandra usa CQL (Cassandra Query Language, similar a SQL
Soporte ACID: No siempre
Escalabilidad: Horizontal
Uso frecuente: Big Data, IoT
Ejemplos: e-Commerce, redes sociales, procesamiento en tiempo real y análisis, gestión de conocimiento con Ontologías, web semántica, linked data, logs masivos, IoT y sensores
Ventajas:
- Escalable y flexible
- Ideal para big data y procesamiento en tiempo real
- Buen rendimiento con grandes volúmenes de datos
Desventajas:
- No dispone de un lenguaje único (falta de un estándar)
- Consistencia de datos disminuida en algunos casos
- Falta de madurez
- Complejidad de aprendizaje según modelo
- Dificultad para las relaciones complejas entre datos</td><td>Documental - XML</td><td>BaseX
eXist-db</td></tr>
<tr><td>Clave-Valor</td><td>Redis
Riak</td></tr>
<tr><td>Grafos</td><td>Neo4j
ArangoDB</td></tr>
<tr><td>Columnar</td><td>Cassandra
Hbase</td></tr>
<tr><td>RDF</td><td>Virtuoso
GraphDB</td></tr>
<tr><td></td><td></td></tr>
<tr><td>NewSQL</td><td>Esquema: Fijo / Flexible
Lenguaje: SQL
Desde punto de vista estructural se considera Relacional, la diferencia radica en la arquitectura y rendimiento
Soporte ACID: SI
Escalabilidad: Horizontal
Uso frecuente:
E-commerce global, IoT con transacciones críticas
FinTech (sistemas financieros modernos), SaaS
Ventajas:
- Combinan lo mejor de SQL (consistencia, integridad) y NoSQL (escalabilidad masiva)
- Ideales para transacciones críticas, generalmente en la nube
- Diseñados para baja latencia y tiempo real.
Desventajas:
- Falta madurez incluso comparado con los NoSQL
- Comunidad y ecosistema más pequeños
- Curva de aprendizaje</td><td>Relacional (desde el punto de vista estructural)</td><td>Google Spanner,
CockroachDB,
VoltDB
NuoDB,
SingleStore</td></tr>
</table>

Elaborado por los Autores

ISBN: 978-631-6557-89-6

b) El *número de usuarios* soportados por el DBMS. Las características básicas simultaneidad y concurrencia para el manejo de usuarios, miden la capacidad del DBMS. La *simultaneidad* se refiere al número de usuarios que pueden conectarse y utilizar la base de datos al mismo tiempo, de esta forma, un DBMS puede ser *monousuario*, como por ejemplo SQLite o Access; éstos solo permite el acceso de un usuario a la vez. Por otro lado, el DBMS *multiusuario*, por ejemplo PostgreSQL, SQLServer u Oracle, permiten miles de conexiones simultáneas. En cuanto a la característica *concurrencia,* es la forma en que el DBMS gestiona las operaciones que los usuarios realizan en paralelo, garantizando la integridad y consistencia de los datos.

c) El *número de sitios* sobre los cuales está distribuida la base de datos, de esta manera, si el DBMS con la base de datos residen en un solo computador este es *centralizado*. Si el DBMS se encuentra en varios sitios conectados por una red de computadores al igual que la base de datos, entonces se dice que es *distribuido*. Por otro lado, un DBMS es *homogéneo* cuando utiliza el mismo software DBMS en varios sitios. Así también, se tiene al DBMS *federado* (o *sistema multibase*), en el que los DBMSs que participan se acoplan con cierto grado de autonomía.

ISBN: 978-631-6557-89-6

CAPÍTULO 2

2 MODELO RELACIONAL

2.1 Historia

En 1970, Edgar Frank Codd, investigador de IBM, publicó el artículo seminal "*A Relational Model of Data for Large Shared Data Banks*", en el cual se introduce la teoría del modelo relacional. Este modelo se basa en dos ramas de las matemáticas, la teoría de conjuntos y la lógica de predicados de primer orden. La teoría matemática fundamenta al modelo relacional, haciendo que sea fiable y seguro. En este modelo se utiliza el concepto de una *relación matemática* como una *tabla* de valores (Connolly & Begg, 2015; Elmasri & Navathe, 2016; Huawei Technologies Co., 2022).

El modelo relacional, principalmente por su simplicidad, se ha establecido como el principal modelo de datos para las aplicaciones de procesamiento de datos. Los DBMSs relacionales (RDBMS) totalmente desarrollados estuvieron disponibles comercialmente a principios de los años ochenta.

La teoría del modelo relacional se apoya en términos formales provenientes del fundamento matemático. Sin embargo, en el ámbito de la tecnología de las bases de datos relacionales, algunos de los conceptos se han conservado, y otros se han reemplazado o transformado (evolución).

En la terminología formal del modelo relacional, una *fila* recibe el nombre de una *tupla*, un encabezado de *columna* es un *atributo* y la *tabla* es una *relación*. El *tipo de datos* que describe los valores que pueden aparecer en cada columna está representado por un *dominio* de valores posibles.

En la Tabla 2-1 se presenta la correspondencia entre algunos de los términos formales del modelo relacional y sus equivalentes en las implementaciones prácticas de los RDBMS. Asimismo, se incluyen los términos tradicionales que utilizan los sistemas de archivos, con el fin de mostrar la relación y coexistencia entre las distintas terminologías empleadas en el ámbito de las bases de datos relacionales.

Tabla 2-1.- Relación entre los términos empleados en bases de datos relacionales

ISBN: 978-631-6557-89-6

Término tradicional o informal	Término forma del Modelo Relacional	Equivalencia - Términos comunes en RDBMS
Archivo	Relación	Tabla
Registro	Tupla	Fila
Campo	Columna / Atributo	Columna
Tipo de dato	Dominio	Tipo de datos / Dominio
Asociación entre archivos (campo común)	Referencia	Relación entre tablas (vínculo visual entre tablas)
Archivo maestro	Dependencia referencial (Relación padre)	Tabla padre
Archivo detalle	Dependencia referencial (Relación hija)	Tabla hija
Relación maestro-detalle	Dependencia referencial	Activada en DBMS mediante "Integridad referencial" (tabla padre y tabla hija)
Campo clave	Clave primaria	Clave primaria (Primary Key - PK)
Campo relacionado /enlace	Clave foránea	Clave foránea (Foreign Key - FK)
-	Dependencia funcional	Restricción (Constraint)

Elaborado por los Autores

2.2 Estructura del Modelo Relacional

Un modelo de datos debe tener la capacidad de representación, tanto para las propiedades estáticas como las dinámicas de los datos. Es así como, en el modelo relacional se pueden representar las propiedades estáticas mediante el lenguaje de definición de datos (Data Definition Language – DDL) y las dinámicas con el lenguaje de manipulación de datos (Data Manipulation Language – DML).

Las propiedades estáticas corresponden a cómo están definidos y organizados los datos, es decir, la *estructura* del modelo. Se describe mediante el lenguaje de definición de datos (DDL, en inglés).

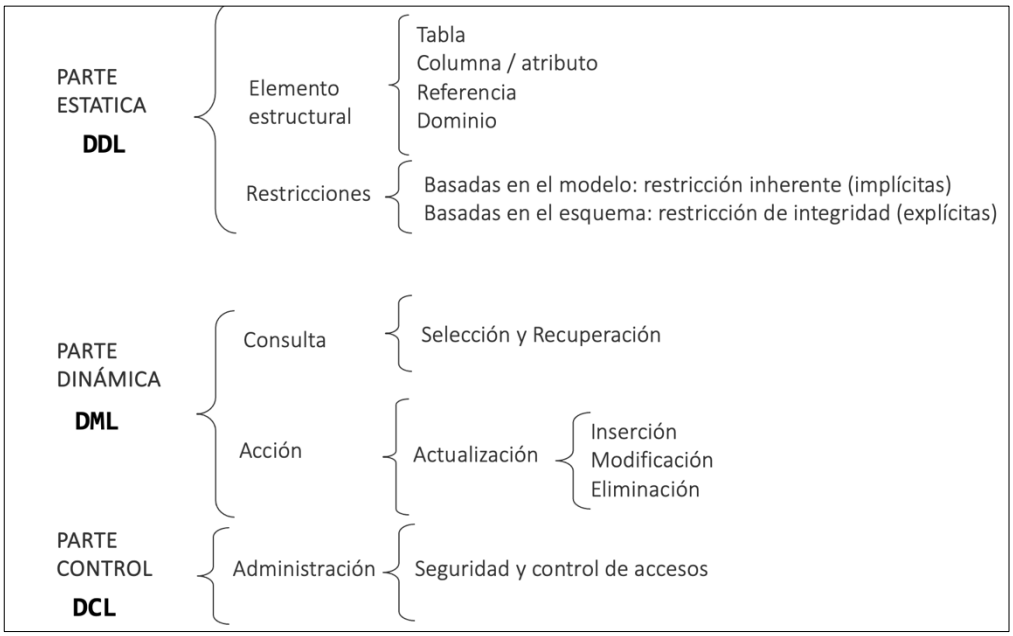
Las propiedades dinámicas son aquellas que varían en el tiempo. En el modelo de datos son las OPERACIONES. Corresponde a cómo los datos se manipulan y cambian

ISBN: 978-631-6557-89-6

dentro de la estructura definida. En otras palabras, estas alteran el estado de la base de datos. Se expresa mediante el lenguaje de manipulación de datos (DML, en inglés).

El DBMS congruente con el modelo de datos proporciona el DDL para la definición de los datos y el DML que permite a los usuarios insertar, modificar, eliminar y consultar datos (Lucas Gomez, 1993). Adicional a estas operaciones, el DBMS debe permitir acciones de administración, como el control de la seguridad y permisos de acceso a la base de datos. Para este propósito, el DBMS también provee el lenguaje de control de datos (Data Control Language - DCL). Con base en Lucas Gomez (1993), en la Figura 2-1 se sintetiza las propiedades de la parte estática, dinámica y de control que un DBMS debe proporcionar.

Figura 2-1. Propiedades del modelo de datos relacional proporcionadas por el RDBMS



Elaborado por los Autores

Cuando se habla de la estructura y restricciones en el modelo relacional, se refiere a elementos u objetos definidos con DDL en el DBMS; tales como: tabla, dominio, columna, referencias y restricciones.

- **Tabla**

En una base de datos relacional, la *tabla* (*relación*, término formal) es el elemento bidimensional central donde se almacenan los datos organizados en *filas* (*tupla*, término

ISBN: 978-631-6557-89-6

formal) y *columnas* (o atributos). Por otro lado, en el contexto de diseño de base de datos, una tabla se constituye en la representación estructurada de una entidad o concepto del mundo real.

Desde el enfoque formal del modelo relacional, la *Relación* (R) se define como un conjunto finito de *Tuplas* (t) que comparten los mismos *Atributos* (A) definidos por un esquema con un *Dominio* (D) específico (Connolly & Begg, 2015; Elmasri & Navathe, 2016; Piñeiro Gómez, 2013).

Una tupla (t) es el elemento del conjunto R.

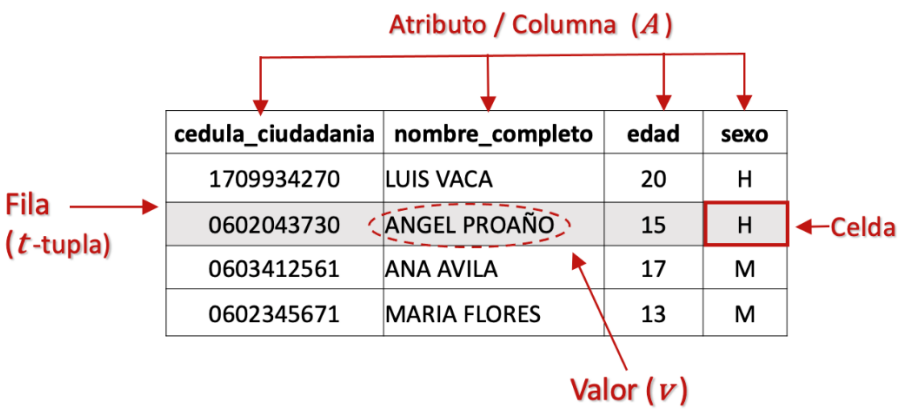
$$R=\{t_1, t_2, t_3,\ldots, t_n\}$$

Donde, en cada *celda* de la tupla t_i, el valor del dato v se corresponde a un *atributo* A de la *relación* R. Desde una perspectiva semántica, el valor al asociarse con el atributo (A_m, v_m), solo entonces el ***dato*** **adquiere significado**. A su vez, el *valor* v contenido en la *celda* es *atómico* y pertenece al *dominio* D del *atributo* A correspondiente; garantizando la propiedad de atomicidad del modelo relacional.

$$t_i = \{ (A_1, v_1), (A_2, v_2),..., (A_m, v_m) \}$$

Ejemplo 2-1. Representación gráfica de una tabla

estudiante (cedula_ciudadania, nombre_completo, edad)



ISBN: 978-631-6557-89-6

- **Dominio**

Un **dominio** D es el **conjunto de** ***valores*** *atómicos* v **posibles** que puede tomar un atributo A dentro de una relación R.

> Si el atributo A_i pertenece a una relación R, existe un dominio D_i tal que, todo valor v_i asociado a A_i cumple $v_i \in D_i$

En el modelo relacional, el dominio es predefinido ***por el sistema*** (DBMS); por ejemplo, el tipo de dato (numérico, texto, fecha, entre otros), con la longitud, formato y precisión. El dominio también puede ser definido ***por el usuario***, es decir, crear su propio dominio y asociarlo a uno o varios atributos de una relación. Con el dominio se asegura la coherencia e integridad de los datos.

> *Dato atómico:* Por atómico se entiende que cada valor de un dominio es indivisible. La atomicidad de los valores fue establecida por Codd como una de las propiedades **fundamentales del modelo relacional,** cada celda debe contener **un valor que no puede subdividirse**; es decir, no una lista, conjunto o estructura compuesta.

Ejemplo 2-2. Representación gráfica del dato atómico

cliente (num_cliente, nombre_completo, ciudad_residencia, teléfono_contacto)

num_cliente	nombre_completo	ciudad_residencia	telefono_contacto
100	JORGE ARIAS	QUITO	0994410187, 2941567
101	GISEL FLORES	AMBATO, MANTA	0854123456
102	MARTHA SALINAS	RIOBAMBA	0890123456, 2245643
103	JUAN CAMILO VARGAS	GUAYAQUIL	0987654321

En el Ejemplo 2-2 se identifica problemas de atomicidad. El cliente con num_cliente 101 registra dos ciudades de residencia (lista de valores), AMBATO y MANTA. Según la propiedad de atomicidad del modelo relacional, solo debe registrarse un valor por celda . De la misma forma ocurre en el atributo teléfono_contacto para los clientes con num_cliente 100 y 102.

ISBN: 978-631-6557-89-6

Si se requiere registrar varios valores de dato, como el caso de teléfono o de ciudades de residencia, debe insertarse una nueva fila; mientras, no se incumpla con la regla de clave primaria.

2.3 Restricciones del modelo relacional

Las restricciones (constraints, en inglés) para una base de datos relacional, generalmente se pueden dividir en tres categorías, que son: restricciones basadas en el modelo, restricciones basadas en el esquema y las restricciones basadas en la aplicación (Elmasri & Navathe, 2011).

- Restricciones basadas en el modelo, también llamadas implícitas. Estas restricciones son inherentes al modelo relacional mismo; por esta razón, en este texto se los denomina restricciones inherentes.
- Restricciones basadas en el esquema, también llamada explícitas. Estas restricciones son aquellas que pueden ser expresadas con DDL directamente en el esquema. En este texto se las llama restricciones de integridad.
- Restricciones basadas en la aplicación, también llamadas semánticas o de reglas de negocio. Estas restricciones no pueden ser expresadas por el esquema del modelo de datos, por lo que deben ser implementadas por los programas de la aplicación.

En la siguiente sección del texto se centra en la restricción basada en el modelo o inherente, y la basada en el esquema, también llamada restricción de integridad.

2.3.1 Restricciones inherentes

Las restricciones inherentes son aquellas que el modelo relacional impone de manera natural, como consecuencia de su fundamento matemático; las cuales se listan a continuación (Connolly & Begg, 2015; Marqués, 2011):

- **Unicidad de nombres**
  - La tabla (relación) tiene un nombre distinto de todos los demás nombres de la tabla (relación) en el esquema relacional

    Ejemplo: No se puede haber dos o más tablas (relaciones) con el mismo nombre "estudiante"

ISBN: 978-631-6557-89-6

- Cada atributo tiene un nombre distinto

  Ejemplo: No se pueden tener dos atributos (columnas) llamados "nombre".

- **Atomicidad**
  - Cada celda de la tabla (relación) contiene exactamente un valor atómico (único)

    Ejemplo: No almacenar una lista de teléfonos en una sola celda

- **Dominio definido**
  - Los valores de un atributo pertenecen al mismo dominio

    Ejemplo: el atributo edad tiene el dominio número entero

- **Duplicidad**
  - Cada fila (tupla) es distinta, no hay filas (tuplas) duplicadas

    Ejemplo: No pueden existir dos filas exactamente iguales en una tabla.

- **Independencia del orden**
  - El orden de los atributos no tiene relevancia
  - El orden de las filas (tuplas) no tiene relevancia teórica (en la práctica, el orden puede afectar la eficiencia del acceso a las filas).

- **Identificabilidad de filas (tuplas) - unicidad**
  - Cada fila (tupla) debe distinguirse de las demás, generalmente mediante una clave.

    Ejemplo: Poder identificar a un solo estudiante (LUIS) en una tabla con *n* estudiantes ingresados

NOTA:

Una base de datos relacional es un conjunto de relaciones normalizadas.

2.3.2 Restricciones de integridad

El modelo relacional admite varios tipos de restricciones (constraints, en inglés) de integridad sobre los valores de los atributos, tales como: restricción de dominio, de clave, de *Null*, integridad de la entidad e integridad referencial (Elmasri & Navathe, 2011; Lemahieu et al., 2018). Estas restricciones aseguran que los datos sean correctos y consistentes, en

ISBN: 978-631-6557-89-6

otras palabras, apoyan la calidad de los datos. El DBMS es el encargado de verificar que las restricciones se cumplan y que las violaciones sean reportadas (Lemahieu et al., 2018).

- **Restricción de dominio**

En la sección anterior del texto se trató sobre la forma en que se define el dominio en el modelo relacional. A continuación, se lista las características de una restricción de dominio.

- El domino restringe los valores permitidos para el atributo. Para garantizar la validez de los datos, antes que estos sean insertados o actualizados en la base de datos, el dominio se apoya de condiciones lógicas o reglas que valide el valor del dato para cada atributo. Un ejemplo práctico, es el uso de la restricción de integridad CHECK.
- Cada atributo se asocia a un dominio, el cual puede ser predefinido ***por el sistema*** (DBMS) o definido ***por el usuario***.
- El dominio actúa a nivel de atributo (columna)
- Los dominios pueden ser distintos para cada atributo, o se pueden definir dos o más atributos con el mismo dominio.

Ejemplo 2-3. Restricción de dominio

estudiante (cedula_ciudadania, nombre_completo, edad, sexo)

cedula_ciudadania	nombre_completo	edad	sexo
1709934270	LUIS VACA	20	H
0602043730	ANGEL PROAÑO	15	H
0603412561	ANA AVILA	17	M
0602345671	MARIA FLORES	13	M

Atributo	**Dominio**		
	por sistema	**por usuario**	
cedula_ciudadania	Tipo de dato: character Tamaño: 10 Valor: 10 dígitos		CHECK: cedula_ciudadania LIKE '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'

ISBN: 978-631-6557-89-6

nombre_completo	Tipo de dato: character Tamaño: 100		
edad	Tipo de dato: integer Rango: 15 (min) – 50 (max)		CHECK: edad >15 and edad< 50
sexo		Nombre del dominio: d_sexo Tipo de dato: character Tamaño: 1 Valor: 'H' or 'M'	CHECK: sexo='M' or sexo='H'

- **Restricción de clave**

Cada tabla (relación) tiene una clave que permite identificar de forma única sus filas (tulas).

> *Todos los elementos de un conjunto son distintos; por lo tanto, todas las tuplas de una relación también deben serlo. Esto significa que dos tuplas no pueden tener la misma combinación de valores para todos sus atributos* (Elmasri & Navathe, 2011).

Como se mencionó anteriormente, no debe existir filas (tuplas) duplicadas dentro de una tabla (relación); para este propósito, se define la clave primaria (Primary Key, en inglés - PK).

- **Null**

Nulo (Null) representa un valor para un atributo que actualmente se desconoce o no es aplicable par al fila (Connolly & Begg, 2015).

El Null puede interpretarse lógicamente, como: "*sin dato*"

Un nulo no representa el valor cero ni la cadena vacía, éstos son valores que tienen significado. El nulo implica *ausencia del valor del dato*, un atributo puede ser nulo por dos razones:

- Porque al insertar la fila *se desconocía el valor* del atributo (en ese instante), o
- Porque para dicha fila el atributo *no tiene sentido* (es inaplicable).

ISBN: 978-631-6557-89-6

- **Integridad de entidad**

Integridad de entidad en el modelo relacional es una regla que se aplica a la clave primaria (PK). Aunque, el término "entidad" de la restricción se vincula más al nivel conceptual del diseño de base de datos; y, una tabla del modelo relacional se corresponde a una entidad del esquema conceptual, la cual resulta de la abstracción y representación desde el mundo real.

En la restricción de entidad, el atributo o los atributos que componen la clave primaria siempre deben cumplir la restricción Not null (Connolly & Begg, 2015; Elmasri & Navathe, 2011). En resumen, la clave primaria no puede ser nula.

- **Integridad Referencial**

Integridad referencial es una regla que se aplica a la clave foránea (FK). Una clave foránea (FK) tiene el mismo dominio que los atributos de la clave primaria (PK) a los que hace referencia o ser Null (Elmasri & Navathe, 2011).

2.4 Enfoque práctico con un RDBMS

En el contexto de bases de datos, los RDBMSs y otros software relacionados, para asegurar la coherencia del modelo lógico relacional con su aplicación práctica, deben permitir la correcta implementación de las restricciones, tales como los distintos tipos de claves y la dependencia referencial.

A partir de esta sección, se adoptará un enfoque práctico, utilizando términos comunes en los DBMS y otros software, como *tabla*, *fila* y *atributo* (o *columna*), en lugar de la terminología formal del modelo relacional, para facilitar la comprensión de su aplicación en entornos reales.

2.4.1 Claves para una base de datos relacional

Cuando se trabaja con una base de datos relacional es necesario identificar las claves a ser utilizadas, algunas de éstas ya se han revisado, como son: la clave primaria (PK) y la clave foránea (FK); las cuales son definidas mediante DDL. Sin embargo, en un enfoque práctico es necesario que se consideren otros tipos de claves; entre las

ISBN: 978-631-6557-89-6

frecuentemente utilizadas, tenemos las claves: candidata, búsqueda (o secundaria) y la subrogada (o sustituta).

- **Clave candidata (CK)**

La clave candidata (Candidate Key - CK) es un atributo o grupos de atributos cuyo contenido (valor) puede representar de manera única a cada fila de una tabla.

Características

- *Unicidad*: La clave candidata cumple la propiedad de Unicidad, es decir el valor del atributo no debe duplicarse. Por tanto, se debe asegurar que el valor sea único.
- En una tabla puede haber varias claves candidatas, pero sólo una de ellas se designa como clave primaria (PK).
- La clave candidata que no se elige para ser clave primaria, también es llamada *clave alterna*.
- Si la tabla solo tiene una clave candidata, entonces, ésta será la clave primaria (PK) y no se tendría que elegir.

Ejemplo 2-4. Clave Candidata

estudiante (codigo, cedula_ciudadania, nombre_completo, edad, sexo)

PK CK

codigo	cedula_ciudadania	nombre_completo	edad	sexo
1001	1709934270	LUIS VACA	20	H
2005	0602043730	ANGEL PROAÑO	15	H
2008	0603412561	ANA AVILA	17	M
1099	0602345671	MARIA FLORES	13	M

En la tabla ESTUDIANTE del Ejemplo 2-4, se identifican dos claves candidatas (CK), los atributos: codigo y cedula_ciudadania. Los dos atributos proceden del contexto del mundo real y permiten identificar una fila de la tabla; por tanto, se constituyen en claves candidatas. De estas candidatas, se ha elegido codigo como PK; por tanto, cedula_ciudadania se mantiene como clave candidata, constituyéndose en una alternativa de búsqueda que, en un momento dado se utilizará en lugar de la PK. Para asegurar la *Unicidad* de cedula_ciudadania, en SQL utilizar la cláusula UNIQUE.

ISBN: 978-631-6557-89-6

- **Clave Primaria (PK)** 

La clave primaria (Primary Key – PK) de una tabla es aquella clave candidata que se escoge para identificar las filas de manea única.

La tabla siempre tiene una clave primaria, la cual evita filas duplicadas. Es recomendable que la clave primaria sea un solo atributo, en el peor de los casos, puede formarse por todos los atributos de la tabla; aunque, normalmente habrá un pequeño subconjunto de los atributos que haga esta función.

Características

- La clave primaria (PK) se define en la estructura de la tabla mediante DDL, y la tabla solo puede tener una PK.
- *Unicidad*: Único valor de PK en la tabla (NO SE REPITE el valor de la PK)
- *No nulidad* (*NOT NULL* - es obligatoria): No permite valores nulos
- El resultado de la búsqueda es una sola fila de la tabla
- La clave primaria (PK) puede formarse de un atributo (simple) o más (compuesta)

En representaciones de esquema y de valores de datos, la PK se subraya como una manera de identificarla, como se ilustra en los Ejemplos 2-5 y 2-6.

Ejemplo 2-5. Clave Primaria (PK)

estudiante (codigo, nombre_completo, edad, sexo)

Duplicidad

codigo	nombre_completo	edad	sexo
1001	LUIS VACA	20	H
2005	ANGEL PROAÑO	15	H
2008	ANA AVILA	17	M
1099	MARIA FLORES	13	M
1099	JORGE MARTINEZ	20	H
2035	MARIA FLORES	13	M

En el Ejemplo 2-5, si el valor 1099 de codigo se repite en dos filas, y si este atributo quiera ser declarado como PK, hay un error de duplicidad, no permitido en el modelo

ISBN: 978-631-6557-89-6

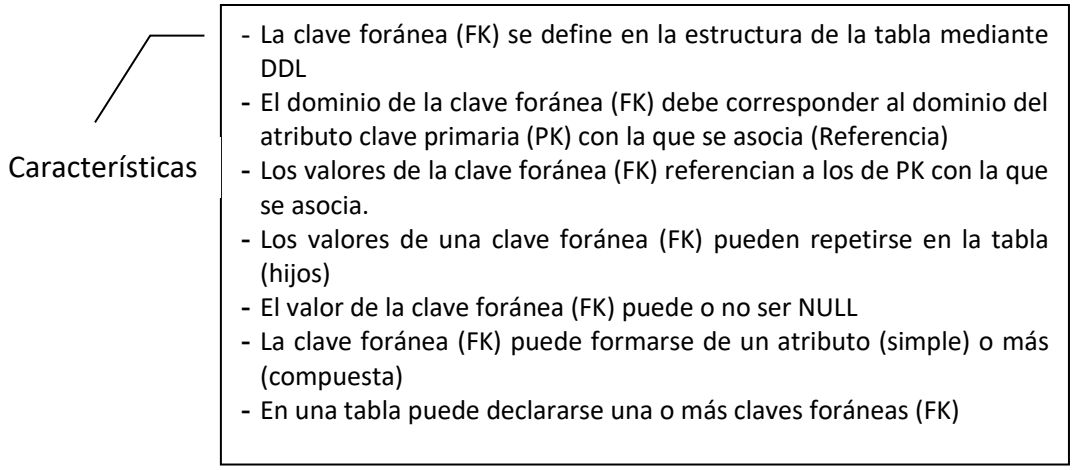
relacional. En cambio, la fila del codigo 2035 no se considera duplicada, indistinto que los valores de los otros atributos (cedula_ciudadania, edad, sexo) puedan coincidir.

Al elegir una clave primaria (PK) entre las claves candidatas, Connolly & Begg (2015) recomienda seguir algunas pautas que facilite la selección, las cuales se listan a continuación:

- Para una clave primaria (PK) compuesta, que ésta tenga el conjunto mínimo de atributos
- Los valores de la PK tengan la menor probabilidad de sufrir cambios.
- Una PK con el menor número de caracteres, principalmente para aquellas con atributos de tipo texto.
- Una PK con el menor valor máximo, principalmente para aquellas con atributos numéricos.
- Una PK que, preferentemente sea la más fácil de usar para los usuarios.

- **Clave Foránea (FK)**

La clave foránea (Foreign Key) también llamada externa o ajena, es un atributo o un conjunto de atributos de una tabla cuyos valores coinciden con los valores de la clave primaria de alguna otra tabla. Las claves foráneas representan *relaciones entre datos*, mediante la *referenciación* de una tabla con otra.



ISBN: 978-631-6557-89-6

Ejemplo 2-6. Clave Foránea (FK)

cliente (num_cliente, nombre_completo, ciudad_residencia, teléfono_contacto)

venta (id_venta, num_cli, valor_venta, fecha_venta)

PK

num_cliente	nombre_completo	ciudad_residencia	telefono_contacto
100	JORGE ARIAS	QUITO	0994410187
101	GISEL FLORES	AMBATO	0854123456
102	MARTHA SALINAS	RIOBAMBA	0890123456
103	JUAN CAMILO VARGAS	GUAYAQUIL	0987654321
104	MARIA VELEZ	QUITO	0987654320

FK

PK

id_venta	num_cli	valor_venta	fecha_venta
1	100	20	01-05-2024
2	100	15	15-12-2024
3	102	10	15-12-2024
4	103	25	7-10-2024

El atributo num_cli es clave foránea (FK) en la tabla venta, esta clave (FK) referencia al atributo num_cliente que es la clave primaria (PK) de la tabla cliente. Como se ilustra en el ejemplo, *no es necesario que sean los mismos nombres de los atributos* con los que se establece la referencia; pero si, *deben tener el mismo dominio*.

- **Clave de búsqueda**

Una tabla se conforma de varios atributos, de los cuales se elegirá la clave primaria; los demás atributos se constituyen en potenciales claves para realizar búsquedas, con la diferencia que éstas claves no tienen que ser declaradas como parte de la estructura de la tabla; sino, únicamente son utilizadas en las consultas y el resultado de éstas búsquedas son *n* filas. Una clave de búsqueda no requiere ser declarada mediante DDL.

Si en la tabla cliente del Ejemplo 2-6 se desea consultar los clientes por la ciudad de residencia QUITO (ciudad_residencia ='QUITO'); el resultado de la búsqueda serán dos filas.

ISBN: 978-631-6557-89-6

- **Clave subrogada**

La clave subrogada (*surrogate key*) también llamada clave sustituta, al igual que la clave búsqueda, es un concepto que no forma parte de la propuesta original del modelo relacional dada por Codd.

Una clave subrogada es una clave primaria artificial, no natural (no proviene del contexto del mundo real), creada por el diseñador de la base de datos para identificar de forma única cada fila de una tabla.

En la tabla cliente del Ejemplo 2-6 se ha definido id_venta como una clave subrrogada.

Características

- Se declara dentro de la estructura de la tabla mediante DDL, como una clave primaria (PK)
- Generada automáticamente por el sistema. Generalmente es un número entero ***autoincremental***, es decir, el valor se genera automáticamente.

2.4.2 Dependencia referencial Padre – Hijo

Desde un enfoque práctico, especialmente en los distintos DBMS, se utiliza la analogía **padre – hijo** para representar la *dependencia referencial* (asociación) entre una tabla con otra. De ésta manera, al asociarse **dos tablas a la vez**, una actuará como padre y la otra como hijo; restringiendo a que un padre pueda tener uno o varios hijos, como también que un hijo tenga solo un padre.

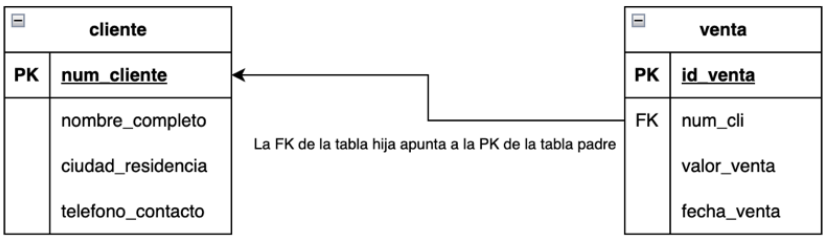
Características

- En la tabla padre se encuentra la clave primaria (PK)
- En la tabla hija se encuentra la clave foránea (FK dependiente).
- El equivalente conceptual a la analogía padre – hijo, es el tradicionalmente llamado maestro – detalle.
- Primero existe el padre, luego los hijos (se aplica desde la creación de una tabla hasta el ingreso de datos)
- No admite huérfanos (si se elimina un padre afecta a sus hijos)

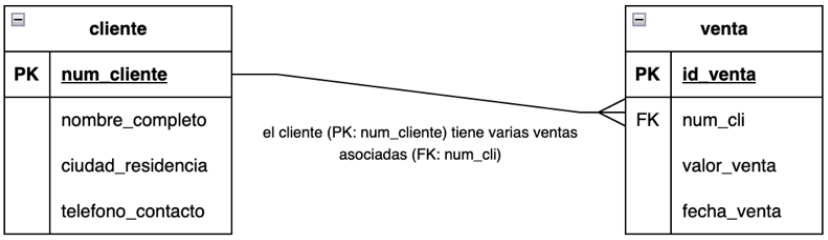
La analogía padre – hijo adoptada por diversos DBMS y herramientas de modelado de datos, es una forma pedagógica y visual para describir la dependencia referencial entre tablas.

ISBN: 978-631-6557-89-6

Figura 2-2. Símbolos para la representación de dependencia referencial



***a) Flecha**: la FK apunta (flecha) a la PK del padre*



***b) Pata de gallo**: La PK tiene varios (pata gallo) hijos en FK*

Elaborado por los Autores

Con el propósito de ilustrar los símbolos más utilizados para la representación e interpretación visual de la dependencia referencial (analogía padre – hijo), en la Figura 2-2 muestra dos diagramas de un mismo esquema relacional. Aunque ambos indica que existe una dependencia, en el diagrama *a)* con la flecha, la tabla hija tiene una FK que referencia a la PK de la tabla padre. Por otro lado, el diagrama *b)* con la pata de gallo, es un símbolo más utilizado para representar *cardinalidades*, concepto propio del modelo Entidad Relación; sin embargo, la interpretación de dependencia referencial del modelo relacional, sería que, la PK del padre tiene hijos en la tabla hija (valores en atributo FK).

Desde la perspectiva del diseño de base de datos, las cardinalidades se identifican a partir de un contexto de mundo real, y éstas serán **implementadas mediante restricciones referenciales** entre tablas.

2.5 Ejercicios: Modelo Relacional

1) Dadas las siguientes tablas (relaciones), realizar el diagrama de esquema correspondiente utilizando simbologías: a) patas de gallo, b) flecha

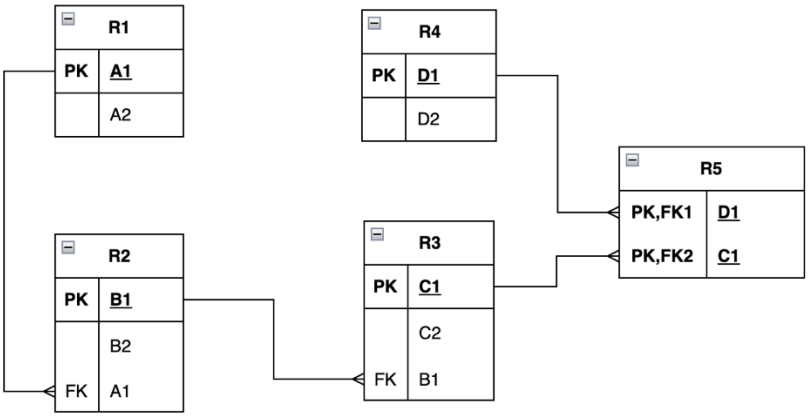
ISBN: 978-631-6557-89-6

R1(A1, A2) R2(B1, B2, A1) R3(C1, C2, B1) R4(D1, D2)

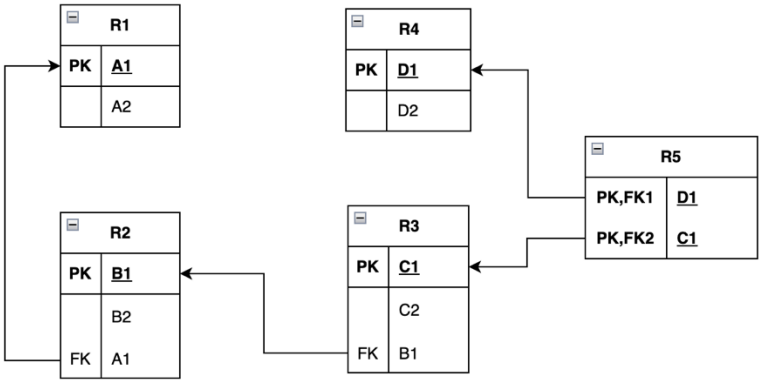
R5(C1, D1)

Solución:

a) uso de Pata de gallo



b) uso de Flecha



2) Dada el esquema relacional, a) identificar las claves primarias y foráneas y realizar una representación de tablas con datos del esquema dado que evidencia dependencia referencial b) realizar el diagrama de esquema (gráfico) correspondiente

ISBN: 978-631-6557-89-6

Solución:

Esquema relacional

RelacionLaboral (CodRL, Descripcion)

Empleado (CodEmp, CIEmp, NombreEmp, Direccion, Genero, CodRL)

a) tablas con datos

Padre RelacionLaboral

CodRL	Descripcion
1	Contrato tiempo completo
2	Nombramiento
3	Contrato tiempo parcial

Hijo

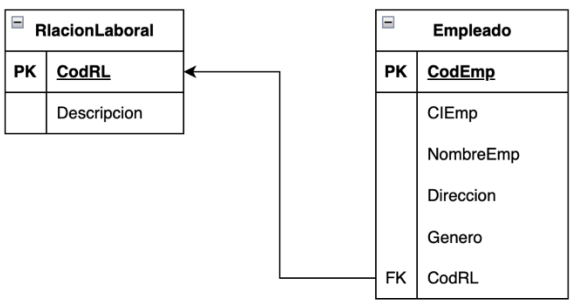
Empleado

La FK referencia al PK del padre

El padre tiene varios hijos

CodEmp	CIEmp	NombreEmp	Direccion	Genero	CodRL
0001	0609834270	ALBERTO GARCIA	DABC	M	1
0005	0662076895	VERONICA PLAZA	DXYZ	F	1
0064	1707773341	ANA GALEAS	DWSD	F	2
0052	0606543213	JOSE CAZAR	DRES	M	3
0643	1235431098	JOSE CAZAR	DFDS	M	2

b) diagrama de esquema (utilizado en diseño)



ISBN: 978-631-6557-89-6

El atributo CodRL de Empleado relaciona a cada empleado con la relación laboral que tiene en la empresa. CodRL es una clave foránea (FK) cuyos valores hacen referencia al atributo CodRL, clave primaria (PK) de RelacionLaboral. Se dice que un valor de clave foránea *referencia* a la fila que contiene el mismo valor en su clave primaria (fila o *tupla referenciada*).

NOTA:

La FK que está en la tabla hija referencia al PK del padre : Cada vez que se ingresa un valor en la FK (CodRL de Empleado) busca que exista la fila (PK) padre (CodRL de RelacionLaboral).

El padre tiene varios hijos : Padre único (CodRL =1) tiene varios hijos en la tabla hija

Al crearse las tablas, primero se creará RelacionLaboral y luego Empleado. Al ingresarse los datos primero se ingresarán en RelacionLaboral para luego hacerlo en la tabla hija

3) Dada la siguiente tabla con datos EMP_PROY, identificar la clave primaria

EMPLEADO	PROYECTO	TAREA
E01	100A	B-1
E01	200B	B-1
E05	100A	B-6
E05	200B	B-1
E05	150D	C-1

Solución:

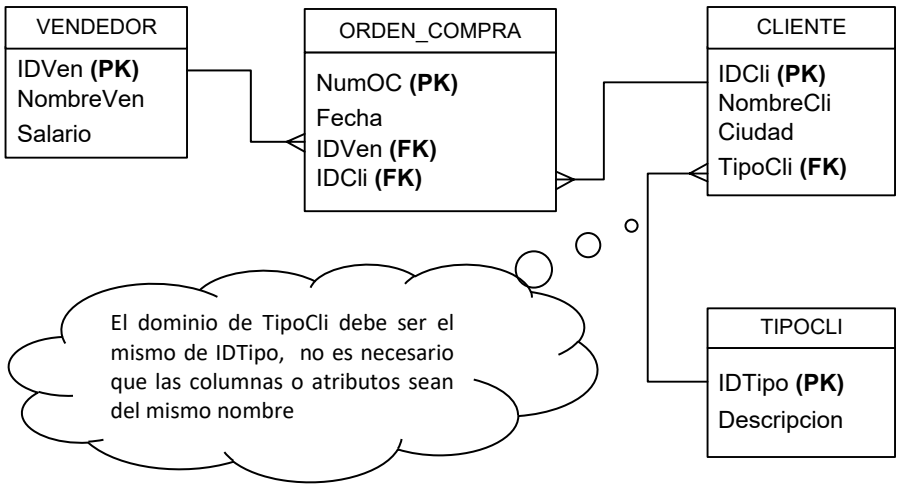
PK compuesta de los atributos: EMPLEADO + PROYECTO

4) Dadas las siguientes tablas (relaciones), identificar las claves primarias y foráneas, y realizar el diagrama de esquema correspondiente utilizando simbología patas de gallo

VENDEDOR (IDVen, NombreVen, Salario)
ORDEN_COMPRA (NumOC, IDVen, IDCli, Fecha)
CLIENTE (IDCli, NombreCli, Ciudad, TipoCli)
TIPOCLI (IDTipo, Descripción)

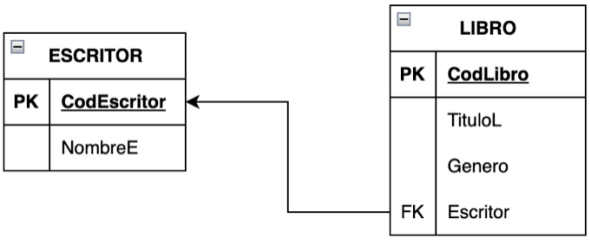
ISBN: 978-631-6557-89-6

Solución:



5) Proporcione un ejemplo de dos tablas con dependencia referencial padre - hijo y represente los datos en las tablas. Además, realice la definición de la estructura de las tablas especificando los dominios de cada atributo.

Solución:



ESCRITOR

CodEscritor	NombreE
E001	STEPHENIE MEYER
E002	JOHANNA LINDSEY
E003	ALE MENDELZON

LIBRO

CodLibro	TituloL	Genero	Escritor
L1	CREPÚSCULO	NOVELA	E001
L2	LUNA NUEVA	NOVELA	E001

ISBN: 978-631-6557-89-6

L3	CORAZÓN INDÓMITO	NOVELA	E002
L75	INTRODUCCIÓN A LAS BASES DE DATOS RELACIONALES	TÉCNICO - INFORMÁTICA	E003
L99	EL HEREDERO		E002

TABLA	COLUMNA	TIPO - TAMAÑO	PERMITE NULOS	VALIDACIÓN
ESCRITOR PK : CodEscritor	CodEscritor	Char(5)	Not null	CHECK : Primer carácter Letra y tres siguientes dígitos
	NombreE	Varchar(80)		Not null

TABLA	COLUMNA	TIPO - TAMAÑO	PERMITE NULOS	VALIDACIÓN
LIBRO PK : CodLibro FK: Escritor	CodLibro	Char(3)	Not null	
	TituloL	Varchar(150)	Not null	
	Genero	Varchar(80)		
	Escritor	Char(5)	Not null	CHECK : Primer carácter Letra y tres siguientes dígitos

ISBN: 978-631-6557-89-6

CAPÍTULO 3

3 PRINCIPIOS BASICOS DEL DISEÑO DE BASE DE DATOS

3.1 Introducción al diseño de base de datos

El diseño de bases de datos se refiere a la construcción de un modelo lógico y una estructura física de base de datos para un entorno de aplicación determinado (Huawei Technologies Co., 2022).

En el diseño de bases de datos se identifica cada elemento de dato y su función dentro del sistema, la clave del proceso de diseño es un esquema de base de datos, que *describa la estructura de los datos en una forma clara y estable*, que *fielmente refleje los requerimientos* del sistema y que *permita cambios razonables*.

El diseño de base de datos es importante por las razones que se listan a continuación:

- Es la tarea más importante en el desarrollo de eficaces aplicaciones de bases de datos
- Representa la visión (percepción) que los usuarios poseen de los datos
- Es el núcleo de todo trabajo subsecuente en el desarrollo de la base de datos y de sus aplicaciones
- Una base de datos implementada bien diseñada tiene mayor esperanza de vida aún en un medio ambiente dinámico, que una base con un diseño pobre.

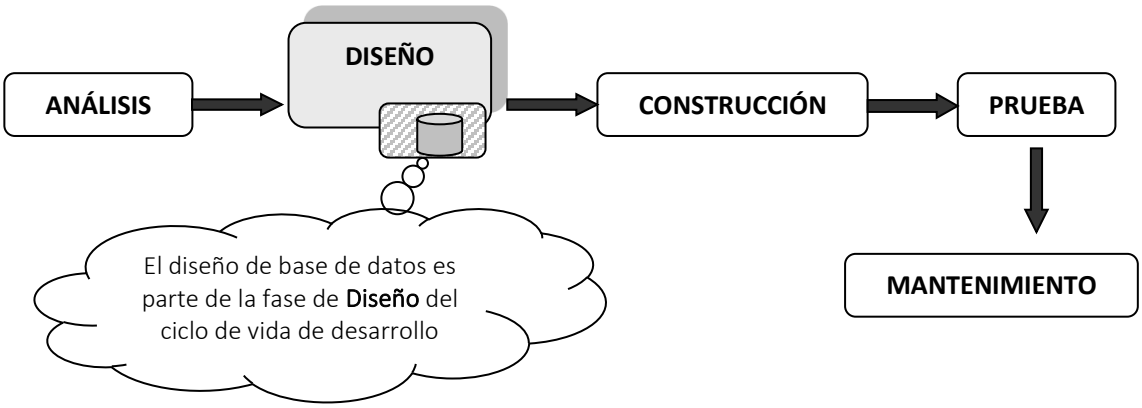
Es importante ubicar el diseño de una base de datos dentro del Ciclo de Vida clásico de desarrollo de software, para este propósito, la Figura 3-1 presentan las cinco fases, que son: el análisis, diseño, construcción, prueba y mantenimiento.

La fase de diseño tiene como propósito principal convertir los requerimientos recolectados en la fase de análisis en una solución técnica detallada, es decir, en una arquitectura del sistema que especifique *cómo* será construido el software y *cómo* funcionará cada uno de sus componentes. En esta línea, la base de datos como un componente que define la estructura, organización e integridad de la información necesaria

ISBN: 978-631-6557-89-6

para los procesos de negocio; requiere de un diseño que caracterice correctamente las necesidades de datos a partir de los requerimientos recolectados desde la fase de análisis.

Figura 3-1. Ciclo de vida clásico de desarrollo de Software



Elaborado por los Autores

Las metodologías de diseño de bases de datos y las metodologías de ingeniería de software están interrelacionadas, ya que estas actividades están estrechamente vinculadas. El diseño de bases de datos se considera una disciplina con métodos y técnicas propias (Elmasri & Navathe, 2016; Huawei Technologies Co., 2022).

En el año 1978, expertos en bases de datos de más de 30 países se reunieron en Nueva Orleans, EE. UU., dando lugar a la metodología de diseño de Nueva Orleans, conocida por sus siglas en inglés NODM (New Orleans Database Design Methodology); constituyéndose en un método de diseño de base de datos con fases que incluyen el análisis de requerimientos, modelado de datos y validación estructural (Huawei Technologies Co., 2022).

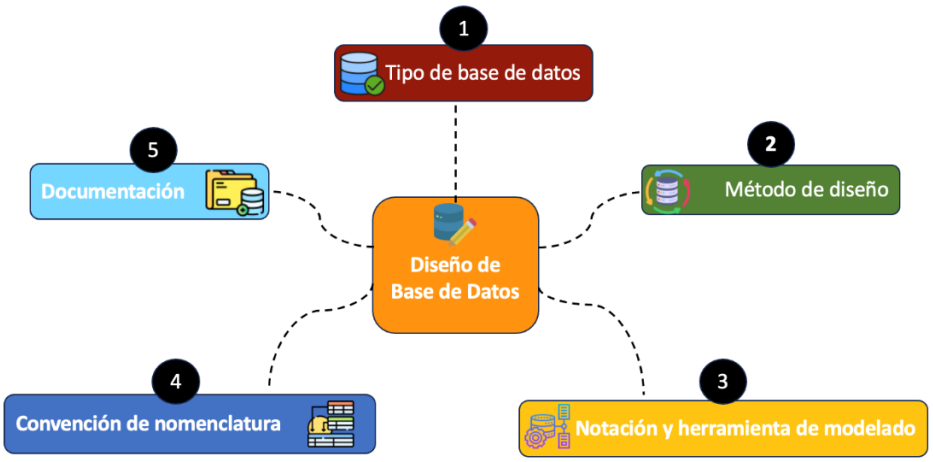
Además de la NODM, existen otras propuestas para el diseño de bases de datos; las cuales han integrado enfoques conceptuales-lógicos-físicos, tales como: modelo E-R de Peter Chen (1976), IDEF1X (estándar ANSI: Integration Definition for Information Modeling, 1992), método ERD de Richard Barker (Entity - Relationship Diagram Method, 1989) y la Normalización de Edgar F. Codd (1970). Todas ellas se constituyen en métodos

y técnicas que se utilizan en el diseño de bases de datos, y que se describirán en detalle en secciones posteriores.

3.1.1 Factores técnicos que considerarse en el diseño de base de datos

En un contexto más amplio del diseño de base de datos, es indispensable considerar una serie de factores técnicos y metodológicos que garanticen la correcta implementación del modelo de base de datos en un entorno real. Los factores técnicos que se muestran en la Figura 3-2 son complementarios, pero influyen directamente en la calidad del diseño.

Figura 3-2. Factores técnicos de diseño de base de datos



Elaborado por los Autores

1) Determinar el tipo de base de datos que se desea implementar

A partir de la fase análisis, es importante determinar el tipo de base de datos que se desea implementar; lo que implica, con la selección de un DBMS. Tomando como referencia la clasificación de la Tabla 1-6, se puede elegir entre DBMSs de los grupos: SQL, No SQL o New SQL. Este texto está dirigido para bases de datos relacionales, por tanto, se contemplan DBMSs del tipo SQL y New SQL.

2) Definir el método de diseño de base de datos

Un método de diseño de base de datos define cómo se llevará a cabo el diseño, los pasos a seguir, en qué orden y con qué profundidad. Estos aspectos están determinados por el enfoque adoptado y por las fases que conforman el proceso de diseño.

ISBN: 978-631-6557-89-6

El enfoque establece la dirección de análisis, entre los principales enfoques, tenemos: Bottom-Up (ascendente) y Top-Down. (descendente). Mientras que, el proceso aporta la *estructura sistemática. El enfoque y el proceso para el diseño de base datos se trata en la siguiente sección del texto.*

3) Precisar la notación y seleccionar la herramienta de modelado de datos

Dependiendo el método (enfoque y proceso) a utilizarse para el diseño de base de datos se puede precisar la notación.

La notación también denominada lenguaje de representación o convención gráfica, corresponde al conjunto de símbolos, sintaxis y reglas formales que permiten expresar visualmente el modelo de la base de datos. La notación forma parte de las **técnicas de modelado** utilizadas al aplicar un método.

Por ejemplo:

Método: El método de diseño de base de datos se conforma con el *enfoque* Top-Down y el *proceso* basado en modelado de datos (conceptual, lógico y físico).

Técnica: Modelo Entidad Relación para diseño conceptual

Notación: DER con notación de Chen

Es importante utilizar una herramienta especializada que apoye el diseño de base de datos, más conocida como herramienta de modelado de datos, ya que, garantiza la coherencia y correcto diseño; reduciendo la posibilidad de errores conceptuales o inconsistencias que pueden surgir sin el apoyo de la herramienta. Otro aspecto a considerar, es la estandarización del proceso de diseño; esto se logra con herramientas que incluyan notaciones formales (estándar), por ejemplo, Chen, Barker o IDEF1X, entre otras. Las herramientas de modelado también facilitan la documentación y la trazabilidad del modelo.

En resumen, el uso de una herramienta de modelado de datos para el diseño de la base de datos es esencial, ya que, asegura la calidad del modelo, estandariza el proceso, facilita la documentación técnica y mejora la implementación de la base de datos.

ISBN: 978-631-6557-89-6

4) Establecer la convención de nomenclatura para los objetos de la base de datos

En el diseño de base de datos es crucial establecer una norma que incluya las convenciones de estilo para nombrar las tablas, atributos, claves y restricciones. La importancia de la convención de nomenclatura radica por las siguientes razones:

- *Asegura consistencia interna*. Con una norma coherente se evita el caos o estilos personales.
- *Mejora la legibilidad*. Un nombre claro permite comprender el propósito de una tabla o atributo sin consultar documentación adicional.
- *Facilita el mantenimiento*. Un diseño ordenado es más fácil de depurar y revisar incluso por nuevos miembros del equipo.
- Reduce ambigüedades. Evita que dos objetos tengan nombres parecidos o poco claros.
- *Claridad semántica*. Los nombres reflejan reglas de negocio y conceptos del dominio de forma uniforme.
- *Aumentan la interoperabilidad*. Muchas herramientas (ORM, generadores de código, ETL, modeladores) funcionan mejor cuando hay un estándar claro.

Tabla 3-1. Convenciones comunes para nombrar objetos de la base de datos

Convención	Objeto de la base de datos	Ejemplo
• Nombre en singular Usado frecuente en modelado conceptual y en la mayoría de estándares modernos	Tabla	Cliente, Producto
• Nombre en plural Uso frecuente en SQLServer y modelos lógicos tradicionales		Clientes, Productos
• Nombre en minúscula que mantenga la semántica de la tabla	Atributo (columna)	fecha_venta, fechaVenta nombre_cliente, nombreCliente
• Uso de prefijo	Atributo (columna)	id_venta (clave subrrogada)
	Restricciones	pk_venta (clave primaria) fk_producto (clave foránea)

ISBN: 978-631-6557-89-6

	Indice, Secuencia, Dominio, Vista	idx_<tabla>_<columna> idx_estudiante_apellido
• Uso de nombre compuestos	Tabla intermedia	Usuario_Rol UsuarioRol

Elaborado por los Autores

Ejemplos de convenciones de nomenclatura comúnmente utilizadas se presentan en la Tabla 3-1. También se pueden adoptar estilos de nombrado que provienen de las buenas prácticas establecidas en la ingeniería de software y de los estándares utilizados en lenguajes de programación. En la Tabla 3-2 se presentan estilos como PascalCase, camelCase y otros.

Tabla 3-2. Estilos de nomenclatura para nombrar objetos de la base de datos

Estilo	Ejemplo de uso frecuente	Objeto de la base de datos	Ejemplo
camelCase	Java, C# Entornos: OOP /ORM	Atributo (columna)	fechaVenta idCliente
PascalCase	C# SQL Server	Tabla Vista	FechaVenta IdCliente
snake_case	PostgreSQL, MySQL, Oracle Python	Tabla Atributo (columna) (uso general)	fecha_venta id_cliente
UPPER_SNAKE_CASE	DBMS que facilite su aplicación	Tabla Vista	CLIENTE VENTA

Elaborado por los Autores

5) Documentación

El diseño de la base de datos debe documentarse, de manera que, los usuarios técnicos (administradores y diseñadores de bases de datos) puedan comprender, implementar y mantener de forma adecuada una base de datos. La documentación es esencial del manual técnico, asegura la trazabilidad del proceso de diseño e informa sobre

ISBN: 978-631-6557-89-6

la estandarización del modelo de datos. Los documentos técnicos de la base de datos deben estar siempre disponible para la administración y futuros cambios de la base de datos.

La documentación de una base de datos debe incluir, principalmente: el método de diseño, notación empleada, los esquemas de base de datos resultantes del modelado de datos, reglas de negocio y el diccionario de datos (componente esencial).

3.2 Método de diseño de base de datos

En la literatura, el diseño de bases de datos se define por sí mismo como un proceso, el cual, a su vez, es entendido como un método. Sin embargo, en este texto se plantea que el método de diseño está determinado por el enfoque adoptado y por las fases que conforman el proceso de diseño.

3.2.1 Enfoques para diseño de base de datos

Entre los enfoques de estrategias para el diseño de bases de datos, tenemos: Bottom-Up (ascendente) y Top-Down. (descendente) como los enfoques principales; además, se destacan los enfoques Inside-Out (adentro hacia afuera) y el Mixed (mixto) (Connolly & Begg, 2015; Elmasri & Navathe, 2016).

- **Bottom-Up**

El Enfoque Bottom-Up comienza en el nivel fundamental de los atributos (es decir, las propiedades de las entidades y las relaciones), que, mediante el análisis de las asociaciones entre atributos, éstos se van agrupando (Tablas). Este enfoque es adecuado para el diseño de bases de datos simples con un número relativamente pequeño de atributos. Sin embargo, este enfoque se vuelve difícil al aplicarlo al diseño de bases de datos más complejas con un mayor número de atributos, donde es difícil establecer todas las dependencias funcionales entre ellos.

El proceso de normalización representa un enfoque Bottom-Up para el diseño de bases de datos. La normalización implica la identificación de los atributos necesarios y su posterior agregación en tablas normalizadas basadas en las dependencias funcionales entre ellos.

ISBN: 978-631-6557-89-6

- **Top-Down**

El enfoque Top-Down comienza con el desarrollo de modelos de datos que contienen algunas entidades y relaciones de alto nivel, y luego aplica refinamientos Top-Down sucesivos para identificar entidades, relaciones y atributos asociados de nivel inferior. Este enfoque es más adecuada para el diseño de bases de datos complejas con un alto número de atributos.

El enfoque Top-Down se ilustra, por ejemplo, mediante los conceptos del modelo Entidad Relación (MER), comenzando con la identificación de entidades y relaciones entre ellas que son de interés para la organización.

- **Inside-Out**

El enfoque Inside-Out está relacionado con el enfoque de Bottom-Up, pero se diferencia en que primero identifica un conjunto de entidades principales y luego se expande para considerar otras entidades, relaciones y atributos asociados con las identificadas inicialmente

- **Mixed**

El enfoque mixto utiliza tanto el enfoque de Bottom-Up como el de Top-Down para varias partes del modelo antes de finalmente combinarlas todas.

3.2.2 Proceso de diseño de base de datos

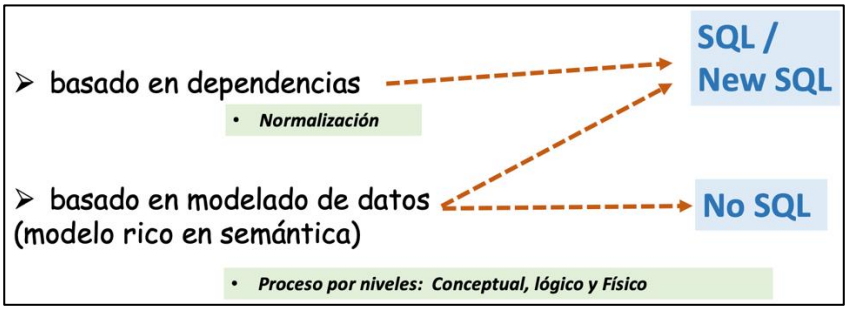
El proceso de diseño de base de datos es sistemático que abarca desde la identificación de los requerimientos hasta la implementación de la base de datos, con el propósito de garantizar la integridad, coherencia y eficiencia en el almacenamiento y recuperación de datos en un entorno determinado (Gobert et al., 2021).

Para el diseño de una base del tipo SQL o New SQL se opta por procesos basados en dependencias o uno basado en modelado de datos. Mientras que, para bases de datos No SQL se puede realizar diseños basado en modelado de datos, como se ilustra en la Figura 3-3.

ISBN: 978-631-6557-89-6

En resumen, la elección del enfoque y el proceso depende del tipo de base de datos, así tenemos, para bases de datos relacionales (tipo SQL), de acuerdo con la complejidad de la base de datos y la experiencia del diseñador se puede optar por un proceso basado en dependencias o uno por modelado de datos.

Figura 3-3. Procesos para el diseño de bases de datos



Elaborado por los Autores

3.3 Proceso de diseño basado en el modelado de datos

El proceso de diseño basado en el modelado de datos permite, de forma sistemática, representar y estructurar los datos de un dominio específico mediante un modelo. Esta orientación se fundamenta en niveles de abstracción que, en la práctica se tratan como *Fases de diseño*; los cuales, permiten separar progresivamente la visión del usuario de los detalles técnicos de implementación, facilitando la comprensión de los requerimientos del mundo real, la definición precisa de entidades y relaciones, y la transformación progresiva de dichos modelos en estructuras implementables en un sistema de gestión de bases de datos (DBMS) (Connolly & Begg, 2015; Elmasri & Navathe, 2016; Silberschatz et al., 2014) .

3.3.1 Proceso de diseño a tres niveles

El diseño de una base de datos se concibe como un **proceso de diseño a tres niveles** o fases, compuesto por el diseño conceptual, lógico y físico. La separación en tres niveles garantiza claridad, independencia y coherencia en el diseño.

La Figura 3-4 resume el proceso de diseño basado en el modelado de datos, el cual se componen de tres fases principales de diseño, el conceptual, el lógico y el físico. En cada fase del diseño de la base de datos, el *modelo* establece la abstracción, la *estructura*

ISBN: 978-631-6557-89-6

define sus componentes internos y el *esquema* constituye la representación formal de dicha estructura.

- **El modelo**: Define qué se está representando en un nivel (conceptual, lógico o físico).
- **La estructura**: Es el contenido interno de ese modelo (elementos que conforman el modelo y su organización)
- **El esquema**: Es la representación formal de la estructura, ya sea mediante diagramas, descripciones detalladas o scripts (ejemplo: SQL).

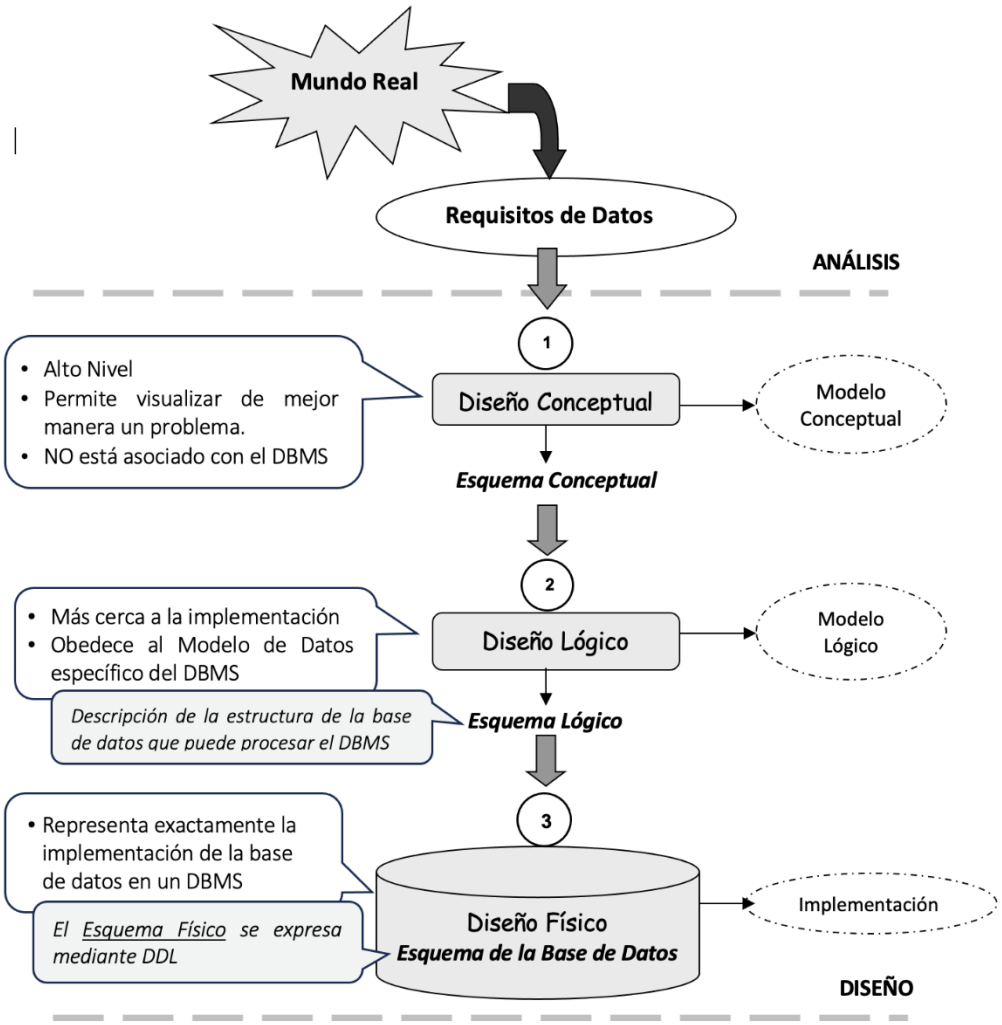
1 El **diseño Conceptual** permite visualizar de mejor manera un problema. Además, este diseño NO está asociado con la implementación (DBMS y todas las consideraciones físicas), sino que está más cercano a la realidad.

El propósito de esta fase de diseño es obtener un ***Esquema Conceptual***, es una descripción de alto nivel de la estructura de la base de datos, independiente del DBMS que se vaya a utilizar para manipularla. Para conseguir un Esquema Conceptual es necesario de un ***Modelo Conceptual***, que permita generar esquemas de este tipo.

Entre los principales modelos utilizados en el diseño conceptual tenemos los de tipo semántico (ver Tabla 1-5), como son: Modelo Entidad Relación y Modelo Orientado a Objetos – conceptual. Por el hecho de aplicar modelos conceptuales a esta fase también se la conoce como modelamiento o **modelado Conceptual**.

ISBN: 978-631-6557-89-6

Figura 3-4. Proceso de diseño de bases de datos basado en modelos - 3 niveles de abstracción



Elaborado por los Autores

2 El **diseño lógico** se acerca más a la implementación, integrando consideraciones para la plataforma específica en cuestión. Es decir, el diseño lógico obedece a un *Modelo de Datos* específico.

El propósito de esta fase de diseño es obtener un ***Esquema Lógico***, es una descripción de la estructura de la base de datos que puede procesar un DBMS específico, por ejemplo, Modelo Relacional.

ISBN: 978-631-6557-89-6

En el caso del modelo lógico relacional, este tiene su propia herramienta, existe un proceso que sirve para verificar que hemos aplicado bien el modelo, y en caso contrario, corregirlo para que sea así. Este proceso se llama *normalización*, y también es bastante mecánico.

Por el hecho de aplicar modelos Lógicos de datos a esta fase también se la conoce como Modelamiento o **modelado Lógico**.

3 El **diseño físico** es una especificación tal que representa exactamente la implementación de la base de datos en un DBMS. A esta fase se la conoce y trata también como **Implementación**.

El diseño físico parte del Esquema Lógico y da como resultado el *esquema de la base de datos* (alguna literatura lo llama Esquema Físico), que es una descripción de la implementación de una base de datos. Este Esquema se expresa mediante DDLs (Lenguajes de Definición de Datos).

- **Resumen:**

El proceso de diseño a tres niveles se puede resumir de la siguiente manera:

Partiendo del mundo exterior (aquel que interesa representar en los datos), el cual se lo debe aprender, comprender y conceptualizar para transformarlo en un conjunto de ideas y definiciones que supongan una imagen fiel del comportamiento del mundo real. A esta imagen del mundo exterior lo llamaremos "***Modelo Conceptual***.

Una vez definido el modelo conceptual, éste se ha de transformar en una descripción de datos, atributos y relaciones que denominaremos "***Modelo Lógico de datos"***.

Por último, el Modelo lógico habrá que traducirlo a estructuras almacenables en soportes físicos (DBMS).

El diseño de bases de datos es posible hacerlo mediante el sentido común (enfoque intuitivo). Para el experto diseñador derivar entidades o registros de tipo conceptual de un grupo de datos se puede hacer intuitivamente. Sin embargo, tal intuición no siempre surge

ISBN: 978-631-6557-89-6

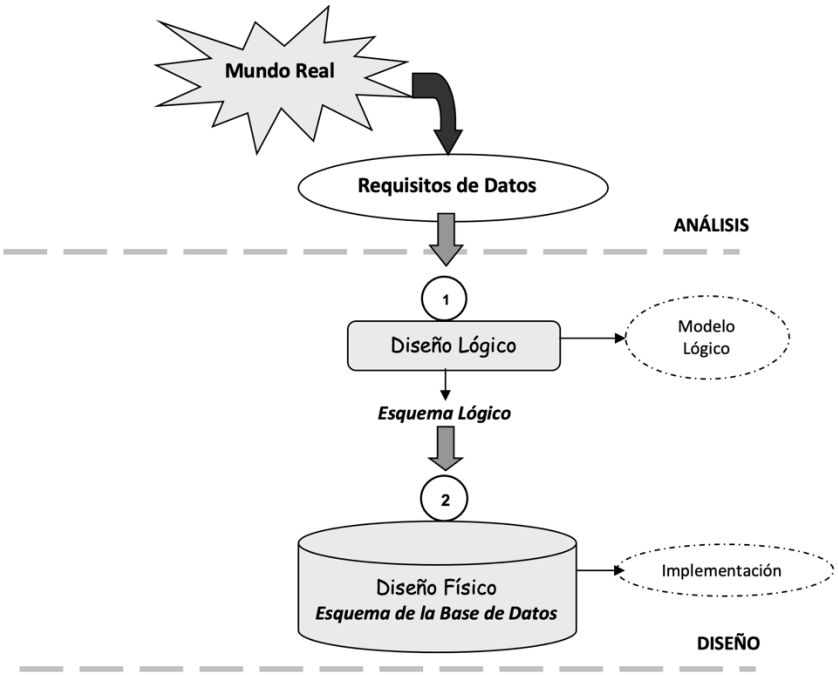
espontáneamente en los principiantes, especialmente cuando el diseño es muy complejo y es justamente cuando se hace necesario utilizar como apoyo algunas herramientas de diseño (herramientas de modelado).

3.3.2 Proceso de diseño a dos niveles

La pregunta frecuente es, *¿es obligatorio seguir la secuencia de tres fases del proceso de diseño*?, la respuesta es **NO.**

Es posible realizar el diseño de una base de datos a partir del modelo lógico, omitiendo el diseño conceptual, como se ilustra en el Figura 3-5. Este proceso a dos niveles, comúnmente aplicado en la práctica; si bien es cierto, simplifica el proceso pero puede sacrificar claridad o consistencia semántica del modelo. Por esta razón, debe aplicarse principalmente, cuando los requisitos de datos estén bien definidos o sean pequeños.

Figura 3-5. Proceso de diseño de bases de datos basado en modelos - 2 niveles de abstracción



Elaborado por los Autores

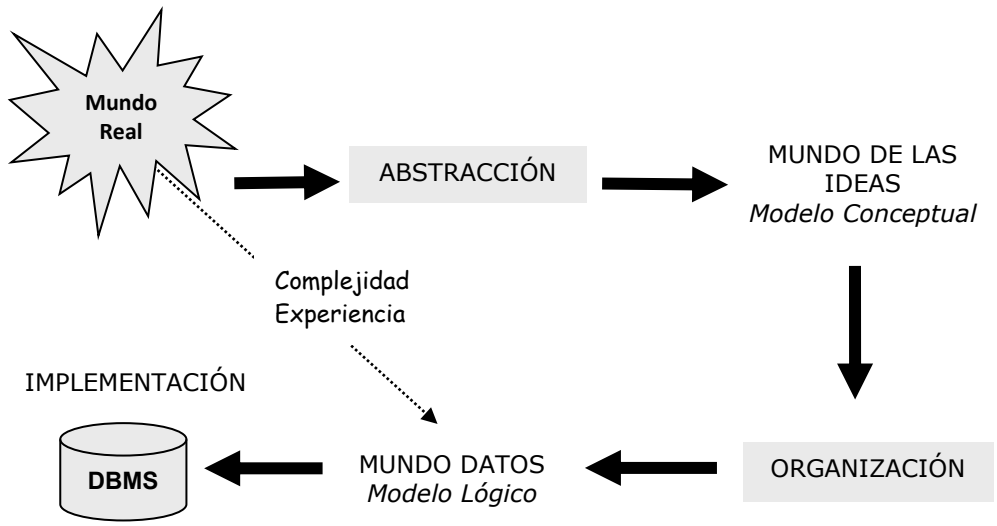
ISBN: 978-631-6557-89-6

En caso de **NO *seguir la secuencia tradicional de las tres fases del proceso de diseño***, conceptual, lógico y físico (3 niveles); es necesario tomar en cuenta las siguientes consideraciones:

- Proceso de diseño a dos niveles, implica pasar del Mundo Real directamente al modelado o diseño Lógico, para lo cual, es necesario considerar dos variables: **Complejidad y Experiencia**
  - *Experiencia*: Es frecuente que el diseñador de la base de datos tenga mucha experiencia en trabajos de este tipo, lo que le faculta NO realizar el modelado conceptual e ir directamente al modelado lógico. Sin embargo, esto es recomendable sólo cuando el Mundo Real no es complejo, caso contrario es necesario efectuar el modelado conceptual pues ayuda a aclarar todo el panorama del problema.
  - *Complejidad*: Cuando el Mundo real tiene cierto grado de complejidad, o el diseñador no tiene experiencia en el campo del diseño, es vital iniciar con el Modelado conceptual. El modelado lógico ayudará posteriormente para cambios menores, sobre todo en aspectos relevantes para la construcción de las aplicaciones.

Lo expuesto en los párrafos anteriores se resume en la Figura 3-6.

Figura 3-6. Resumen de la descripción del proceso de diseño de Bases de Datos



Elaborado por los Autores

ISBN: 978-631-6557-89-6

- **ERROR:** Pasar directamente del Mundo Real hacia la Implementación

  Es frecuente desear pasar por alto algunos de los pasos del proceso de diseño inclusive el Modelado o diseño Lógico, y crear directamente en el DBMS las bases de datos, esto es un gran ERROR. Siempre será mejor seguir todos las fases del diseño, esto asegura la calidad en la base de datos y ahorrará (con toda seguridad) mucho tiempo más adelante. Sobre todo, si alguna vez se tiene que hacer modificaciones para corregir errores o para implementar nuevas características, algo que sucede con mucha frecuencia.

- **Resumen:**

  El proceso de diseño a dos niveles se recomienda en proyectos de pequeña escala o cuando el dominio de los datos está claramente definido; en sistemas complejos, la ausencia del modelo conceptual puede provocar ambigüedades y errores estructurales. Además, seguir el proceso de diseño a tres niveles, facilitará la documentación que permite revisiones y futuros mantenimientos al sistema de base de datos.

ISBN: 978-631-6557-89-6

CAPÍTULO 4

4 DISEÑO CONCEPTUAL: MODELO ENTIDAD RELACIÓN

✶ **Descripción**

El diseño conceptual es completamente independiente de los aspectos de implementación, como puede ser el DBMS que se vaya a usar, los programas de aplicación, los lenguajes de programación, el hardware disponible o cualquier otra consideración física. Se da tratamiento únicamente a una realidad específica (Mundo real), sin considerarse los aspectos de la aplicación, los cuales serán tomados en cuenta en el diseño lógico.

> *El diseño conceptual es una tarea humano-dependiente, en el sentido que requiere de habilidades que son muy difíciles de automatizar. El diseñador debe analizar la realidad bajo modelamiento, documentar los hechos relevantes para satisfacer un conjunto de requerimientos, y complementar el documento de especificación de requerimientos una vez que obtiene nueva información a través del proceso de diseño* (Elmasri & Navathe, 2011, 2016).

Esta fase del proceso de diseño de base de datos está muy ligada al *Análisis* del ciclo de vida clásico (Figura 3-1), ya que se alimenta y depende del análisis de requerimientos para un correcto diseño conceptual. Muchos elementos serán descubiertos en esta fase que contribuirán y complementarán a los requerimientos previamente establecidos.

✶ **Actores**

Diseñador, Usuario.

✶ **Entrada**

Documento de Especificación de Requerimientos**.**

✶ **Salida**

El modelo conceptual que representa el *esquema conceptual* de la base de datos, describiendo su estructura de manera abstracta y libre de detalles lógicos o físicos.

ISBN: 978-631-6557-89-6

✶ **Modelos y Técnicas para utilizar**

El modelo entidad relación, en la actualidad es el modelo de datos más popular para el diseño conceptual de bases de datos.

✶ **Actividades**

El Diseñador crea el esquema conceptual mediante el mapeo del DER (Diagrama Entidad Relación, asociando las definiciones de Reglas de negocio y restricciones de integridad.

No hay un procedimiento claro y universal para crear el esquema conceptual mediante un DER, aunque sí se pueden dar algunas pautas generales, tales como las siguientes:

a) Hablar con el cliente e intentar dejar claros los parámetros y objetivos del problema o mundo real a modelar. Por supuesto, tomar nota de todo.
b) Estudiar el planteamiento del problema para:
    - Identificar los conjuntos de entidades útiles para modelar el problema.
    - Identificar los conjuntos de relaciones y determinar su grado, cardinalidad y tipo.
c) Realizar el primer DER
d) Identificar atributos y dominios para los conjuntos de entidades y relaciones.
e) Seleccionar los identificadores para cada uno de los conjuntos de entidades.
f) Verificar que el esquema resultante cumpla el planteamiento del problema. Si no es así, se vuelve a repasar el procedimiento desde principio (a).
g) Seguir con la siguiente fase del proceso de diseño de la base de datos: Diseño Lógico.

4.1 Modelo Entidad Relación

El Modelo Entidad Relación (MER) fue introducido por Peter Chen en 1976, y está establecido por conceptos que permiten describir la realidad mediante un conjunto de representaciones gráficas (DER) y lingüísticas. En las décadas siguientes han aparecido

ISBN: 978-631-6557-89-6

otros autores que han extendido el modelo original de Chen incrementando sus capacidades.

El Modelo Entidad Relación originalmente NO fue propuesto como un Modelo de Datos a ser implementado en la práctica por un DBMS, sino como una *herramienta conceptual* útil para el diseño de base de datos; razón por la cual, también se lo categoriza como un ***Modelo Conceptual***

Los modelos semánticos logran sus metas usando abstracción, y tienen una visión más navegacional, por esta razón, son considerados como herramientas para el diseño de esquemas. En este sentido, el modelo Entidad Relación es un modelo semántico.

Como ya se lo ha dicho, el propósito de modelo Entidad Relación es simplificar el diseño de bases de datos, por lo que permite formular la lógica de la empresa (Mundo real) y expresarla gráficamente mediante DER (Diagramas Entidad Relación). Este modelo es de fácil conversión a otros modelos de datos como el relacional, jerárquico, orientado a objetos y otros.

En esencia, el modelo Entidad Relación se basa en una percepción de un mundo real que consiste en un conjunto de Objetos llamados ENTIDADES y de las RELACIONES entre estos OBJETOS. A cada entidad se asocia un conjunto de ATRIBUTOS que describen al objeto.

Al Modelo Entidad Relación tradicional de Peter Chen se han incorporado conceptos adicionales y ha dado lugar a lo que se conoce como Modelo Entidad Relación Extendido (MERE). Estos nuevos conceptos, principalmente son: Relaciones recursivas, jerarquías de Generalización/Especialización y la Agregación.

El presente texto conserva la esencia del enfoque entidad-relación formulado por Chen y, de manera complementaria, integra las propuestas y las notaciones desarrolladas por los autores: Elmasri & Navathe (2016), De Miguel, Piattini & Marcos (2000) y Silberschatz, Korth & Sudarshan (2014). De igual manera, se han incorporado fuentes que amplían y contextualizan el diseño de base de datos, las cuales se encuentran en el apartado “Bibliografía”.

ISBN: 978-631-6557-89-6

4.1.1 Terminología y notaciones

No existe una terminología estandarizada para los conceptos MERE, por lo que se usará la más difundida. Es importante asociar los términos utilizados en el Modelo Entidad Relación (MER) con sus equivalentes en el Modelo Relacional, para este propósito, en la Tabla 4-1 se presenta dichas equivalencias de términos.

Hay diversas notaciones alternativas para la visualización de los Diagramas Entidad Relación (DER), en la Figura 4-1 se presentan algunas de las más populares. Como ya se indicó en párrafos anteriores, en este texto se trabaja con el Modelo Entidad Relación Extendido (MERE) acogiendo lo mejor de varias propuestas, y su conversión es hacia el Modelo Relacional. En este sentido, la notación para los conceptos clave del modelo Entidad Relación y su representación mediante diagramas (Diagrama Entidad Relación - DER) que se utiliza en este texto se resume en la Figura 4-2.

Tabla 4-1. Términos utilizados en MER y sus equivalentes en el Modelo Relacional

Modelo Entidad Relación	Modelo Relacional
Conjunto Entidad (Tipo Entidad) / Entidad	Relación / Tabla
Entidad / Ocurrencia Entidad /Instancia	Tupla / Fila
Atributo	Columna / Atributo
Relación	Dependencia Referencial
Número de ocurrencias/ instancias - Cardinalidad de relación	Número de Tuplas
---	*Grado de una relación* (Número de columnas que posee la relación)
Grado de una relación (Número de entidades que asocia la relación. Ejm. Binaria, ternaria,… N-arias)	Binaria - dos tablas bajo la dependencia referencial (Padre – Hijo)
Restricción en la relación entre entidades: Cardinalidad de relación: 1 a N / 1 a 1 / N a 1 / N a N	Referencia Padre - Hijo
Identificador	Clave primaria

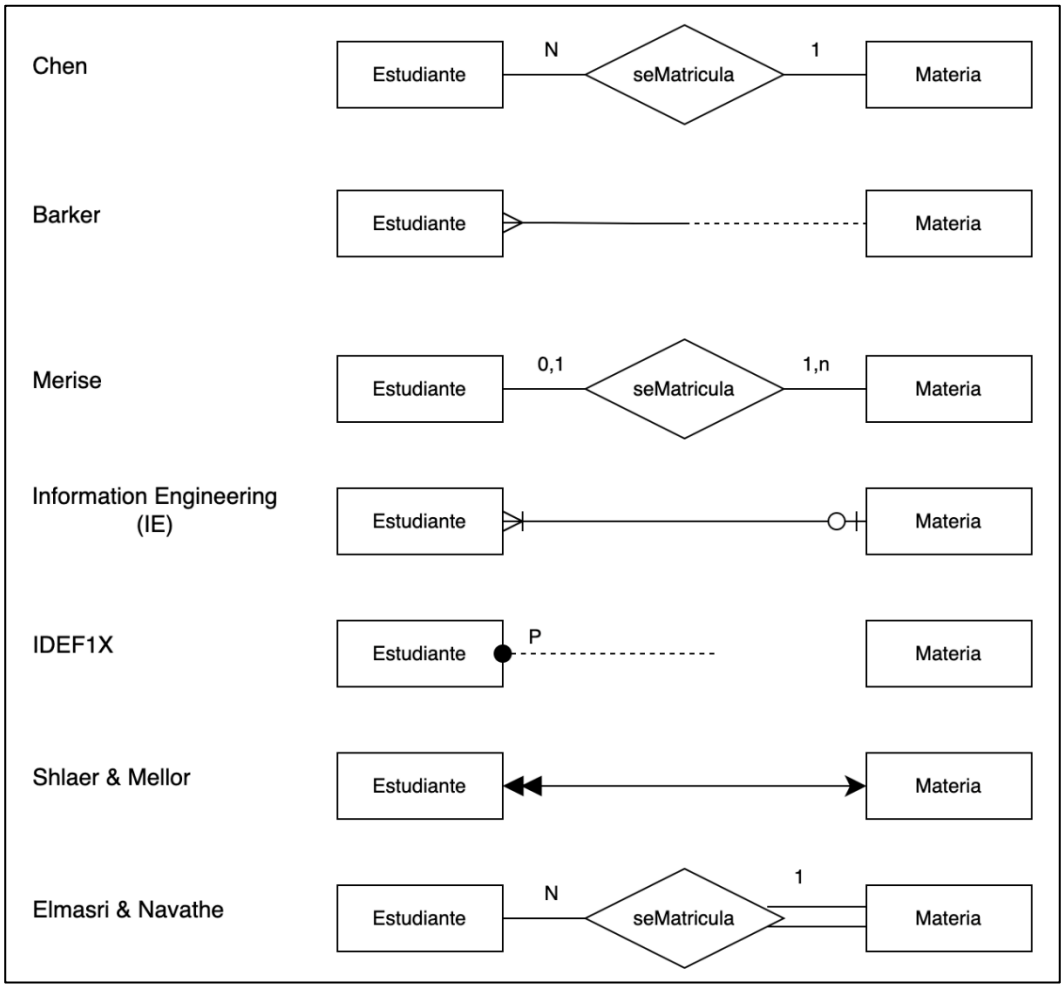
ISBN: 978-631-6557-89-6

Dominio	Dominio

Elaborado por los Autores

MERE = Aportaciones de diversos autores al Modelo Entidad Relación básico de Chen.

Figura 4-1. Notaciones aceptadas para Diagramas Entidad Relación - DER



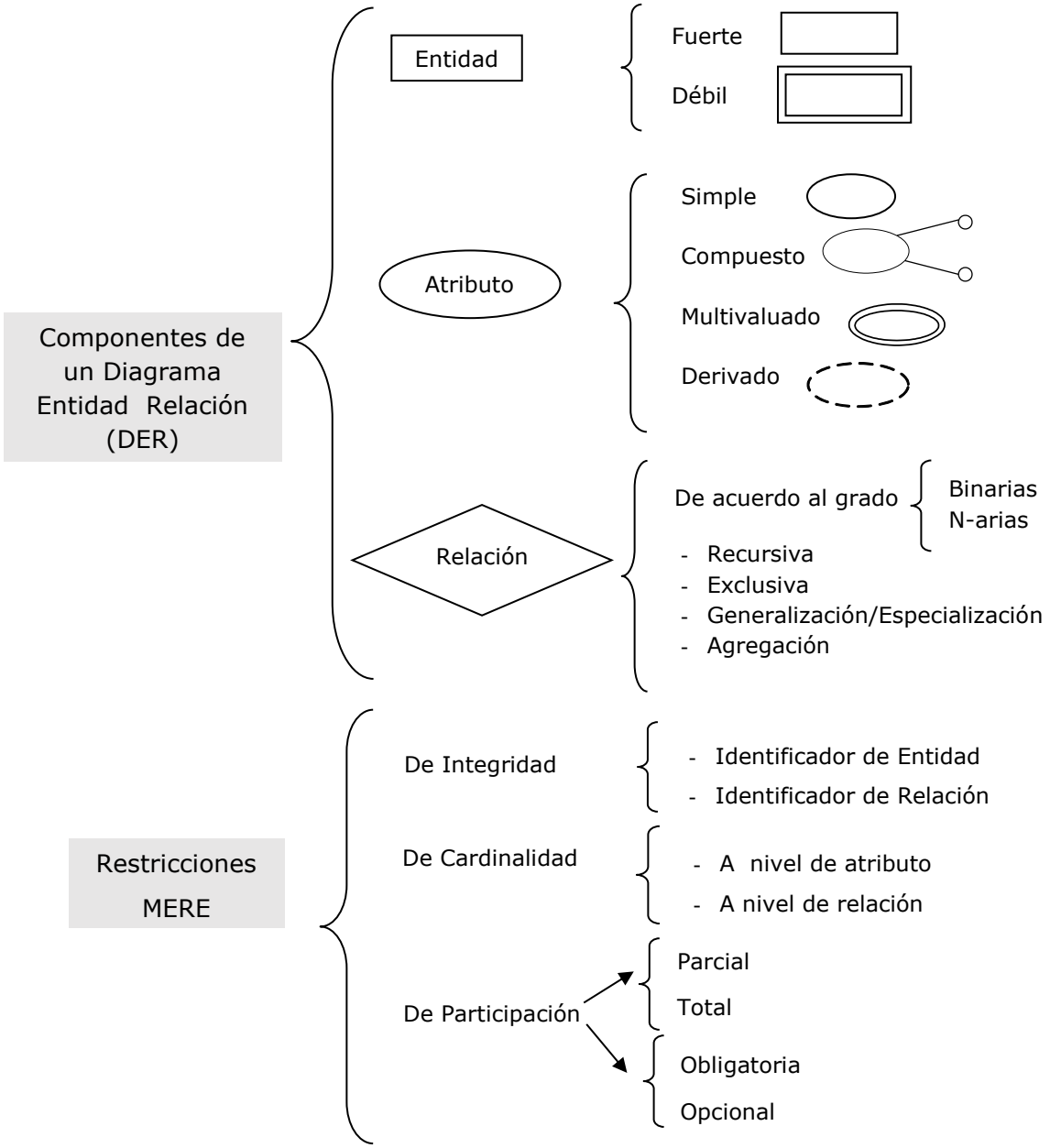
Elaborado por los Autores

4.1.2 Componentes de un Diagrama Entidad Relación

Los conceptos del Modelo Entidad Relación y a su vez los componentes de un Diagrama Entidad Relación (DER) se resumen en la Figura 4-2, y se describen en esta sección del texto.

ISBN: 978-631-6557-89-6

Figura 4-2. Componentes y restricciones que se representan en un DER



Cardinalidades de relación y Participación también se conocen como **Restricciones Estructurales**

Elaborado por los Autores

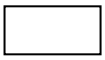
ISBN: 978-631-6557-89-6

- **ENTIDAD**

Definición: Es una cosa u objeto concreto o abstracto que existe, que puede distinguirse de otros acerca del cual se recolectan datos. Un objeto concreto puede ser por ejemplo una persona, un empleado, un lugar, un documento. Un objeto abstracto (acción) puede ser por ejemplo vacaciones, un viaje, préstamo, ventas, etc.

Un Estudiante es una entidad, identificable por su Cédula de identidad, Nombre, Género y Dirección.

Conjunto de entidades*:* Es un grupo de entidades del mismo tipo. Según el enfoque orientado a objetos, sería Clase. También se llaman Tipos Entidad.

Simbología*:*  A una entidad en un DERse la representa con un rectángulo.

Ejemplo:

En un DER se trabaja con conjuntos de entidades, y no con entidades individuales. Es decir, *Estudiante* representado en el ejemplo gráfico expresa un conjunto de estudiantes.

¿Cómo nombrar una entidad?*:* Se recomienda, sujeto singular que describa la función o el rol de la entidad en la empresa. El nombre de una entidad NO puede repetirse en el DER

No confundir los términos utilizados, es común que cuando se utilice el término *entidad* básicamente se refiere al *conjunto de entidades* (también llamado Tipo de entidad), por ejemplo la entidad *Estudiante* se refiere al conjunto Estudiante. En el texto de este libro también se utiliza el término *ocurrencia* se refiere a una entidad del conjunto (también se conoce como instancia de entidad).

Conjunto de entidades (Tipo de Entidad) ⟹ Entidad

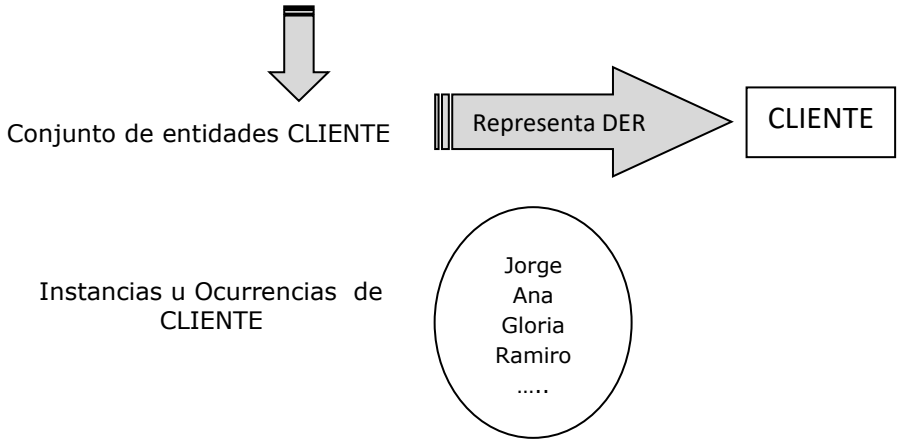
Entidad ⟹ Ocurrencia de entidad (Instancia de entidad)

ISBN: 978-631-6557-89-6

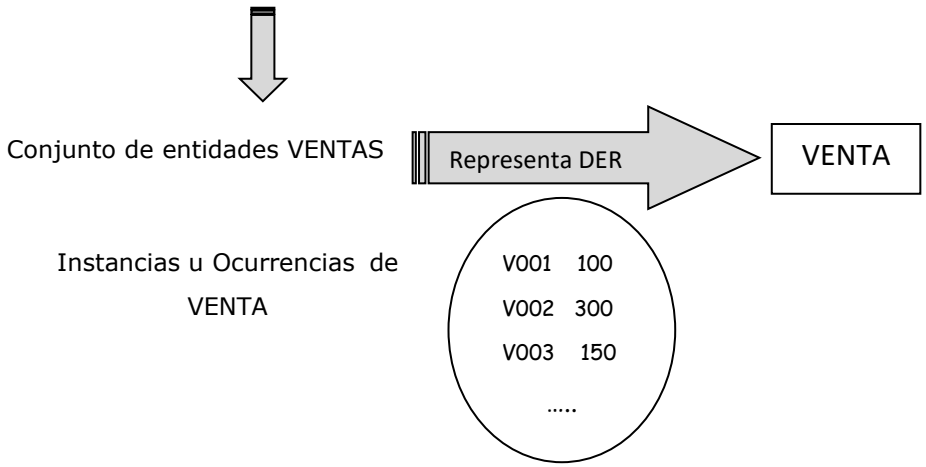
Conjunto de entidades (Tipo de entidad): Colección de entidades del mismo tipo que comparten propiedades.

Por ejemplo, se necesita representar los conceptos del mundo real: Cliente y Ventas

a) Conjunto de personas que son clientes de un supermercado



b) Conjunto de todas las ventas realizadas por el supermercado.



- **ATRIBUTO**

Definición: Es una característica de una entidad. Pueden haber muchos atributos para cada entidad.

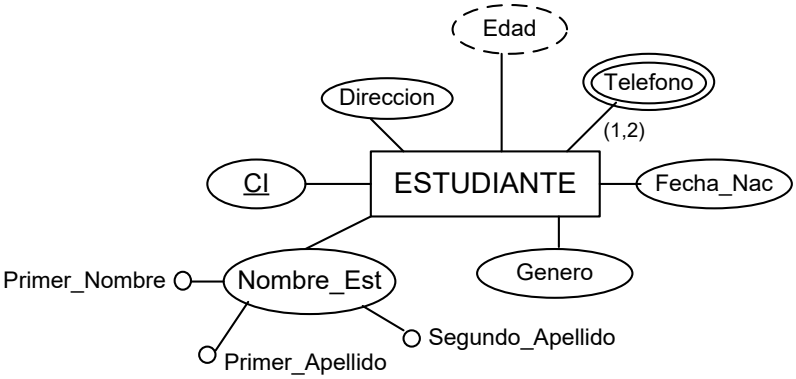
Simbología:  A un atributo en un DER se lo representa con un ovalo (elipse).

ISBN: 978-631-6557-89-6

Ejemplo:

¿Cómo nombrar un atributo?: Se recomienda, sujeto que describa el valor del dato a contener. Se recomienda además, que si los nombres son extensos sean abreviados y claros de identificar. Quizás el mayor incoveniente que presenta esta simbología es que cuando se tiene muchos atributos resulta molestoso su diagramación.

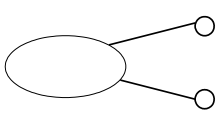
Figura 4-3. Entidad y sus atributos en un DER



Elaborado por los Autores

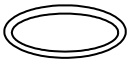
Tipos de Atributos:

Atributo Simple.- Es un atributo que tiene un solo valor con significado propio, que no se puede dividir en partes más pequeñas. Del ejemplo de la entidad Estudiante (ver Figura 4-3) podemos identificar como atributos simples a CI y Direccion.



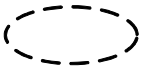
Atributos Compuesto.- Es un atributo que se compone de otros atributos. En la Figura 4-3, el atributo Nombre_Est (Nombre del estudiante) se compone de Primer_Nombre, Primer_Apellido, y Segundo_Apellido.

ISBN: 978-631-6557-89-6



Atributo Multivaluado o Polivalente.- Es aquel que tiene varios valores para cada ocurrencia de la entidad o relación a la que pertenece. Pueden tener un número máximo y un número mínimo de valores, que se los representan mediante *cardinalidades* a nivel de atributo. Se los representa mediante doble ovalo.

En la Figura 4-3 se aprecia que Teléfono es un atributo multivaluado ya que el estudiante puede tener hasta dos teléfonos de referencia, el convencional o fijo y el móvil, mínimo uno.



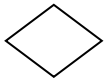
Atributo Derivado.- Un ovalo de líneas interrumpidas indica un atributo derivado, es decir aquel que no está explícitamente almacenado porque puede ser calculado a partir de los valores de otros atributos.

En la Figura 4-3 se aprecia que Edad puede ser un atributo derivado a partir del cálculo de la Fecha_Nac (fecha de nacimiento) y la fecha actual del sistema.

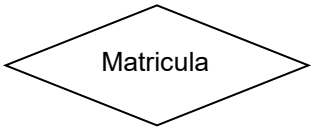
- **RELACION**

Definición: Es la asociación entre dos o más entidades, también se la conoce como Interrelación.

Conjunto de relaciones: Es un grupo de relaciones del mismo tipo

Simbología:  A una relación en un DER se la representa con un rombo.

Ejemplo:



En un DER se trabaja con conjuntos de relaciones, y no con relaciones individuales. Es decir, Matricula representado en el ejemplo gráfico expresa un conjunto de relaciones.

¿Cómo nombrar una relación?: Se recomienda, representar mediante un verbo. Algunas relaciones también son nombradas tomando los nombres o iniciales de las entidades involucradas en la relación. Otras se las nombran con la operación real de la relación, es decir, en términos de mundo. Ejemplo. La relación Matricula, quiere decir que un Estudiante se Matricula en Materia.

ISBN: 978-631-6557-89-6

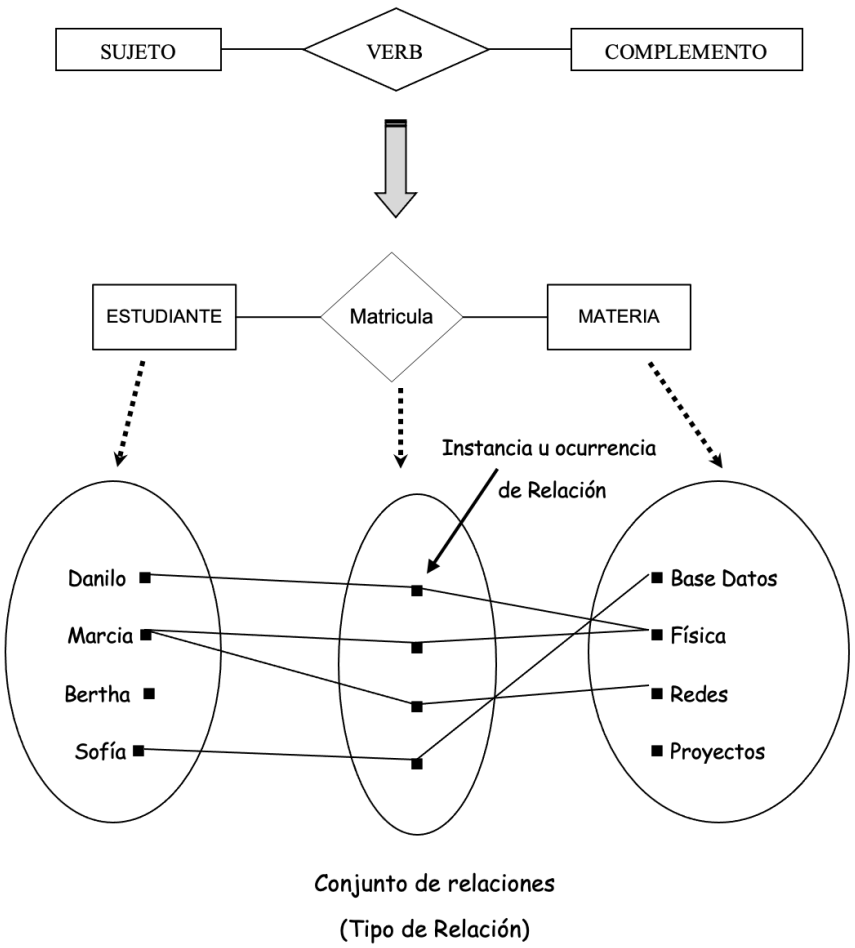
La relación Matricula podría llamarse también Est_Mat, que asocia a las entidades ESTUDIANTE y MATERIA

Conjunto de relaciones (Tipo de Relación) ⟹ Relación

Relación ⟹ Ocurrencia de relación (Instancia de relación)

Cada instancia u ocurrencia de la relación es una asociación entre entidades, en la que participa exactamente una entidad de cada tipo. Representa el hecho de que las entidades están relacionadas entre sí de alguna manera en la situación correspondiente del "mundo real", ver Figura 4-4.

Figura 4-4. Relación entre dos conjuntos de entidades



Elaborado por los Autores

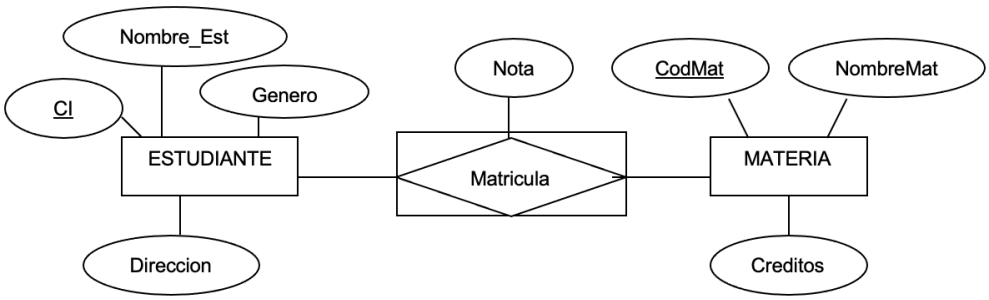
ISBN: 978-631-6557-89-6

Para ayuda en la interpretación de un DER, se debe tomar en cuenta lo siguiente: "*Una entidad actuará como sujeto o complemento, y la relación como verbo, formando todo ello una frase que especifica la gestión que se realiza".* Así tenemos que la estructura a nivel de lenguaje natural sería como se ilustra en la Figura 4-4.

Una relación también puede tener sus propios atributos, sean éstos simples, compuestos o multivaluados. A este tipo de relaciones se presentan con un rombo incluido en un rectángulo. Cierta literatura, cuando una relación tiene atributos propios también la nombra "*Entidad Asociativa".* Como se aprecia en el Ejemplo 4-1 el atributo Nota es propio de la relación, y se lo ha ubicado en el rombo. La nota no es específico solo del ESTUDIANTE ni solo de la MATERIA, sino que la nota es del estudiante en una materia. Para saber la nota, necesito saber primero de cuál estudiante y de qué materia.

Grado de una Relación: Es el número de conjuntos de entidades que intervienen en una relación. En la relación Matricula, del Ejemplo 4-1 intervienen dos entidades, por lo que, diremos que es de grado 2 o binaria. También existen relaciones de grado 3 (ternaria), 4, etc. Pero las más frecuentes son las binarias.

Ejemplo 4-1. Relación con atributos propios



4.1.3 Restricciones de Integridad

Entre las restricciones de integridad, se consideran las de identificador para entidad y para relación.

- **Identificador de Entidad**

Una tarea muy importante dentro de la modelación de base de datos consiste en especificar como se van a distinguir las entidades. Para hacer estas distinciones se asigna

ISBN: 978-631-6557-89-6

un identificador a cada conjunto de entidades. El identificador permite distinguir en forma única a una entidad dentro del conjunto de entidades, por lo que, tiene el mismo concepto que una Clave Primaria en el Modelo Relacional.

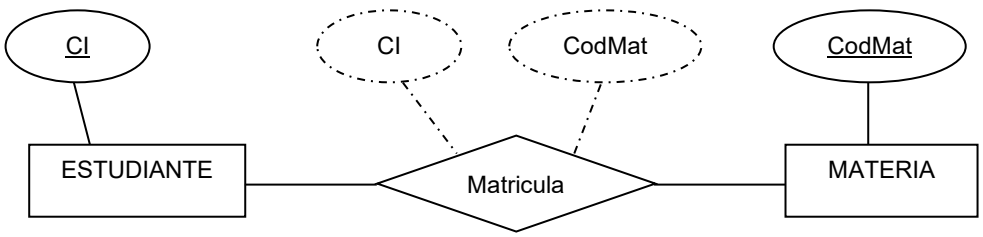
El nombre del *atributo identificador debe ser subrayado*. En la entidad ESTUDIANTE (Ejemplo 4-1) el atributo identificador o clave es CI .

- **Identificador de Relación**

Una relación también tiene un identificador, el cual se corforma por los identificadores de las entidades que asocia. La diferencia con el identificador de la entidad, es que el de la relación NO se especifica en el diagrama, sino que resulta implicita por concepto.

En el Ejemplo 4-2, la clave de ESTUDIANTE es **CI** y la clave de MATERIA es **CodMat,** entonces la clave de Matrícula es **CI + CodMat**

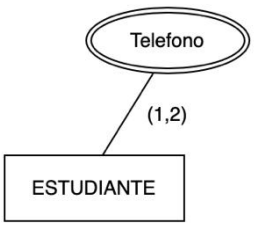
Ejemplo 4-2. Identificador de entidad y de relación



4.1.4 Restricciones de Cardinalidad

Las cardinalidades pueden formularse a nivel de atributo y a nivel de relaciones. Las reglas que definen la cardinalidad de las relaciones son las ***Reglas de negocio***.

4.1.4.1 Cardinalidades a Nivel de Atributo



Cardinalidad a nivel de atributo sólo se formula en atributos multivaluados o polivalentes e indican el número mínimo y máximo de valores que debe tener el atributo asociado a una entidad. De la Figura 4-3, tenemos el atributo multivaluado Telefono con cardinalidad (1,2) lo que significa que el estudiante puede tener mínimo un teléfono de contacto y máximo dos.

ISBN: 978-631-6557-89-6

Si el caso fuese con la cardinalidad (0,2), en cambio se diría que el estudiante NO necesariamente tiene un teléfono para contactarlo, y si lo tuviese, máximo podrían ser dos.

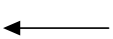
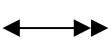
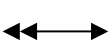
4.1.4.2 Cardinalidades a Nivel de Relación

Las cardinalidades a nivel de relación (también llamadas correspondencias o razón de cardinalidades) expresan el número de entidades con las que puede asociarse otra entidad mediante una relación. Las cardinalidades son más aplicables al describir conjuntos binarios de relaciones, aunque en ocaciones contribuyen a la descripción de conjuntos de relaciones que implican más de dos conjuntos de entidades.

Para un conjunto binario de relaciones R entre los conjuntos de entidades A y B, la cardinalidad debe ser una de las siguientes: Uno a Uno, Uno a Varios, Varios a Uno y Varios a Varios

Existen diferentes notaciones para expresar las cardinalidades, la Tabla 4-2 muestra algunas. La que se utiliza en este texto es la de la primera columna. La notación a utilizarse en este texto se ha seleccionado por ser didacticamente apropiada para el entendimiento de la restricción de cardinalidad; sin embargo, más adelante será complementada con otras notaciones para representar la restricción de particiación.

Tabla 4-2. Notaciones para expresar cardinalidades en un DER

Cardinalidad	Notaciones			
Uno a Uno	**1 : 1**			
Uno a Varios	**1 : N**			
Varios a Uno	**N : 1**			
Varios a Varios	**N : M**			

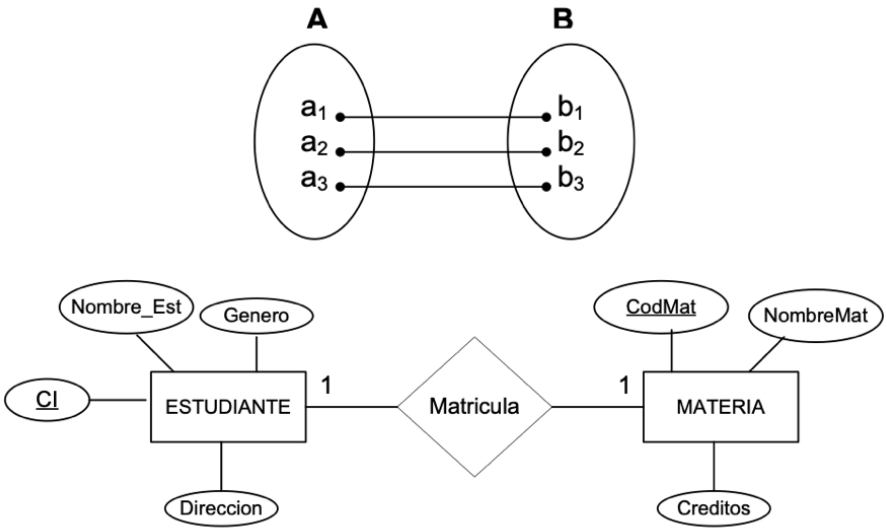
Elaborado por los Autores

- ***Uno a Uno (1:1)***

Una entidad en A esta asociada únicamente con una entidad en B, y una entidad en B esta asociada sólo con una entidad en A.

ISBN: 978-631-6557-89-6

Figura 4-5. Cardinalidad Uno a Uno

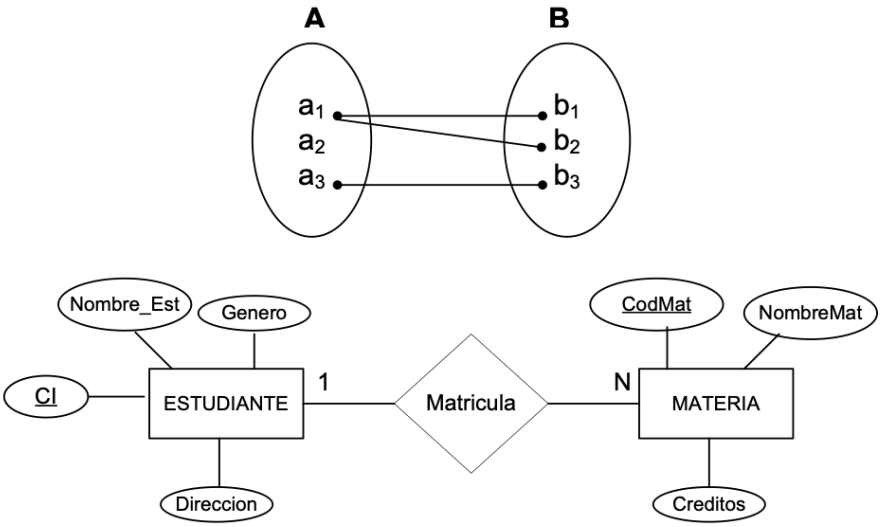


Un estudiante se matricula en una materia, y una materia sólo tiene un estudiante matriculado

- ***Uno a Varios (1:N)***

Una entidad en A se asocia con cualquier número de entidades en B, pero una entidad en B puede asociarse únicamente con una entidad en A.

Figura 4-6. Cardinalidad Uno a Varios



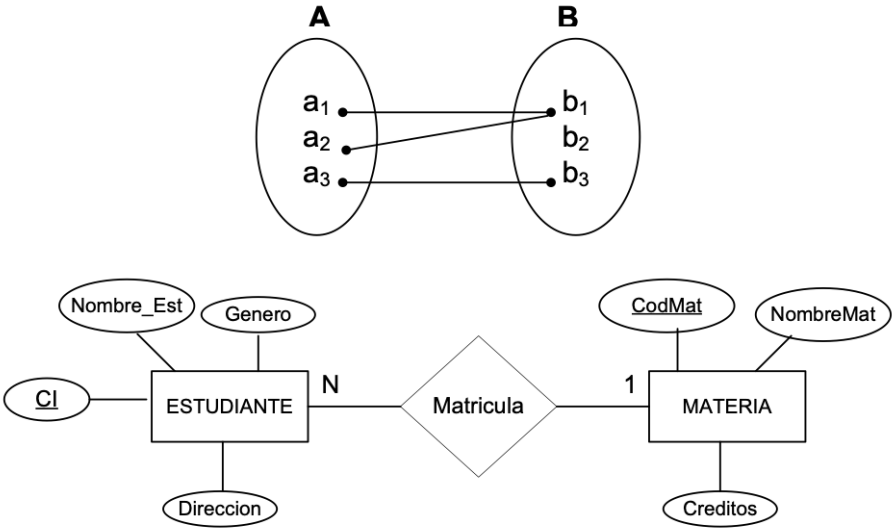
Un estudiante se matricula en varias materias, y una materia sólo tiene un estudiante matriculado

- ***Varios a Uno (N:1)***

ISBN: 978-631-6557-89-6

Una entidad en A esta asociada únicamente con una entidad en B, y una entidad en B se vincula con cualquier número de entidades en A.

Figura 4-7. Cardinalidad Varios a Uno

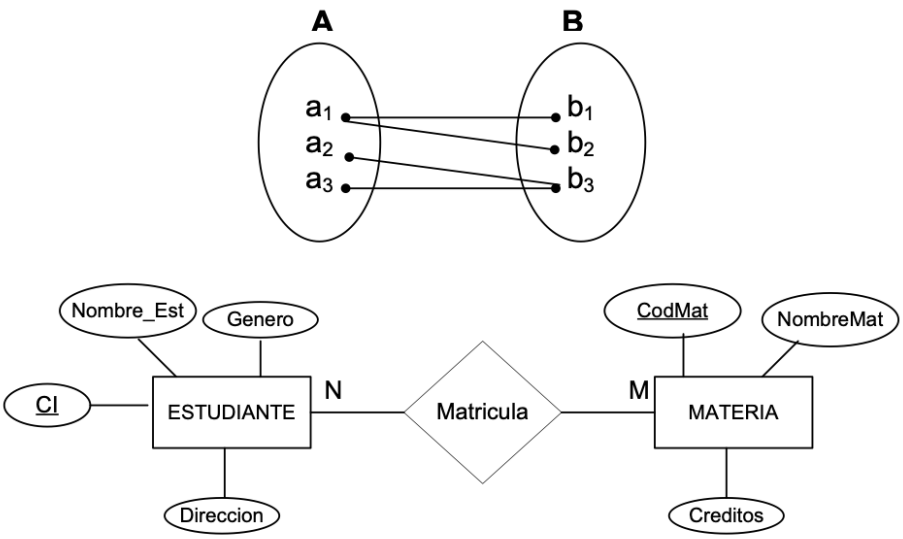


Un estudiante se matricula solamente en una materia, y una materia tiene matriculados a varios estudiantes

- ***Varios a Varios (N:M)***

Una entidadad en A esta sociada con cualquier núumero de entidades en B y una entidad en B esta vinculada con cualquier número de entidades en A.

Figura 4-8. Cardinalidad Varios a Varios

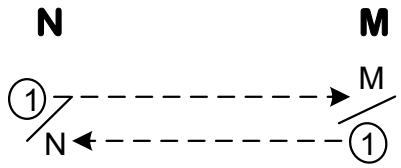


ISBN: 978-631-6557-89-6

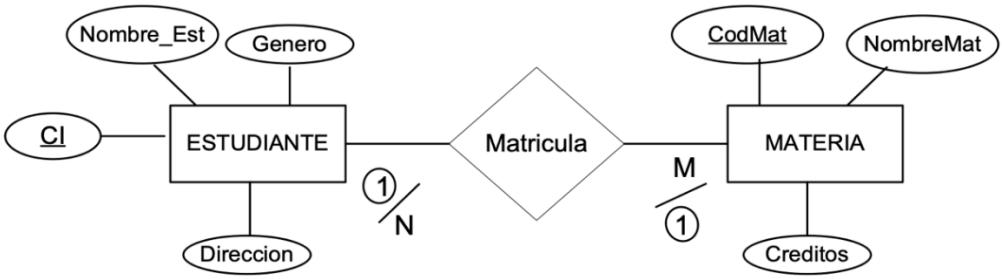
Un estudiante se matricula en varias materias, y una materia tiene matriculados a varios estudiantes

Con los Ejemplos 4-3 y 4-4 se comparte un mecanismo que facilite la interpretación de las cardinalidades, para lo cual tomamos las cardinalidades, N:M y 1:N. Como se aprecia, el círculo marca el inicio y la fecha con línea entrecortada representa su dirección.

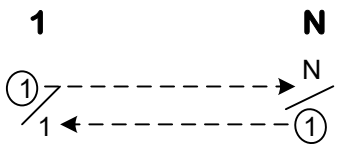
Ejemplo 4-3. Cardinalidad N:M



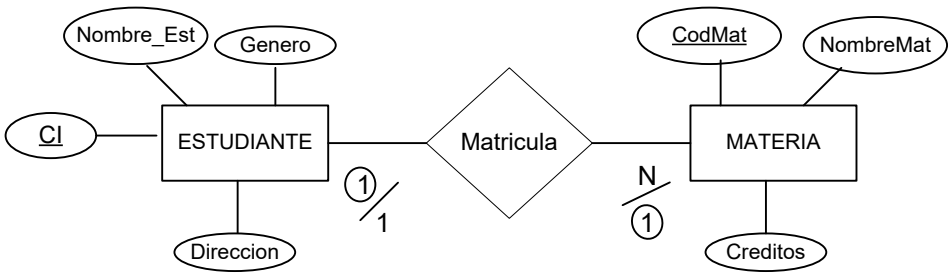
① *estudiante se matricula en* **M** *materias, y* ① *materia tiene matriculados a* **N** *estudiantes*



Ejemplo 4-4. Cardinalidad 1: N



① *estudiante se matricula en* **N** *materias, y* ① *materia sólo tiene* **1** *estudiante matriculado*

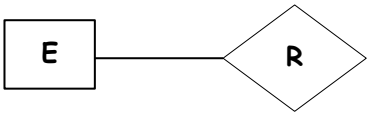


ISBN: 978-631-6557-89-6

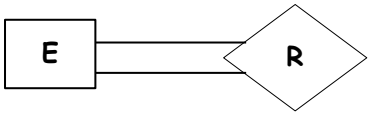
4.1.5 Restricciones de Participación

Las restricciones de participación especifican **si toda** la **extensión** de una **entidad participa en** un tipo de **relación**, **o sólo parte** de la extensión De esta forma la participación puede ser *Total* o *Parcial*

Entidades E1,……,En, PARTICIPAN en el conjunto de relaciones R.



Participación Parcial se representa en el DER con línea simple

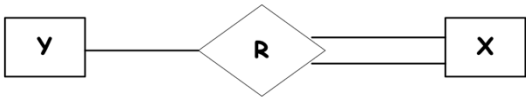


Participación Total se representa en el DER con doble línea.

Cuando se presenta una **participación total** existe una **Dependencia por Existencia**

- **Dependencia por Existencia**

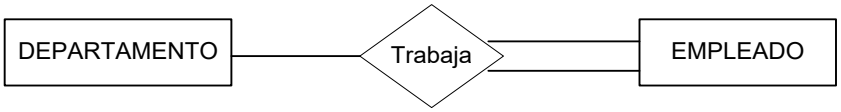
Si la existencia de la entidad **X** depende de la existencia de la entidad **Y**, entonces se dice que **X** es dependiente por existencia de **Y**. Funcionalmente esto quiere decir que si se elimina **Y**, también se eliminará **X**. Se dice también que la entidad **Y** es una entidad Fuerte y que **X** es una entidad subordinada.



En la dependencia por existencia, la participación de la entidad *subordinada es Total* en la relación, mientras que la participación de la entidad *Fuerte es Parcial* en la relación.

Ejemplo 4-5. Dependencia por existencia

Todos los empleados deben trabajar en un departamento



La participación de la entidad EMPLEADO es **TOTAL** en la relación. Por tanto, **todo**, absolutamente todo empleado esta relacionado con DEPARTAMENTO

ISBN: 978-631-6557-89-6

La participación de la entidad DEPARTAMENTO es **PARCIAL** en la relación, es decir, que hay departamentos NO relacionados con EMPLEADO.

4.1.6 Restricciones de cardinalidad y participación respecto a una relación con Mínimos y Máximos

La notación aplicada en las restricciones de cardinalidad y participación, como se pudo observar en capitulos anteriores; no es la única, existen otras formas propuestas por otros autores. En este texto se ha tomado lo mejor de los diferentes autores, de manera que se pueda realizar un diseño más detallado de la base de datos.

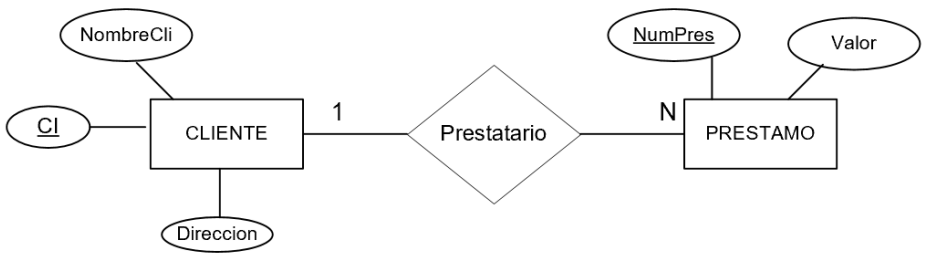
- **Notación A: Sin mínimos y máximos**

Una notación sin mínimos y máximos normalmente se presenta en las propuestas de: **a)** Elmasri & Navathe (2016), y de **b)** Silberschatz, Korth & Sudarshan (2014); donde la cardinalidad y participación se expresan como indica en el ejemplo 4-6.

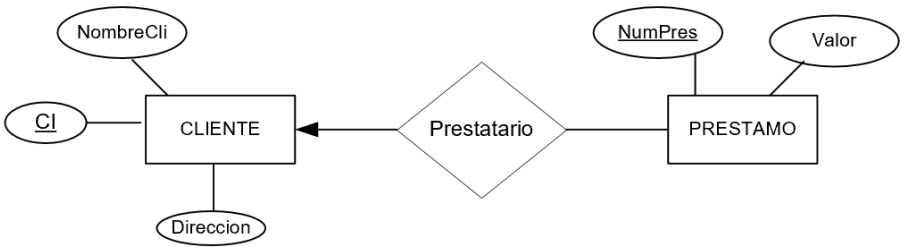
Ejemplo 4-6. Notación de cardinalidad y participación sin mínimos y máximos

Un cliente puede realizar varios prestamos bancarios, cada préstamo es realizado por un solo cliente necesariamente.

a)



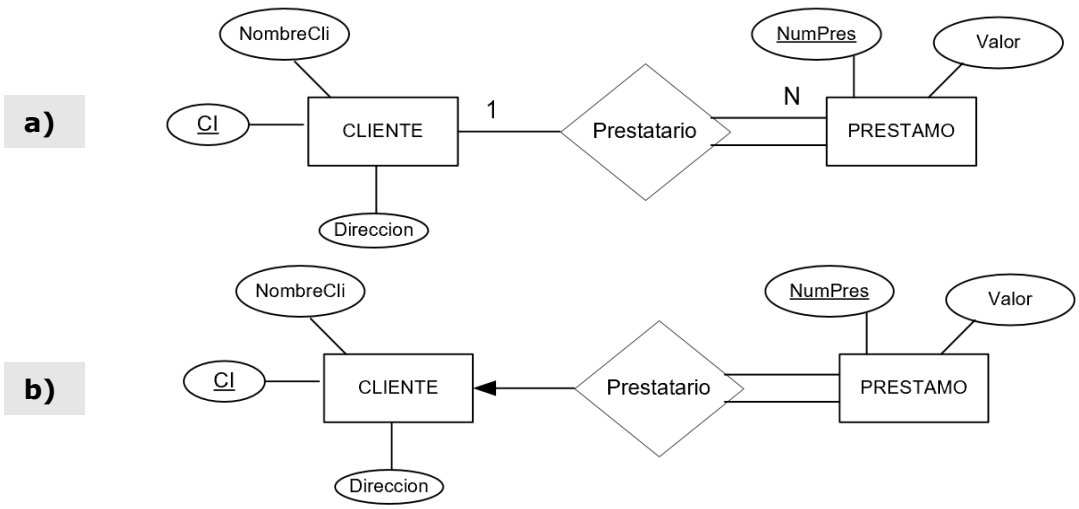
b)



En los DER tanto de Elmasri & Navathe (2016) como de Silberschatz, Korth & Sudarshan (2014), se ha expresado la cardinalidad. Sin embargo, que pasa cuándo un

ISBN: 978-631-6557-89-6

préstamo bacario necesita obligadamente un cliente para que sea otorgado. Una restricción de participación es necesario, de la siguiene manera:



Existirán clientes que no necesariamente realicen prestamos bancarios, por lo que su participación en la relación es *parcial* (es decir habran ocurrencias/clientes que no se encuentren en la relación).

Un préstamo no exisitrá sin un cliente, por lo que su participación en la relación es *total* (todas las ocurrencias/prestamo estarán en la relación)

- **Notación B: Con mínimos y máximos**

Otra forma de representar la cardinalidad es como una pareja de datos (en minuscula) de la forma: (***mínimo, máximo***)

Las parejas de cardinalidades mínima y máxima con la que un tipo de entidad puede intervenir en un tipo de relación son: **(0,1), (1,1), (0, n), (1, n), (n, m).**

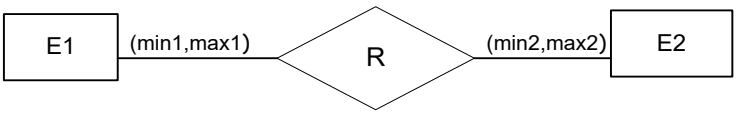
Existen varias notaciones de diferentes autores con mínimos y máximos, entre ellos, también Elmasri & Navathe (2016) propone como notación alterna con un par de números enteros (min, max) para cada participación de un tipo de entidad E en un tipo de relación R.

ISBN: 978-631-6557-89-6

Como refuerzo a los conceptos dados, es importante conocer otras formas de definir a la cardinalidad y a la participación con mínimos y máximos. Para este propósito, en el siguiente apartado se amplía sobre el uso de mínimo y máximos.

Definición B-1:

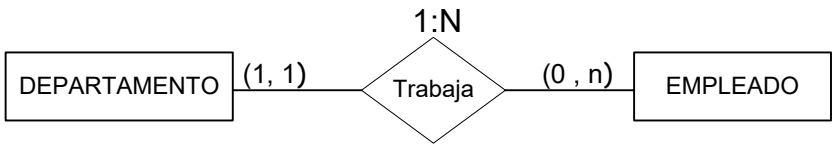
La ***cardinalidad*** con la que una entidad participa en una relación debe especificar el número **mínimo** y el número **máximo** de correspondencias en las que puede tomar parte cada ocurrencia de dicha entidad



En la notación indicada para la definición B-1 (notación B, definición 1), la ***participación*** de una entidad en una relación es *obligatoria* si la existencia de cada una de sus ocurrencias requiere la existencia de, al menos, una ocurrencia de la otra entidad participante. Si No es así, la participación es *opcional*

Ejemplo 4-7.- Notación de cardinalidad y participación con mínimos y máximos

En un departamento pueden trabajar o no empleados. Un empleado necesariamente trabaja en un departamento



Para mejorar la semántica del diagrama en el vértice superior del rombo que representa la relación se ponen las cardinalidades de la relación, que en este caso es 1:N.

- *Un departamento trabaja con (0,n) empleados, quiere cedir que:*

min = 0, la entidad EMPLEADO participa con mínimo **0** en la relación (**participación opcional**)

max = n, la entidad EMPLEADO participa con máximo **n** en la relación

- *Un empleado trabaja con (1,1) departamentos, quiere decir:*

ISBN: 978-631-6557-89-6

min= 1, la entidad DEPARTAMENTO participa con mínimo 1 en la relación (**participación obligatoria**)

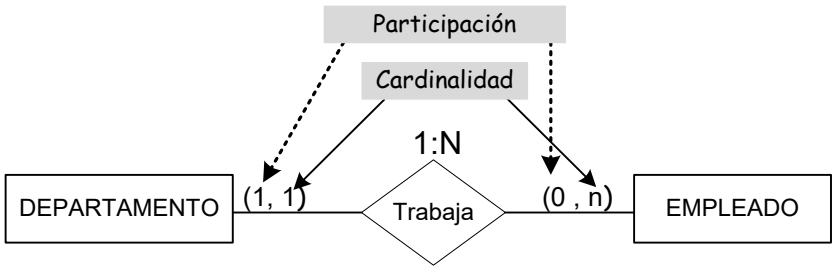
max=1, la entidad DEPARTAMENTO participa con máximo 1 en la relación

Dicho en otras palabras: un departamento está relacionado (trabaja) con 0 como mínimo (opcionalmente) y n (varios) como máximo del tipo empleados. Y, la entidad empleado está relacionado (trabaja) con 1 como mínimo (obligadamente) y 1 como máximo del tipo departamento.

Definición B-2:

La cardinalidad indica el número máximo de relaciones en las que una entidad puede participar

Al igual que en la anterior definición, en el vértice superior del rombo que representa la relación se ponen las cardinalidades máximas (en mayúsculas) con las que intervienen los tipos de entidad.



La ***participación*** se define, como:

Si **min = 1**, toda ocurrencia del tipo de entidad debe participar al menos en una ocurrencia o instancia del tipo de relación; es decir: **participación total** de la entidad en la relación

Si **min = 0**, algunas ocurrencias del tipo de entidad pueden NO participar en el tipo relación; es decir: **participación parcial** de la entidad en la relación.

- **Comparación de las Notaciones A y B**

El propósito es determinar la ventaja de utilizar o NO mínimos o máximos para representar las restricciones de cardinalidad y la participación. De acuerdo con lo

ISBN: 978-631-6557-89-6

expresado en los párrafos anteriores, en la **Notación A**, NO se utiliza minimos y máximos. Y en la **Notación B**, SI utiliza mínimos y máximos, resaltando los siguientes criterios:

- La Notación B, proporciona más información acerca de la relación. La lógica de ésta notación B es la misma a la utilizada en la notación UML.
- Las definiciones B-1 y B-2 dadas en la notación B, son diferentes puntos se vista de expresar las cardinalidades y la participación, mas su resultado es el mismo. La concepción no se pierde, por lo que cualquiera que se utilice es válida; sin embargo, ***la definición B-1 es la se aplica en éste texto***.
- En los modelos conceptuales puede ser suficiente como se indica en la Notación A, pero, *para el análisis detallado se requieren de cuatro (4) puntos para expresar las reglas del negocio que se necesitan hacer cumplir en la estructura de la base de datos*. Por lo tanto, la cardinalidad y participación se expresan con un valor mínimo y un máximo, y se declara gráficamente en el diagrama entidad relación como se indica en la Notación B.
- Hay autores que no utilizan mínimos y máximos, y la cardinalidad es como se indica en la Notación A; mientras que, la participación se restringe a total o parcial. En la Notación B utiliza mínimos y máximos, de los cuales los mínimos permiten determinar la participación obligatoria u opcional. En definitiva, la participación expresada tanto en la notación A como en la B, son equivalentes, tal como se indica en el ejemplo 4-7.

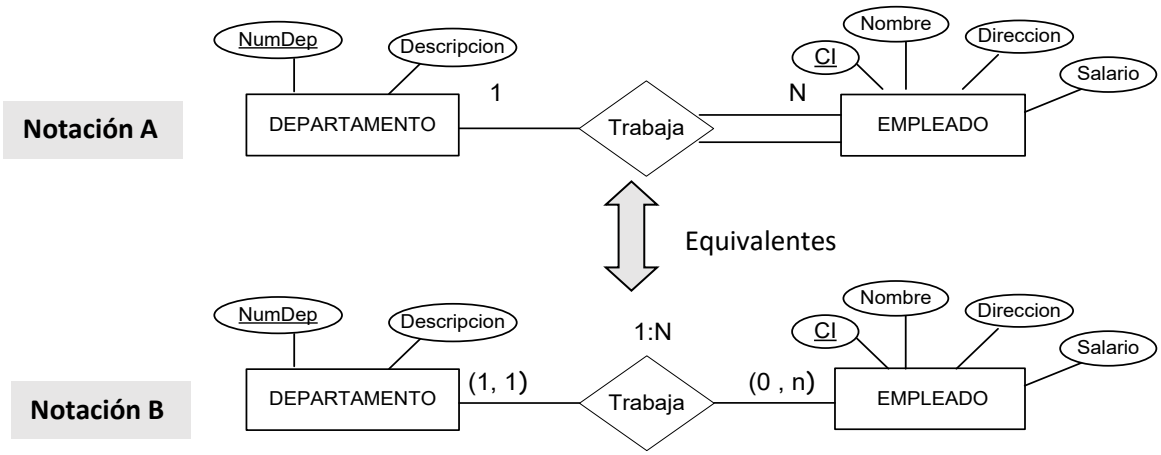
NOTA:
En el texto se asume la Notación B, sin embargo se hace una excepción al representar la entidad débil, como se indica en los siguientes capítulos.

Ejemplo 4-8. Participación total y parcial - Participación opcional y obligatoria

Para mayor explicación, en este ejemplo se contemplan varios casos a ser representados mediante DERs.

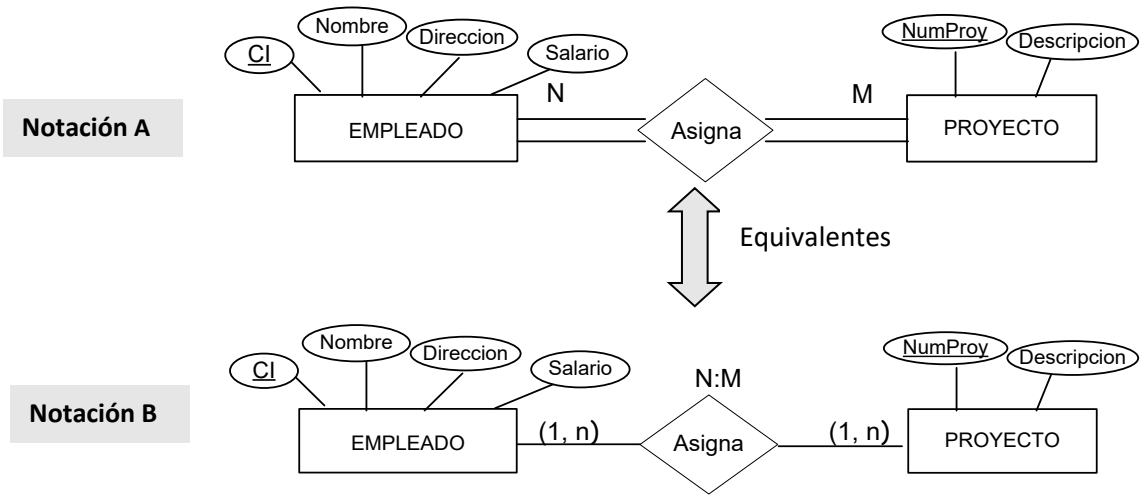
- **CASO 1:** En un DEPARTAMENTO pueden trabajar o no empleados.
  Un EMPLEADO necesariamente trabaja en un departamento.

ISBN: 978-631-6557-89-6



Tanto la notación A como la B del CASO 1, cumplen el requerimiento del ejemplo. La participación de EMPLEADO en la relación es total, mientras que la de DEPARTAMENTO es parcial. Dicho en otras palabras: En un DEPARTAMENTO pueden trabajar ***min=0*** empleados, y, un EMPLEADO con ***min=1***, indica que necesariamente trabaja en un departamento.

- **CASO 2:** Un EMPLEADO necesariamente es asignado al menos a un proyecto, así como también en un PROYECTO debe tener asignado al menos un empleado.



4.1.7 Notaciones Alternativas para representar mínimos y máximos

En el capítulo anterior se hizo un análisis sobre la utilidad de aplicar o no mínimos y máximos, y se concluyó que: "*el uso de mínimos y máximos proporcionan mayor detalle al momento de expresar las reglas de negocio*". Ahora bien, para representar mínimos y

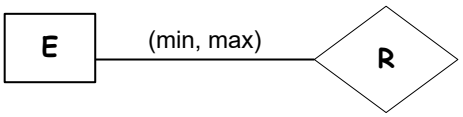
ISBN: 978-631-6557-89-6

máximos también existen varias notaciones. Y, para mejor compresión de este tema se ha tomado las tres notaciones más frecuentes, que son las que indican a continuación:

- **Notación min-max 1:**

Números mínimo y máximo de ocurrencias (instancias) de la Relación en las que puede intervenir una ocurrencia de Entidad.

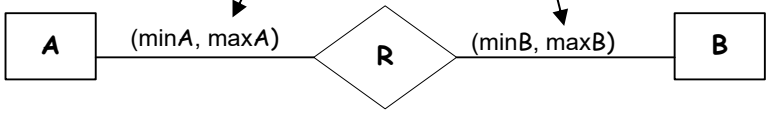
Se representan los mínimos y máximos en la línea que une entidad y relación.



E debe participar al menos *mínimo* veces, y a lo más *máximo* veces en **R**.

Las cardinalidades con mínimos y máximos de *A* y *B* en la relación R, serían:

R (*A* (minA, maxA) : *B* (minB, maxB))



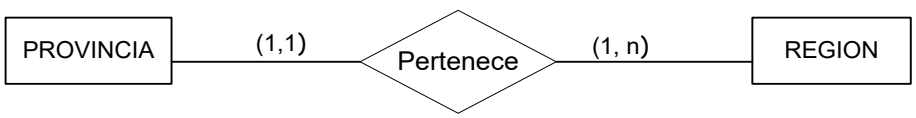
Significa que:

- Cada ocurrencia de A debe participar como mínimo con minA veces en R
- Cada ocurrencia de A puede participar como máximo con maxA veces en R
- Cada ocurrencia de B debe participar como mínimo con minB veces en R
- Cada ocurrencia de B puede participar como máximo con maxB veces en R

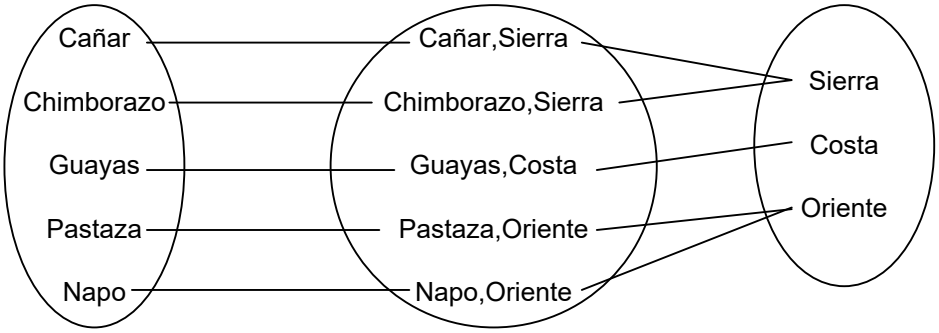
Ejemplo 4-9. Cardinalidad **y** Participación con mínimos y máximos: Notación min-max1

Cada entidad de PROVINCIA participa en la relación Pertenece exactamente una vez, mientras que cada entidad de REGION participa en la relación Pertenece al menos una vez hasta n veces.

Pertenece (PROVINCIA(1,1) : REGION(1,n))



ISBN: 978-631-6557-89-6



Como observamos en el Ejemplo 4-9, en la relación Pertenece una sola vez participa cada ocurrencia de PROVINCIA, mientras que cada ocurrencia de REGION participa desde una hasta n veces. Además, se debe notar que los mínimos tanto de PROVINCIA como de REGION son 1, esto indica que la participación en la relación Pertenece de ambas entidades es ***obligatoria***.

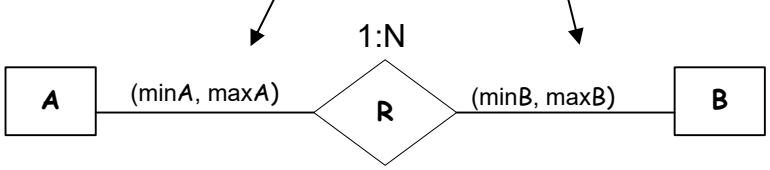
- **Notación min-max 2:**

La notación 2 de mínimos y máximos, es la utilizada en el capítulo anterior (*Restricciones de cardinalidad y participación respecto a una relación con Mínimos y Máximos*).

Números mínimo y máximo de ocurrencias (instancias) de Entidad que pueden estar relacionadas con una instacia de otra entidad. Se representan los mínimos y máximos en la línea que une entidad y relación. En este notación se incorpora la cardinalidad en la parte superior de la relación.

Las cardinalidades con mínimos y máximos de *A* y *B* en la relación *R,* serían:

$$R\ (\ A\ (\min A, \max A) : B\ (\min B, \max B)\ )$$



Significa que:

- Cada ocurrencia de A se debe relacionar como mínimo con minB ocurrencias de B.

ISBN: 978-631-6557-89-6

- Cada ocurrencia de A se puede relacionar como máximo con maxB ocurrencias de B.
- Cada ocurrencia de B se debe relacionar como mínimo con minA ocurrencias de A
- Cada ocurrencia de B se puede relacionar como máximo con maxA ocurrencias de A.

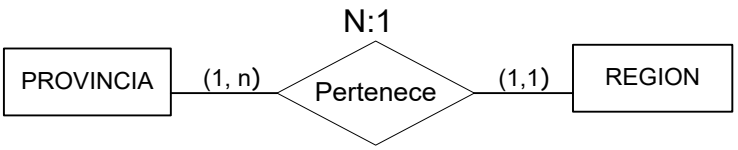
NOTA:

La **Notación min–max 2** es la que se utiliza en éste Texto.

Ejemplo 4-10. Cardinalidad y Participación con mínimos y máximos: Notación min-max 2

Aplicamos la notación min-max 2 sobre el mismo caso del Ejemplo 4-9. Esto es, a cada región pertenecen al menos una provincia, mientras que toda provincia debe pertenecer sólo a una región.

Pertenece (PROVINCIA(1,n): REGION(1,1))



- **Notación min-max 3:**

La notación 3 no utiliza números para mínimos y máximos, sino que aplica una simbología gráfica para representar la participación mínima y máxima de las ocurrencias de entidad; además, permite expresar las cardinalidades de la forma ya indicada.

Notaciones con simbología gráfica como las que se indican es este apartado, son normalmente utilizadas por herramientas de modelado de datos, como las que se indica en los Ejemplos 4-11 y 4-12.

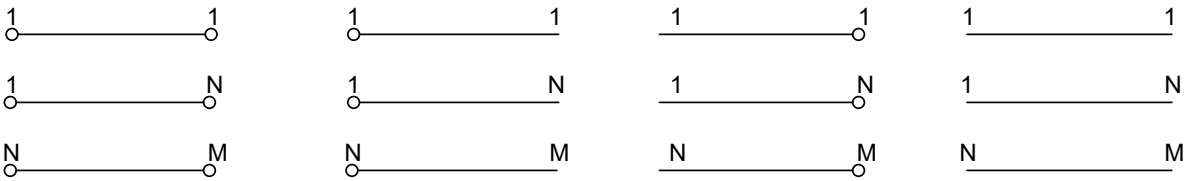
Ejemplo 4-11. Simbología gráfica: Notación min-max 3

——— *Obligatorio*: Indica que a cada ocurrencia de la entidad le corresponde otra ocurrencia de la otra entidad obligatoriamente

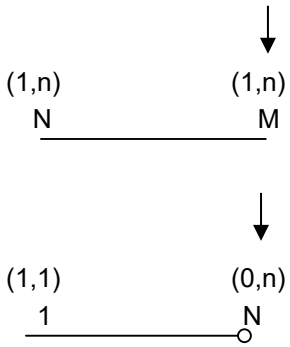
o——— *No Obligatorio (opcional)* : Indica que a una ocurrencia de una entidad le puede corresponder uno o ninguna ocurrencia de la otra entidad.

ISBN: 978-631-6557-89-6

Para Relaciones Binarias (y recordemos que estas son las más generalizadas) se pueden dar un total de doce posibles combinaciones.



Para ilustrar mejor la obligatoriedad/opcional de la participación de ocurrencias y facilitar el aprendizaje, en las siguientes figuras incorporamos rangos e interpretamos las relaciones de izquierda a derecha:

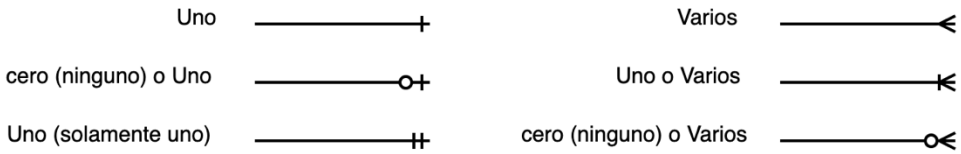


El rango de obligatoriedad ilustra que al menos o mínimo *una* ocurrencia es con la que se relaciona la ocurrencia de la otra entidad

El rango de obligatoriedad ilustra a una ocurrencia (o instancia) de una entidad le puede corresponder *ninguno, uno o más ocurrencias* de la otra entidad.

Ejemplo 4-12. Simbología de pata de gallo: Notación min-max 3

La notación pata de gallo tiene un formato gráfico intuitivo y de fácil interpretación, lo que explica su amplia adopción en la mayoría de herramientas de modelado de datos.



Ejemplo 4-13. Cardinalidad y Participación con mínimos y máximos: Notación min-max 3

Aplicamos Notación min-max 3 sobre el mismo Ejemplo 4-9:

ISBN: 978-631-6557-89-6

Cada entidad de PROVINCIA participa en la relación Pertenece exactamente una vez, mientras que cada entidad de REGION participa en la relación Pertenece al menos una vez hasta n veces. Esto es, que a cada región pertenecen al menos una provincia, mientras que toda provincia debe pertenecer a sólo una región.

Pertenece (PROVINCIA(1,n) : REGION (1,1))

4.1.7.1 Comparación de Notaciones

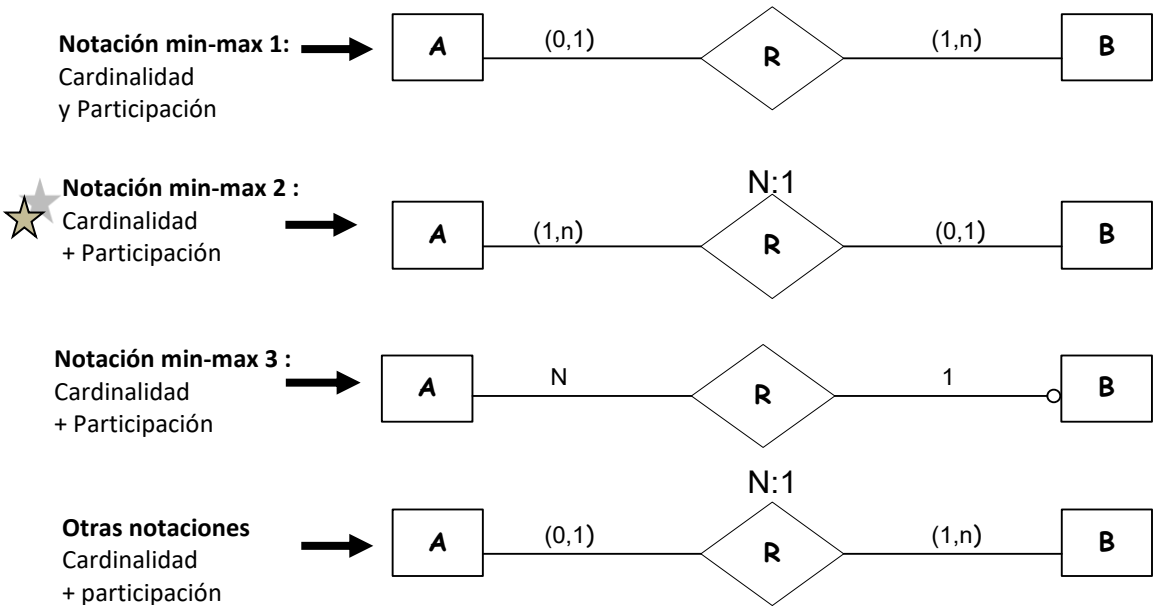
A continuación se lista los aspectos relevantes de las notaciones antes revisadas:

- En Notación min-max 1 se observa que las restricciones de cardinalidad y participación se representan con números mínimos y máximos (min,max). La participación mínima y máxima de las ocurrencias de las entidades se formulan en la relación. Esta notación puede confundir con la forma ya indicada para expresar las cardinalidades, los mínimos y máximos en posiciones inversas.
- Particularmente, con esta notación me resulta más fácil representar mínimos y máximos en relaciones ternarias.
- En Notación min-max 2 también utiliza números mínimos y máximos para las restricciones de cardinalidad y participación, y su posición (min,max) concuerda con las definiciones de cardinalidades anteriormente expuestas. Para separar las restricciones de cardinalidad y participación a esta notación se ha añadido la cardinalidad de la manera indicada (en el vértice superior del rombo). Es la notación utilizada en este texto.
- Con ésta notación se complica más la representación de mínimos y máximos en relaciones ternarias.
- La Notación min-max 3 NO utiliza números para expresar la participación de las ocurrencias de las entidades, sino que utiliza símbolos que se acoplen con la forma de representar las cardinalidades indicada en párrafos anteriores.

ISBN: 978-631-6557-89-6

La Figura 4-9 ilustra las diferentes notaciones para expresar las resticciones de cardinalidad y participación con mínimos y máximos.

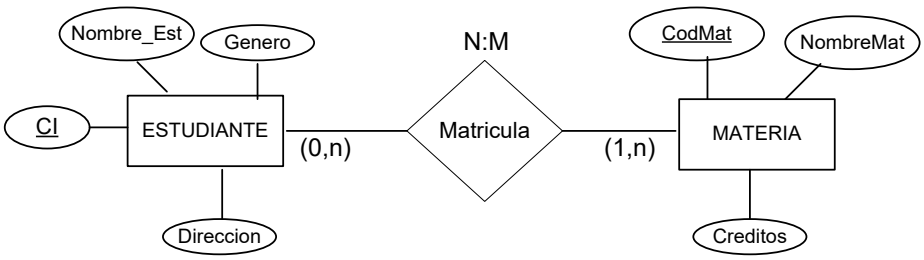
Figura 4-9. Comparación de Notaciones para expresar las restricciones de Cardinalidad y Participación con mínimos y máximos



Elaborado por los Autores

Ejemplo 4-14. Restricciones de Cardinalidad y Participación con mínimos y máximos

En este ejemplo, se puede interpretar: Un estudiante puede matricularse en varias materias obligatoriamente, quiere decir que al menos deberá tomar una materia. Y, una materia puede tener matriculado varios estudiantes NO obligatoriamente, es decir que, puede haber una materia del conjunto que no tenga matriculado estudiantes (ni siquiera uno).



ISBN: 978-631-6557-89-6

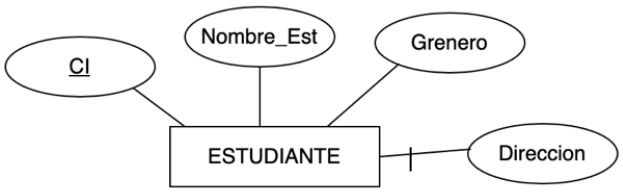
4.1.8 Tipos de entidades

En esta sección se presenta los tipos de entidades, entidad fuerte y la entidad débil.

4.1.8.1 Entidad Fuerte

La entidad fuerte es aquella cuyas ocurrencias son identificadas por sí mismas. Es decir, que los atributos que la identifican son propios de la entidad.

La entidad fuerte se la representa por un rectangulo simple.



4.1.8.2 Entidad Débil

La entidad débil es un tipo especial de entidad que no puede ser identificada de manera única solo con sus propios atributos. Una entidad débil se caracteriza (De Miguel et al., 2000) por los criterios que se indican a continuación:

a) *Dependencia por existencia*: La existencia de las entidades débiles está ligada o subordinada a la de la fuerte. Es decir, existe una dependencia de existencia

b) *Dependencia de identificador*: NO cuenta con un identificador propio suficiente para distinguir de manera única a cada una de sus ocurrencias.

   A pesar de que la entidad débil no cuenta con un identificador propio, pero si posee un grupo de atributos de los cuales se tendrá que elegir alguno(s) que pueda hacer la distinción de las ocurrencias en el conjunto. Por ejemplo el conjunto de entidades DOCUMENTO tiene tres atributos NumDoc, Fecha y Descripcion, de éstos atributos NumDoc es el elegido para actuar como identificador de DOCUMENTO. El problema radica cuando, a pesar de que cada documento (ocurrencia) es disinto, pueden haber varios documentos con el mismo número de documento (NumDoc). De esta forma, la entidad como tal NO cuenta con un identificador completo (valor único).

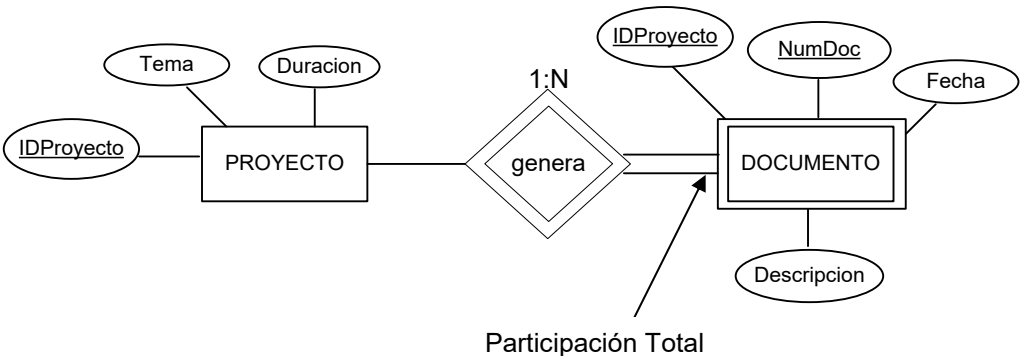
ISBN: 978-631-6557-89-6

Cuando una entidad cumple con los citerios a) y b), es decir es dependiente por existencia de otra entidad fuerte con la que se relaciona, y dependencia de identificador; entonces, se dice que es una *Entidad Débil*.

El identificador completo de una entidad débil se forma por el identificador de la entidad fuerte de la que depende su existencia mas el atributo elegido a ser identificador en la entidad débil. Las entidades débiles se representan mediante un doble rectángulo, y la relación que asocia la entidad fuerte con la débil también lleva líneas dobles (identifica el vínculo).

Ejemplo 4-15. Entidad débil

Por cada PROYECTO se generan documentos, como se visualiza en el ejemplo para ***Pry1*** se asocia el DOCUMENTO ***1*** y ***2*** con las fechas 10/01/2007 y 05/02/2007, respectivamente. Pero, también tenemos un DOCUMENTO ***1*** para ***Pry2*** y para ***Pry3***. NumDoc con el valor 1 se repite a pesar de representar distintos documentos.



Pry1	Seguridad en Redes	8
Pry2	BI en las empresas	7
Pry3	Planificación Estratégica	8

1	10/01/2007	Presentación del proyecto
2	05/02/2007	Resolución de aprobación
1	12/01/2007	Solicitud de prórroga
1	12/01/2007	Presentación del proyecto

Incluso si en DOCUMENTO se hubiese elegido Fecha en lugar de NumDoc se presenta el mismo problema ya que en una misma fecha se pueden generar varios documentos de diferentes proyectos. Por lo tanto, DOCUMENTO no tiene clave única.

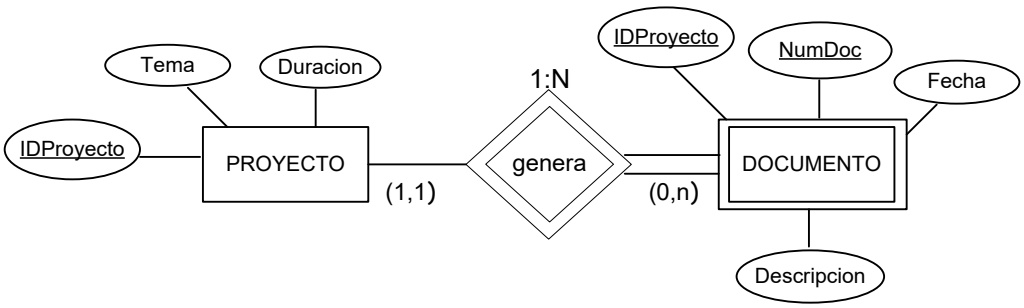
ISBN: 978-631-6557-89-6

Asimismo, DOCUMENTO dependen por existencia de la entidad fuerte PROYECTO, esto quiere decir que si proyecto no existe no se generaría documento alguno, o por lo contrario que si se elimina un proyecto también lo haría sus respectivos documentos.

Consecuentemente, DOCUMENTO es una entidad débil y su clave se conforma por IDProyecto clave de PROYECTO más NumDoc de DOCUMENTO.

La **participación** de la entidad débil en la relación *genera* es ***Total***, por esta razón se representa con doble línea. Todo DOCUMENTO se relaciona con un PROYECTO.

La **participación** de la entidad fuerte en la relación genera es **Parcial**. Un PROYECTO no necesariamente inicia asociándose con un DOCUMENTO.



NOTA:
- La cardinalidad entre una entidad Fuerte y una Débil es de 1 – N
- Se ha representado el identificador de la entidad fuerte (IDProyecto) en la débil, esto sirve para ilustrar el concepto, mas no es obligatorio hacerlo, pues queda sobrentendido como debe conformarse la clave de la entidad débil (IDProyecto + NumDoc). Por esta razón, en el texto puede que no encuentre representado el identificador de la entidad fuerte en la correspondiente débil.
- A pesar de haber indicado que al aplicar mínimos y máximos NO es necesario utilizar la doble línea para especificar la participación total.

RECORDEMOS, con min=1 es equivalente a la doble línea. Sin embargo, es importante conservar esta notación para entidades débiles ya que en el diagrama es más fácil evidenciar la entidad débil de esta manera.

4.1.9 Relación Recursiva

Cuando existen relaciones entre entidades de un mismo conjunto se llama *Relación Recursiva*, en otras bibliografías también la llaman *Reflexiva.* Con este tipo de relación se

ISBN: 978-631-6557-89-6

trata de evitar ambiguedades y **representan un ROL (o Papel)** entre las entidades de un mismo conjunto (Kroenke et al., 2018).

En una relación recursiva es posible que se presenten cardinalidades una a varios y varios a varios.

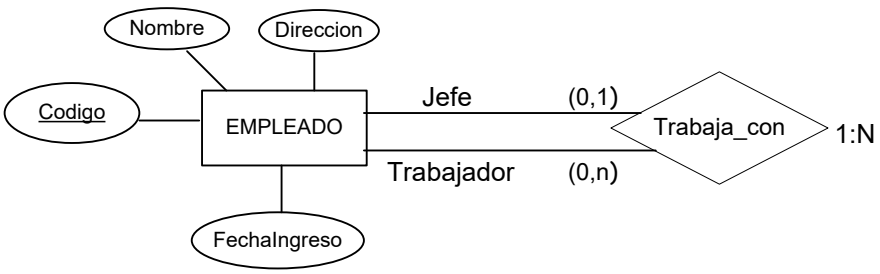
4.1.9.1 Relaciones Recursivas 1 : N

La relación recursiva 1:N se asocia con el caso ***Jefe – Trabajador***, donde un Jefe *trabaja con* uno o varios trabajadores, y por supuesto un trabajador *trabaja con* un jefe.

Las directrices a considerarse en una relación recursiva 1:N, son las siguientes:

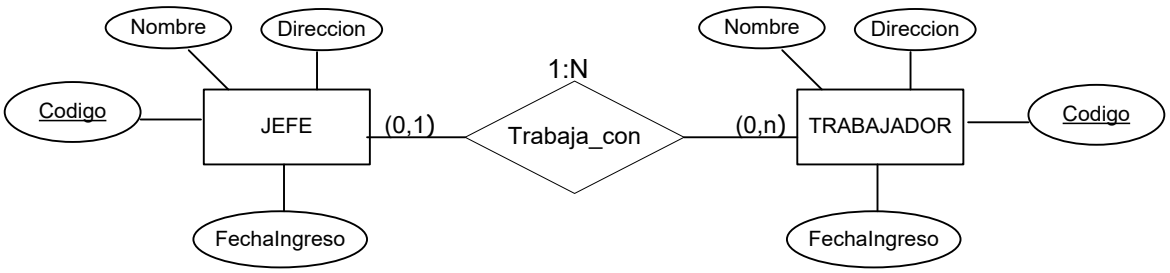
- Tanto el Jefe como el Trabajador son empleados
- También el Jefe es subordinado de otro superior, es decir, el jefe igualmente es trabajador frente a otro jefe superior. Por lo tanto tenemos una combinación ciclica, hasta llegar a un Jefe que no tenga Jefe superior
- Los atributos tanto de jefe como de trabajador son los mismos, ya que ambos son empleados.
- Con la relación recursiva se expresa un rol (jefe y trabajador) entre las ocurrencias del mismo conjunto.

Ejemplo 4-16. Relación Recursiva 1:N



¿Por qué no tener dos entidades una Trabajador y otra Jefe?

ISBN: 978-631-6557-89-6



ERROR: Se pueden tener ocurrencias (entidades) ambiguas, es decir *Jefes* que también estén en el conjunto TRABAJADOR y viceversa.

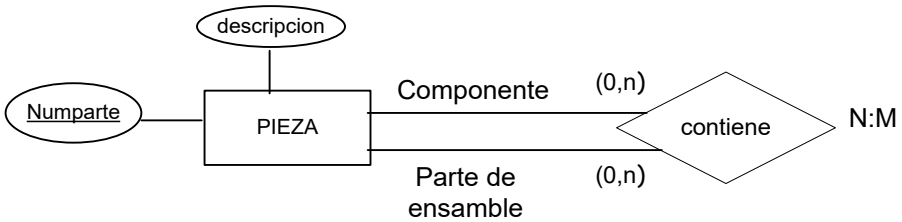
4.1.9.2 Relaciones Recursivas N : M

La relación recursiva con cardinalidad N:M se asocia con el caso ***Componente – parte ensamble***, donde un Componente *se conforma* de una o varias partes de ensamble, y a su vez una parte de ensamble también puede *conformarse* de una o más subpartes elementales, constituyéndose también en un componente.

Las directrices a considerarse en una relación recursiva N:M, son las siguientes:

- Tanto el componente como la parte de ensamble son piezas de un todo.
- Además de contener subpartes, una pieza en particular puede aparecer en muchas superpartes. En última instancia, el ciclo llega a partes indivisibles.
- Con la relación recursiva se expresa un rol o papel (componente y parte de ensamble) entre las ocurrencias del mismo conjunto.

Ejemplo 4-17. Relación Recursiva N:M



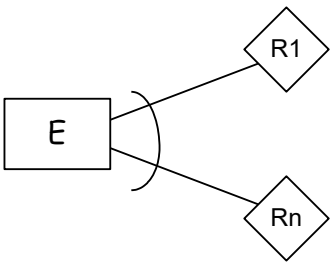
Por ejemplo, un motor está formado de pistones, válvulas, ruedas, entre otros. Cada uno de estos componentes, a su vez contiene partes de ensamble. Así el componente rueda se conforma por las partes de ensamble: un aro, una llanta, y seis tuercas de oreja. Pero

ISBN: 978-631-6557-89-6

también la rueda se constituye una parte de ensamble del componente eje delantero. Asimismo, la parte tuercas de oreja puede conformar cualquier otro componente.

4.1.10 Relaciones Exclusivas

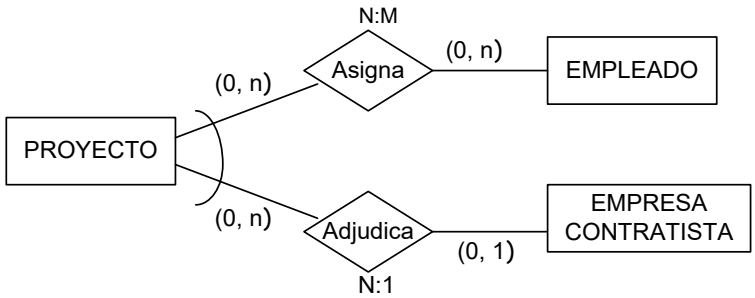
Entre las aportaciones de diversos autores al Modelo Entidad Relación básico está la Relación Exclusiva.



Se dice que una relación es exclusiva si cada ocurrencia de E sólo puede estar presente a lo más en un Ri, i en {1,…,n}.

Ejemplo 4-18. Relación Exclusiva

Considerar el caso de una organización donde se realizan Proyectos, los que pueden ser asignados a Empleados de la organización o a una Empresa contratista, pero no a ambos.



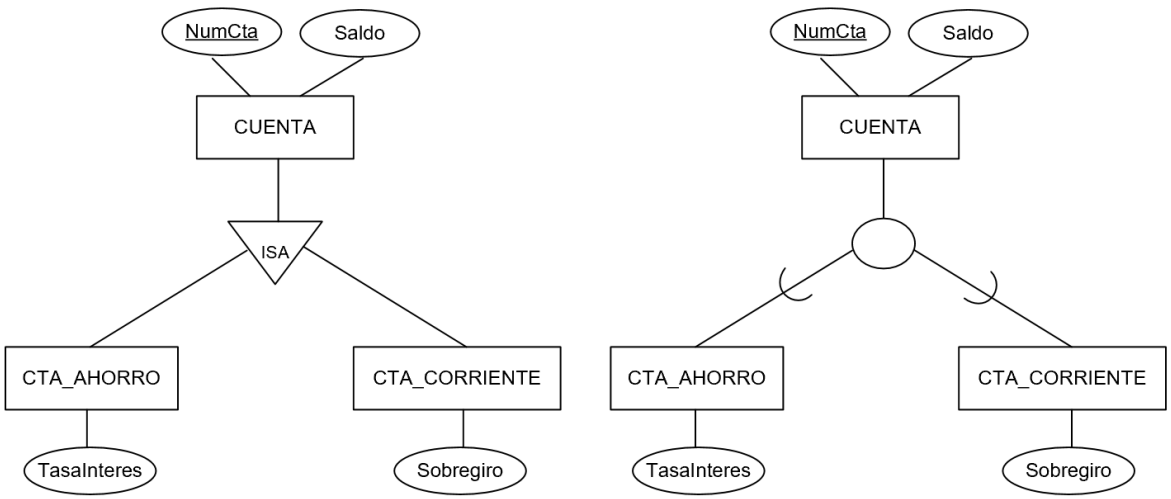
4.1.11 Generalización y Especialización

En el MERE, una entidad agrupa un conjunto de ocurrencias de entidad del mismo tipo. En muchos casos, estas ocurrencias se pueden agrupar a su vez en otros subconjuntos que tienen un significado propio para los propósitos de la Base de Datos y, por tanto, deberían representarse de forma explícita. Para este propósito el MERE presenta las jerarquías de Generalización/Especialización y Categorización; son un caso especial de relación.

ISBN: 978-631-6557-89-6

La Generalización y la Especialización permiten definir jerarquías de entidades que corresponden a la noción de "***es un***" o "***es un tipo de***". Existen varias notaciones para representar estas jerarquías, a modo de ejemplo, la Figura 4-10 indica algunas de las más utilizadas.

Figura 4-10. Notación utilizada para representar la Generalización/Especialización



Notación 1 **Notación 2**

Elaborado por los Autores

Notación 1, utiliza un tiangulo marcado "ISA" que significa "***es un (a)***"

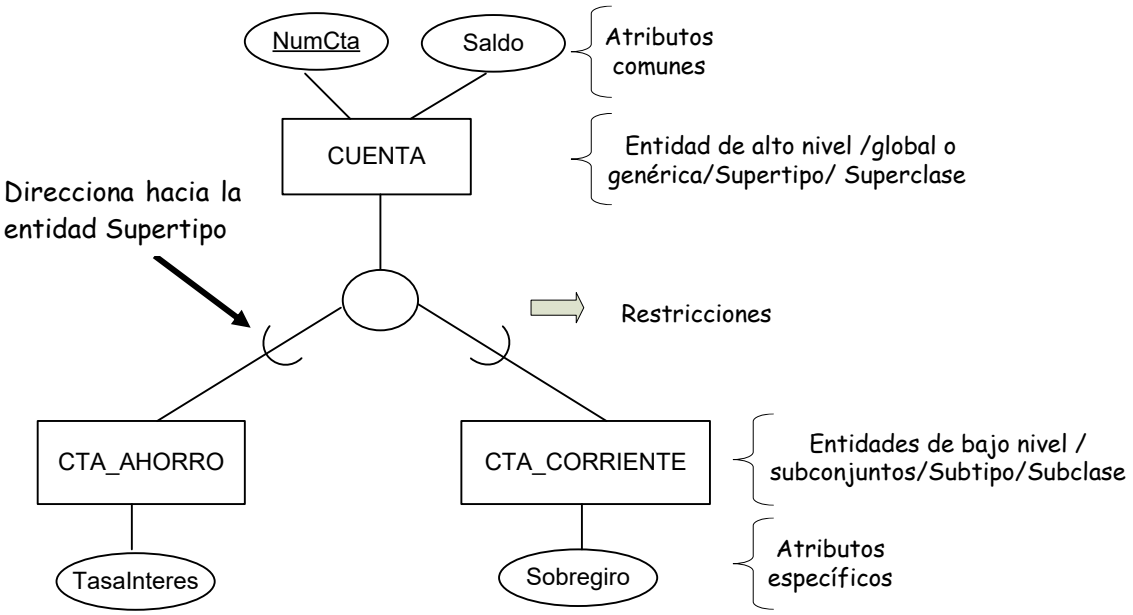
Notación 2, utiliza un círculo y $\in$ para establecer la relación entre los subconjuntos y el conjunto de entidades que forman un tipo de entidad.

NOTA:
En este texto se utiliza la **Notación 2**

Como se observa en la Figura 4-11, tenemos a CUENTA como una entidad de alto nivel también llamada Supertipo (genérica, global o Superclase), con los atributos NumCta y Saldo, siendo estos los atributos comunes, los mismos que se comparten con los subconjuntos llamados Subtipos CTA_AHORRO y CTA_CORRIENTE (entidades de bajo nivel o Subclases). Estos subconjuntos a su vez, tienen los atributos específicos TasaInteres y Sobregiro, respectivamente.

ISBN: 978-631-6557-89-6

Figura 4-11. Relación de Generalización/Especialización



Elaborado por los Autores

- **Generalización**

La Generalización es el resultado de la unión de 2 o más subconjuntos de entidades para producir un conjunto de entidades global (o de más alto nivel, o supertipo).

La generalización consiste en identificar todos aquellos *atributos iguales o comunes* de un conjunto de entidades para formar una entidad global o genérica con dichos atributos semejantes, dicha entidad global quedara a un nivel más alto al de las entidades origen. Es decir, las entidades en la generalización comparten algunas características comunes; iguales atributos y participan en los mismos conjuntos de relaciones.

El diseño en la generalización se realiza en una forma ascendente (bottom-up), en el que los múltiples conjuntos de entidades se sintetizan en un conjunto de entidades de nivel más alto basado en características comunes, ver Figura 4-12.

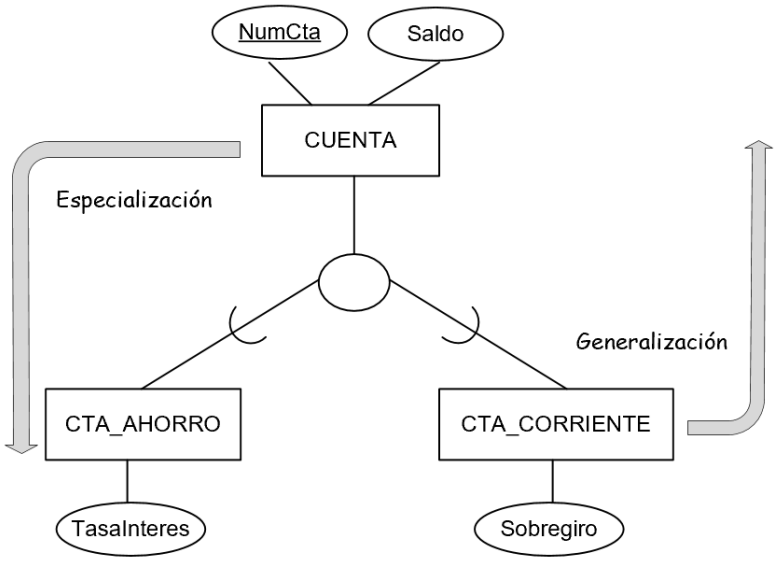
- **Especialización**

La especialización es el resultado de tomar un subconjunto de un conjunto de antidades de alto nivel para formar un conjunto de entidades de más bajo nivel.

ISBN: 978-631-6557-89-6

La especialización parte de un conjunto de entidades simple, *enfatiza las diferencias* entre las entidades dentro del conjunto mediante la creación de distintos subconjuntos de entidades de nivel más bajo (subtipos). Estos subconjuntos de entidades pueden tener atributos o pueden participar en relaciones que no se aplican a todas las entidades del conjunto de entidades de nivel más alto.

Figura 4-12. Mecanismo de abstracción de diseño para la Generalización y Especialización



Elaborado por los Autores

El diseño en la especialización se realiza en una forma descendente (top-down), en el que partiendo de un conjunto de entidades inicial de nivel alto se *clasifica* en entidades de nivel más bajo, ver Figura 4-12. La especialización aplica un mecanismo inverso al de generalización, en lugar de crear una entidad Supertipo a partir de varias Subtipos, descomponemos una entidad Supertipo en varias Subtipos más especializadas.

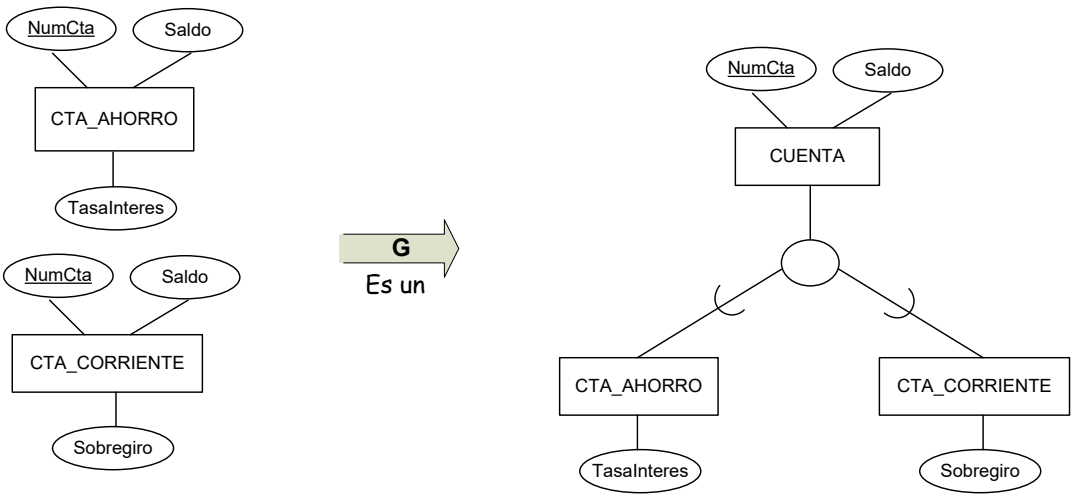
Con base en el ejemplo de la Figura 4-12, a continuación se detalla el mecanismo de abstracción que debe aplicarse en la generalización y la especialización.

4.1.11.1 Mecanismo de abstracción para la Generalización

Para la generalización se ha identificado el subconjunto de entidades CTA_AHORRO con los atributos NumCta, Saldo y TasaInteres, y el subconjunto de entidades CTA_CORRIENTE con los atributos NumCta, Saldo y Sobregiro.

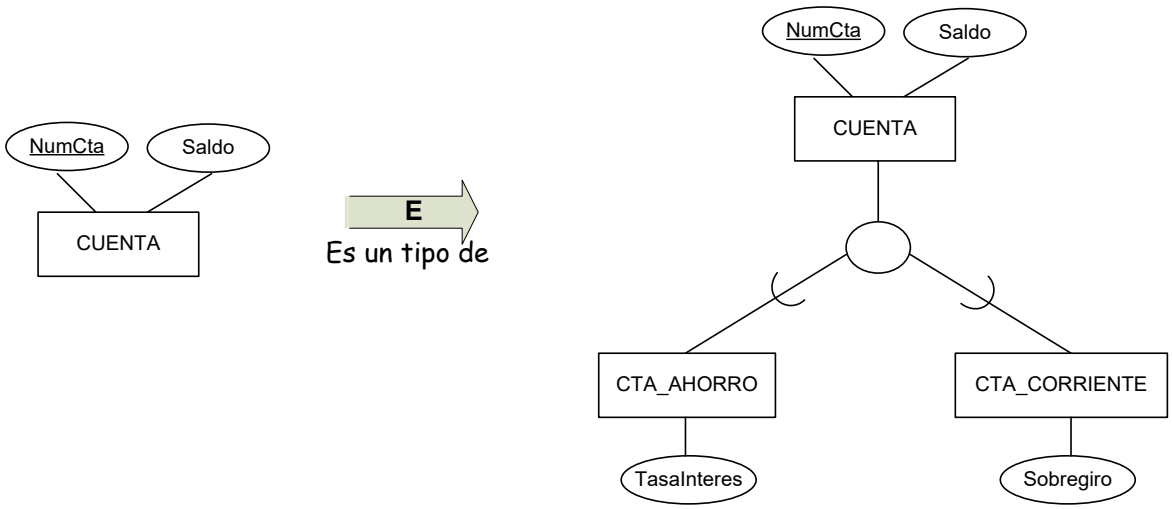
ISBN: 978-631-6557-89-6

Hay similitudes entre CTA_AHORRO y CTA_CORRIENTE, tienen los atributos NumCta, Saldo en común.; por lo tanto se puede aplicar una generalización relacionando estos subconjuntos con un conjunto global CUENTA, el cual asumirá los atributos comunes NumCta, Saldo.



4.1.11.2 Mecanismo de abstracción para la Especialización

Para la especialización se inicia considerando el conjunto de entidades CUENTA con atributos NumCta, Saldo. Pero una cuenta se puede clasificar en CTA_AHORRO y CTA_CORRIENTE y cada uno de estos tipos de cuentas tienen atributos especificos como son: TasaInteres y Sobregiro, respectivamente.



ISBN: 978-631-6557-89-6

*En **resumen**:*

- Tanto la Generalización como la Especialización representan un solo tipo de entidad de alto nivel más conocida como Supertipo que sirve para representar un subconjunto de entidades del mismo tipo de más bajo nivel llamadas Subtipos.
- En el diseño del esquema conceptual se aplicarán en combinación tanto la generalización como la especialización. En términos del DER no se distingue entre especialización y generalización.
- Las jerarquías de entidades serán clasificadas (especialización) o sintetizadas (generalización), el propósito es que, el esquema de diseño llegue a expresar los requisitos de datos.
- La diferencia fundamental entre estas jerarquías es su punto de partida y el objetivo global
- La generalización se utiliza para *representar características comunes*. Se basa en similitudes y sintetiza subconjuntos en un solo conjunto de entidades de nivel más alto, a manera de una entidad Supertipo.
- La especialización se utiliza para *representar características diferentes.*
- La cardinalidad implícita que existe entre el conjunto de entidades de nivel alto (Supertipo) y sus subconjuntos (Subtipos) es de 1 -1

4.1.11.3 Herencia de atributos

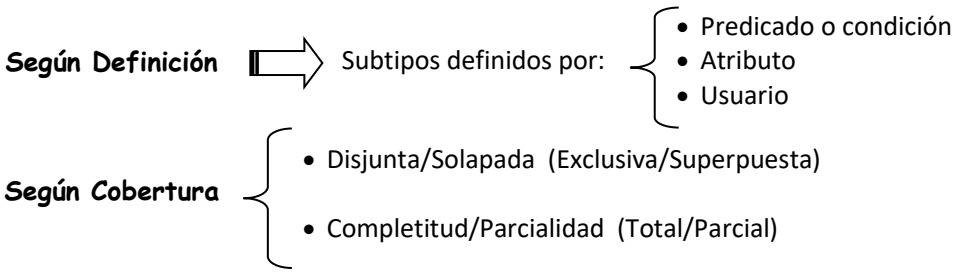
Una propiedad muy importante en las jerarquías de Generalización y Especialización es la herencia de atributos. Los atributos de la entidad Supertipo son heredados por las entidades subtipo. Por ejemplo, CTA_AHORRO y CTA_CORRIENTE heredan los atributos NumCta y Saldo de CUENTA. Es decir, un subtipo hereda todos los atributos del supertipo y toda relación en la que participa el supertipo.

Un subtipo puede tener atributos propios, llamados específicos, y partipar en relaciones por separado (relaciones propias o específicas).

ISBN: 978-631-6557-89-6

4.1.11.4 Restricciones para generalización y especialización

Las restricciones aplicables a la generalización y a la especialización se los han agrupado según la definición y según la cobertura.



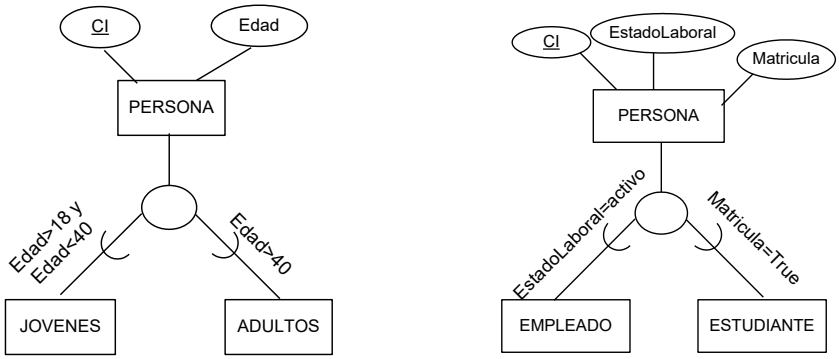
- **Restricciones según definición**

Indica: *¿Qué ocurrencias del Supertipo pertenecen a cada Subtipo*?

Los subtipos son definidos: por Predicado o condición, por atributo o por usuario.

A. Subtipos definidos por predicado o condición: Expresa una condición de pertenencia al subtipo con base en el valor de algún atributo del supertipo. La restricción establece que las ocurrencias del subtipo deben satisfacer la condición definida y que, recíprocamente, todas las ocurrencias del subtipo que cumplan dicha condición deben pertenecer al subtipo.

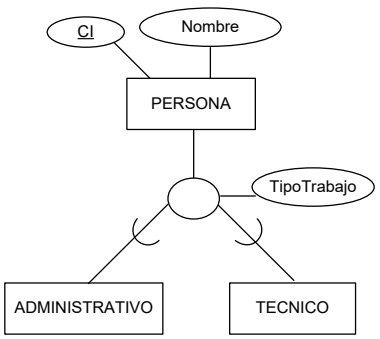
Ejemplo 4-19. Subtipos definidos por predicado o condición



B. Subtipos definidos por atributo: Todos los subtipos se definen a partir de un *mismo atributo*, conocido como discriminante (Tipo); mediante el cual, se determina a qué subtipo pertenece cada ocurrencia.

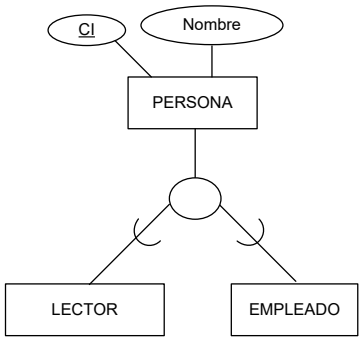
ISBN: 978-631-6557-89-6

Ejemplo 4-20. Subtipos definidos por atributo



C. Subtipos definidos por el usuario: La asignación de una ocurrencia a un subtipo es hecha por el usuario. Dicho en otras palabras, el usuario al insertar una ocurrencia elige a qué subtipo pertenece.

Ejemplo 4-21. Subtipos definidos por el usuario



- **Restricciones según la Cobertura**

Una jerarquía también cumple con la propiedad de cobertura y esta puede ser total o parcial (Completitud/Parcialidad), y exclusiva o superpuesta (Disjunta /Solapada).

A. Cobertura Disjunta / Solapada (d/o): La cobertura *disjunta* o *solapada* permite especificar una restricción entre los subconjuntos o subtipos.

- La cobertura *disjunta* también llamada *exclusiva* requiere que una ocurrencia no pertenezca a más de un subtipo.

ISBN: 978-631-6557-89-6

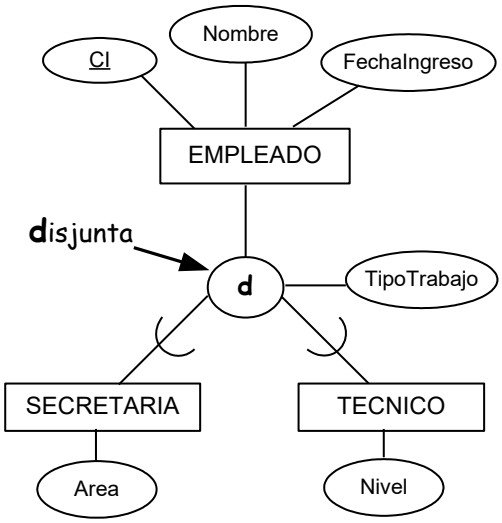
- La cobertura *sOlapada* o *Superpuesta* se presenta cuando una misma ocurrencia puede pertenecer a más de un subtipo.

En un DER se utiliza la letra **d** para indicar una cobertura disjunta, también se utiliza la **x** (exclusiva). La letra **o** (inglés: overlapped) se utiliza para la cobertura solapada.

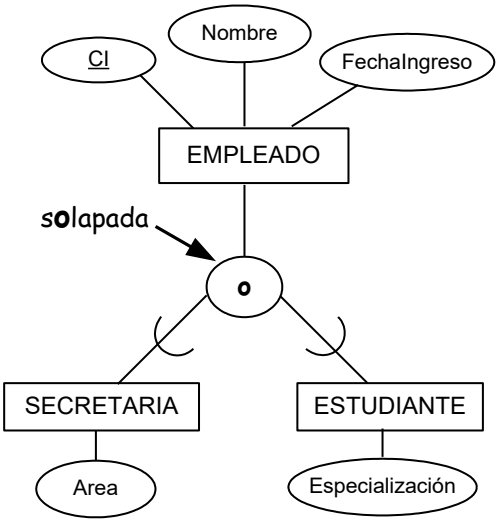
Ejemplo 4-22. Cobertura Disjunta / Solapada

Disjunta: El Empleado es Secretaria o Técnico, pero no ambas cosas a la vez

Solapada: El Empleado pueden ser Secretaria y Estudiante a la vez



(a) Disjunta



(b) Solapada

B. Cobertura Total / Parcial: La cobertura Parcial o Total permite especificar una restricción entre el tipo de entidad global (supertipo) y sus entidades subtipos.

- Es *Total* si todas las ocurrencias del supertipo pertenecen al menos a uno de los subtipos.
- Es P*arcial* si algunas ocurrencias del supertipo no pertenecen a algún subtipo.

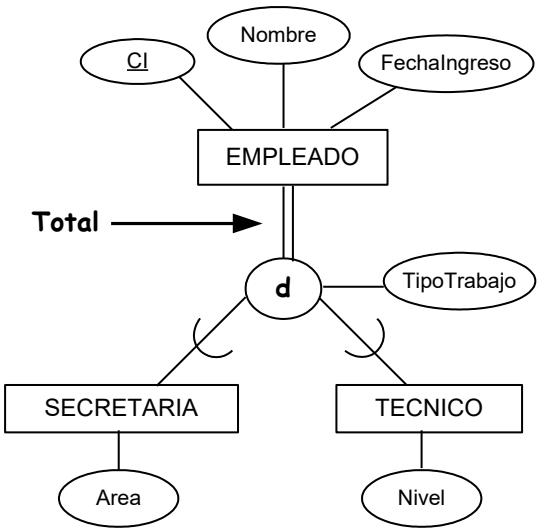
En un DER se utiliza doble línea para representar la cobertura Total, y línea simple para la Parcial.

ISBN: 978-631-6557-89-6

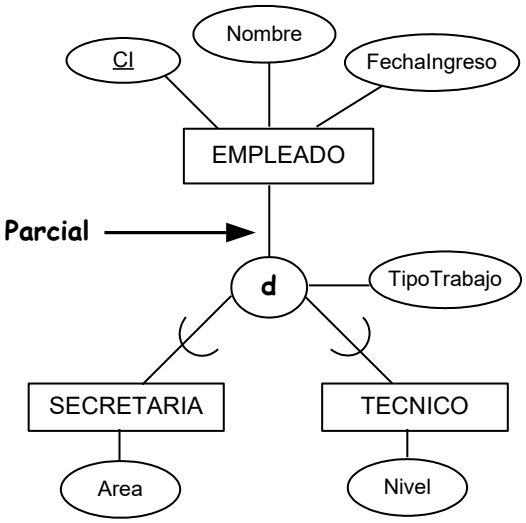
Ejemplo 4-23. Cobertura Total / Parcial

Total: Todos los empleados o son de Secretarias o son Técnicos. Este caso presenta una cobertura Total y Disjunta

Parcial: Algunos empleados o son Secretarias o son Técnicos. Este caso presenta una cobertura Parcial y Disjunta



(a) Total

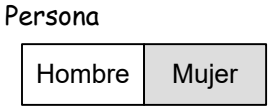


(b) Parcial

Como se pudo apreciar en los ejemplos anteriores, la cobertura puede combinarse y presentar los casos: Cobertura Total y Disjunta, Cobertura Total y Solapada, Cobertura Parcial y Disjunta, Cobertura Parcial y Solapada

- *Caso Cobertura Total y Disjunta:*

  Todas las personas son Hombres o Mujeres, pero no ambos.



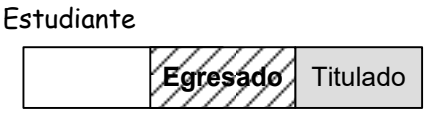
- *Caso Cobertura Total y Solapada:*

ISBN: 978-631-6557-89-6

Todos los empleados son Administrativos o Docentes, pudiendo haber empleados desempeñando ambas funciones

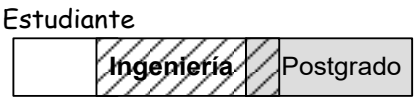
- *Caso Cobertura Parcial y Disjunta:*

  Algunos estudiantes son egresados, mientras que otros están titulados, pero no hay ningún estudiante en ambas situaciones.



- *Caso Cobertura Parcial y Solapada:*

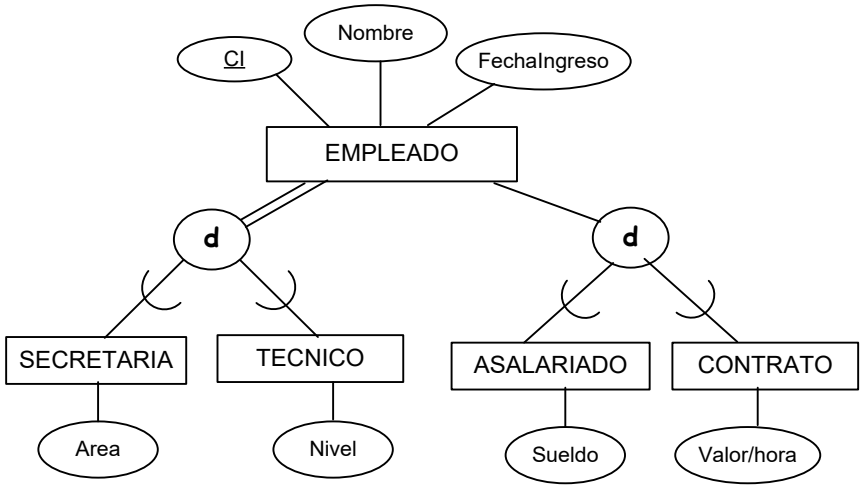
  Algunos estudiantes son de Ingeniería y otros son de postgrado, y hay algunos estudiantes que son de ingeniería y también participan en postgrado.



NOTA:
En realidad, es irrelevante si una entidad en fruto de una generalización o de una especialización, no deja de ser una entidad, y por lo tanto, no afecta al modelo.
En la generalización la entidad supertipo debe ser también una entidad subtipo. La especialización no tiene este limitante.

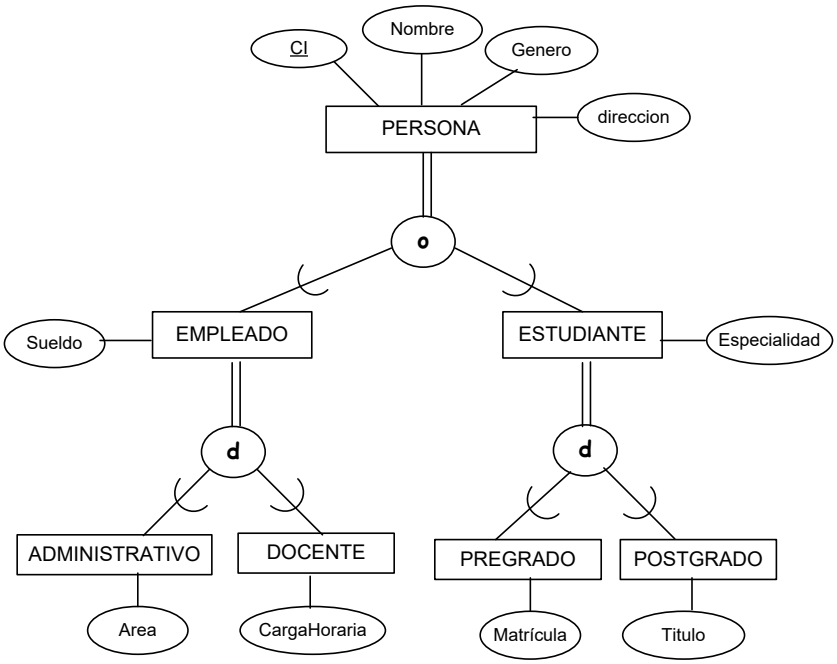
Ejemplo 4-24. Varias especializaciones desde un tipo de entidad

ISBN: 978-631-6557-89-6



Ejemplo 4-25. Especialización de Especialización

Cada subtipo hereda atributos y relaciones de sus supertipos predecesores, hasta la raíz. Es decir, DOCENTE hereda de EMPLEADO y PERSONA (raíz).



4.1.11.5 Reglas de inserción y borrado para Especialización y Generalización.

En las jerarquías de especialización y de generalización, para operaciones de inserción y de borrado, es importante considerar las reglas que se listan a continuación:

ISBN: 978-631-6557-89-6

- Si se elimina una ocurrencia de la Supertipo, significa que hay que eliminarla de los subtipos
- Si la cobertura es TOTAL, cuando se inserta una ocurrencia en la Supertipo hay que insertar esta ocurrencia en al menos una de las subtipos.
- Si la cobertura es Definido por atributo o por condición, cuando se inserta una ocurrencia en la Supertipo hay que insertar esta ocurrencia en las subtipos definidas por atributo o por condición.

4.1.12 Agregación

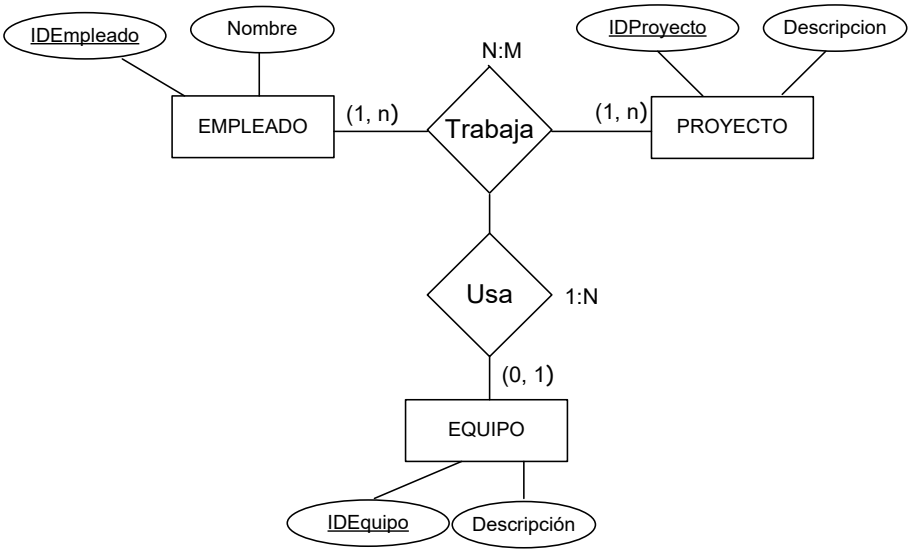
Una limitante del modelo Entidad Relación es que no es posible expresar relaciones entre relaciones. Para mejorar el entendimiento sobre la necesidad de utilizar una agregación se presenta un caso ejemplo, el cual, describe información acerca de los empleados que trabajan en diferentes proyectos, y dependiendo del trabajo que realicen, pueden utilizar un equipo o maquinaria.

Supóngase que cada par Empleado-Proyecto usa un EQUIPO para que se realice el trabajo. Al construir un DER básico, Figura 4-13, se aprecia que las relaciones Trabaja y Usa no pueden asociarse directamente. Una posible solución sería, formar relaciones simples como lo indica la Figura 4-14; pero, se tendría los siguientes problemas:

- La relación Usa de la Figura 4-13 muestra una relación ternaria entre EMPLEADO, PROYECTO y EQUIPO. Cada par empleado-proyecto en Trabaja está también en Usa provocandose redundancia
- Es necesario garantizar que cada ocurrencia de la relación Usa esta también en Trabaja

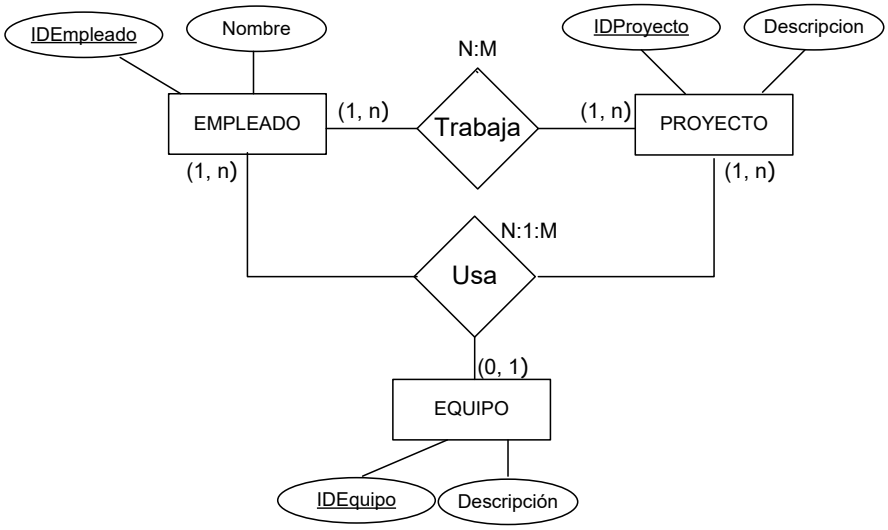
ISBN: 978-631-6557-89-6

Figura 4-13. Limitante del Modelo E-R: Relación entre relaciones



Elaborado por los Autores

Figura 4-14. Relaciones simples con redundancia

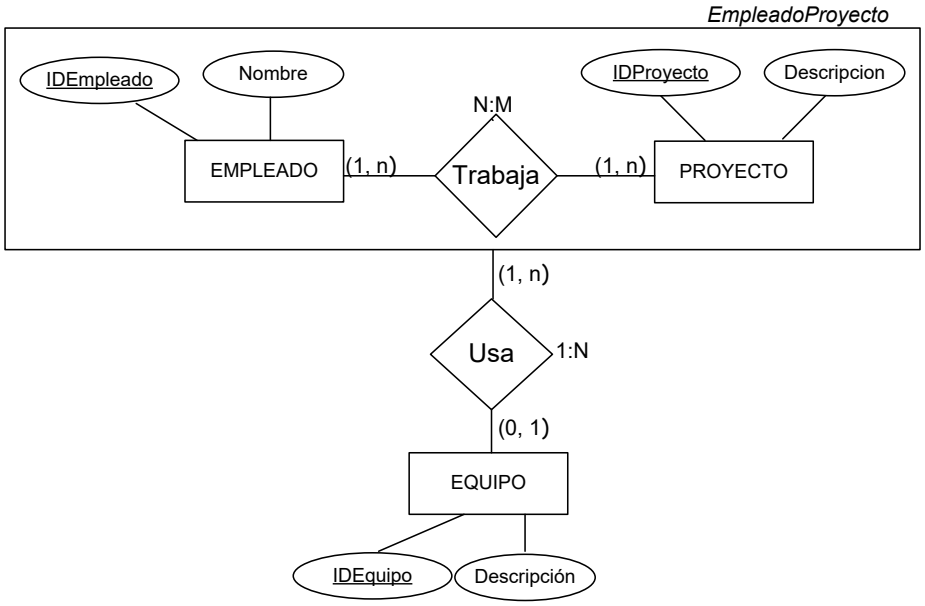


Elaborado por los Autores

La mejor forma de modelar casos como el expuesto, es mediante una *Agregación*. La Agregación es una abstracción a través de la cual las relaciones se tratan como entidades. Por ejemplo, Trabaja puede abstraerse como una entidad y puede llevar el mismo nombre de la relación u otro representativo, por ejemplo: EmpleadoProyecto de la Figura 4-15. La nueva entidad EmpleadoProyecto tiene el mismo trato que cualquier otra entidad.

ISBN: 978-631-6557-89-6

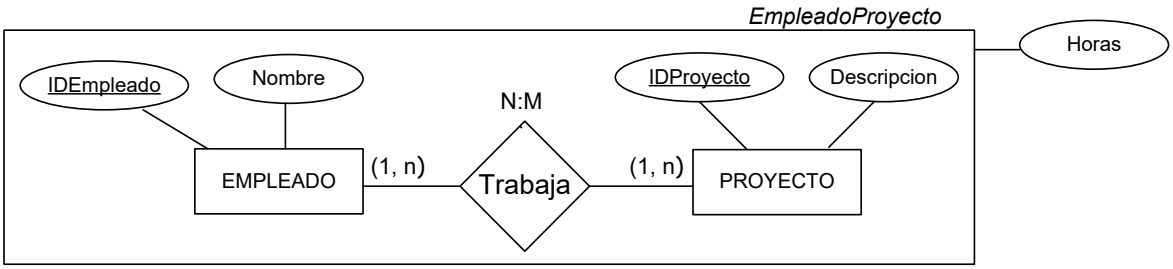
Figura 4-15. DER con Agregación



Elaborado por los Autores

En el caso que Trabaja tuviese atributos, como por ejemplo Horas que indica el número de horas que el empleado le ha dedicado en el proyecto, este atributo se los representa como parte de la entidad abstracta EmpleadoProyecto (ver Figura 4-16).

Figura 4-16. Entidad Abstracta – Agregación con atributos



Elaborado por los Autores

4.2 Ejercicios: Diseño Conceptual

En este texto se han planteado algunas pautas para crear un esquema conceptual mediante un DER. Para el desarrollo de los ejercicios, las pautas en mención se recuerdan a continuación:

ISBN: 978-631-6557-89-6

a) Hablar con el cliente e intentar dejar claros los parámetros y objetivos del problema o mundo real a modelar. Por supuesto, tomar nota de todo.
b) Estudiar el planteamiento del problema para:
   - Identificar los conjuntos de entidades útiles para modelar el problema.
   - Identificar los conjuntos de relaciones y determinar su grado, cardinalidad y tipo.
c) Realizar el mapeo del primer DER
d) Identificar atributos y dominios para los conjuntos de entidades y relaciones.
e) Seleccionar los identificadores (claves) para cada uno de los conjuntos de entidades.
f) Verificar que el esquema resultante cumpla el planteamiento del problema. Si no es así, se vuelve a repasar el procedimiento desde principio (a).
g) Seguir con la siguiente etapa del proceso de diseño de base de datos: Diseño Lógico.

1) El hospital Sultana de los Andes proporciona los servicios de consulta externa y hospitalización, es decir los pacientes pueden utilizar cualquiera de estos servicios. El paciente cuyos datos son: CI, Nombre, dirección y teléfono es atendido por un Médico.

   Al médico se lo identifica por un código, nombre, dirección, y teléfono; cabe mencionar que el médico puede tener una o varias especializaciones.

   El paciente puede ser tratado en consulta externa o ser hospitalizado, en caso de hospitalización se le asigna una cama que corresponde a una área de hospitalización. El paciente de consulta externa tiene una fecha que identifica la primera vez que fue atendido en el hospital. Puede ser que un paciente externo también pase a hospitalización.

Solución:

Identificar los conjuntos de entidades

ISBN: 978-631-6557-89-6

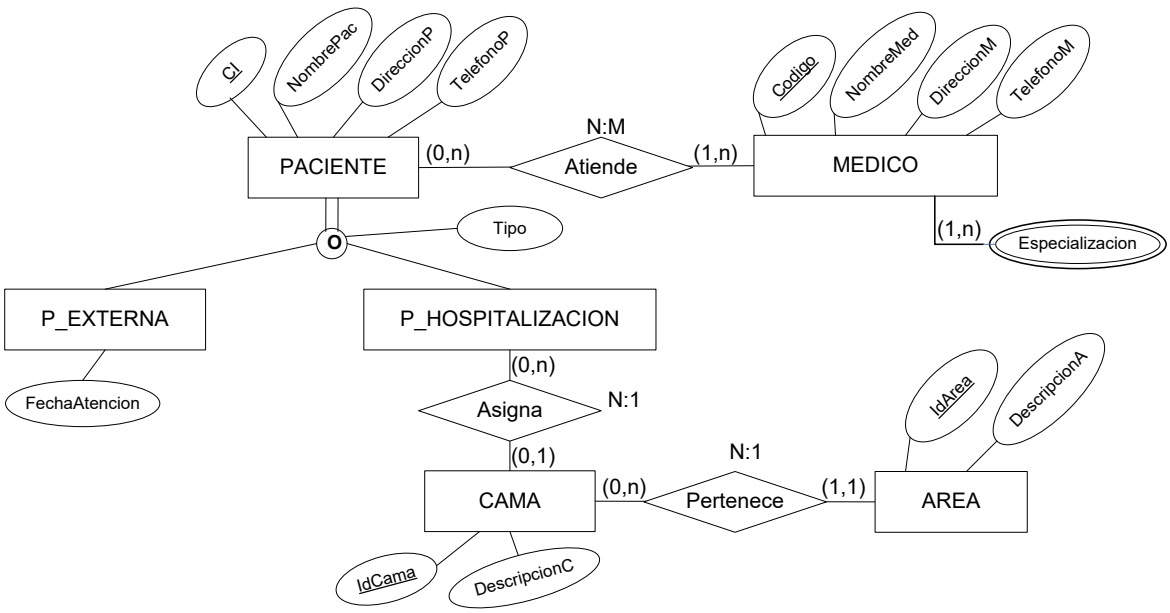
PACIENTE.- representa un rol, y contiene los datos relativos a los pacientes que han utilizado los servicios del Hospital Sultana de los Andes. El paciente puede ser de consulta externa o de hospitalización.

MEDICO.- representa un rol, y contiene los datos relativos a los médicos que trabajan en el hospital

CAMA.- Para cada paciente hospitalizado debemos asignarle una cama. Se ha definido como entidad ya que cada cama está asociada a un área del hospital

AREA.- El hospital tiene áreas y debemos representarlas

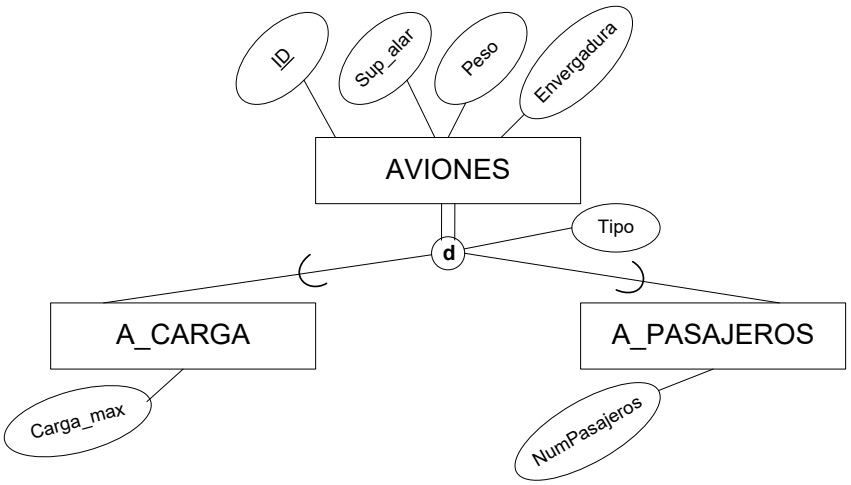
Las restantes pautas para la creación del esquema conceptual se aprecian en el diagrama:



2) Una Línea Área posee aviones para brindar servicios, los datos a registrar de los aviones son: ID, superficie alar, peso, envergadura, carga máxima y número de pasajeros. La carga máxima corresponde a los aviones de carga, y número de pasajeros corresponde a la capacidad de los aviones de pasajeros.

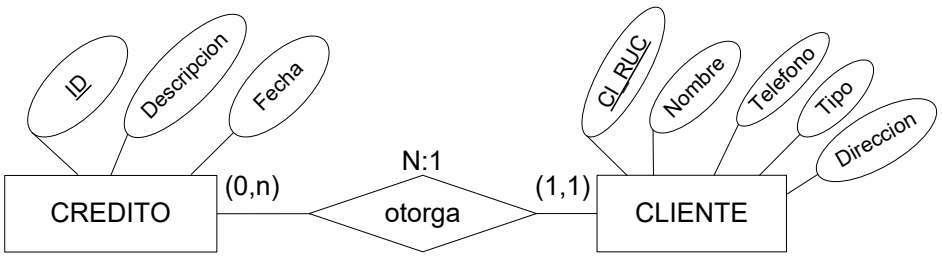
Solución:

ISBN: 978-631-6557-89-6



3) Un banco da créditos para compra de vehículos. Los créditos son tanto para personas naturales como para empresas. Los datos de personas con CI, Nombre, dirección y teléfono. Los datos de la empresa son: RUC, Nombre de empresa, dirección, y teléfono. Cada crédito posee un identificador, una descripción y una fecha.

Solución:

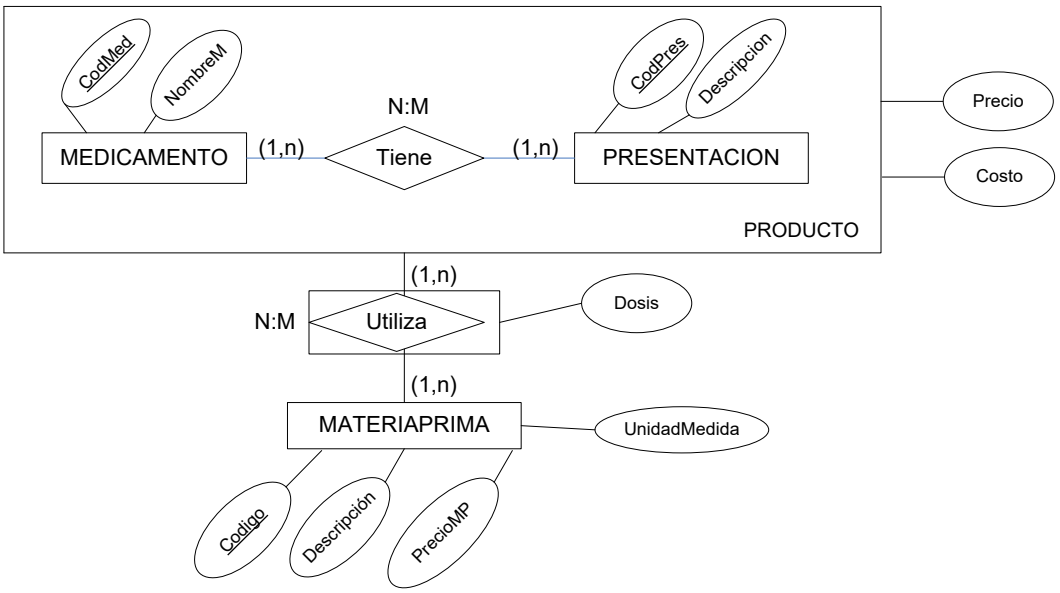


4) Una empresa desea controlar la producción de sus medicinas. Cada medicamento posee un código y un nombre. Además, cada medicamento puede tener diferentes presentaciones (tabletas, jarabe, capsulas, y otros) identificada con su código de presentación y una descripción. El medicamento de acuerdo con su presentación tiene un precio. A su vez, para la elaboración del medicamento de acuerdo con su presentación utiliza diferentes compuestos de materia prima, así también estos compuestos pueden utilizarse en una cantidad específica en la elaboración de diferentes productos (medicamentos según su presentación). Los compuestos de

ISBN: 978-631-6557-89-6

materia prima se identifican con un código, descripción, precio del compuesto y unidad de medida. Es importante también definir el costo de producción del producto (medicamentos según su presentación), para su cálculo es necesario conocer la materia prima y la cantidad utilizada.

Solución:



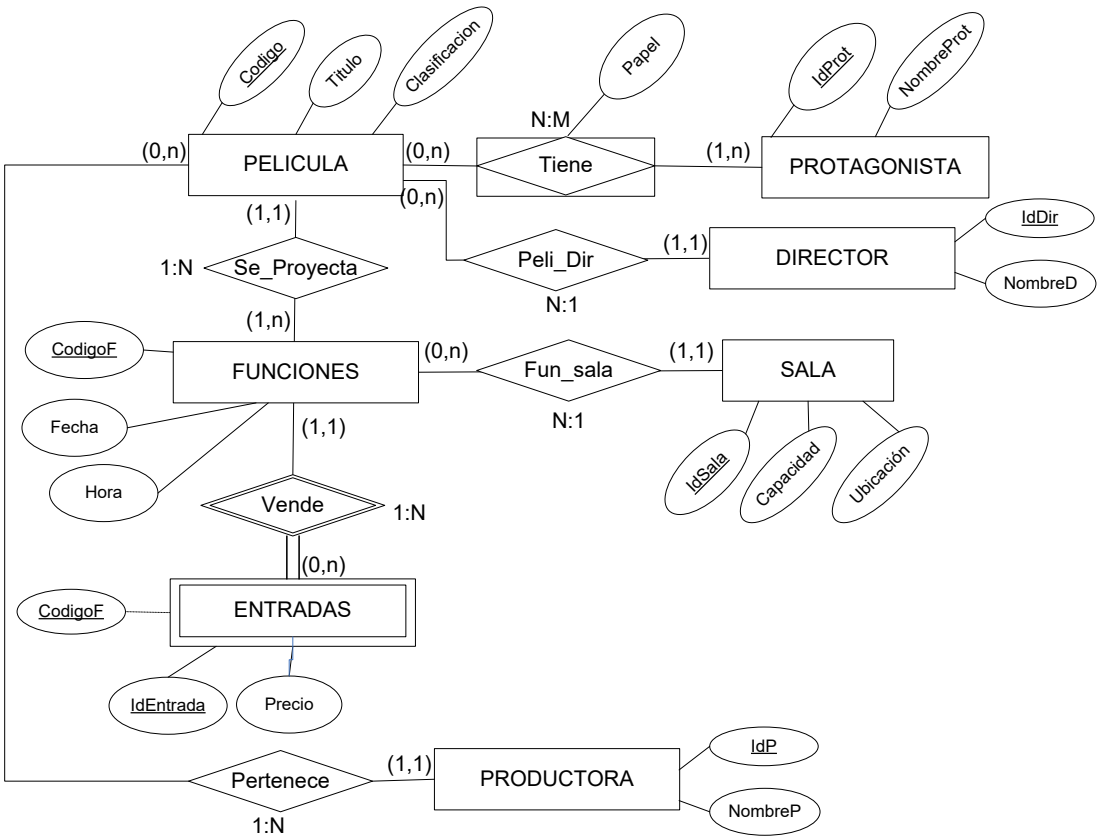
5) CineMas, es un grupo de salas para proyección de películas que requieren llevar un control de funciones programadas y de entradas vendidas.

A una película se asigna un código y título, también tiene una clasificación (adultos, menores hasta 12 años, todo público, e infantiles), y una productora (ejm. MGM, WallDisney,…). Cada película tiene protagonistas (principal, secundario) y un director. Una película se proyecta en varias funciones, una función tiene un código de función, fecha y hora, así como también la sala en la que se proyectará la película. Cada sala tiene su identificación, una capacidad y ubicación.

Se venden entradas por cada una de las funciones. Las entradas tienen un ID secuencial por función (inicia en 1 ……), y un precio

ISBN: 978-631-6557-89-6

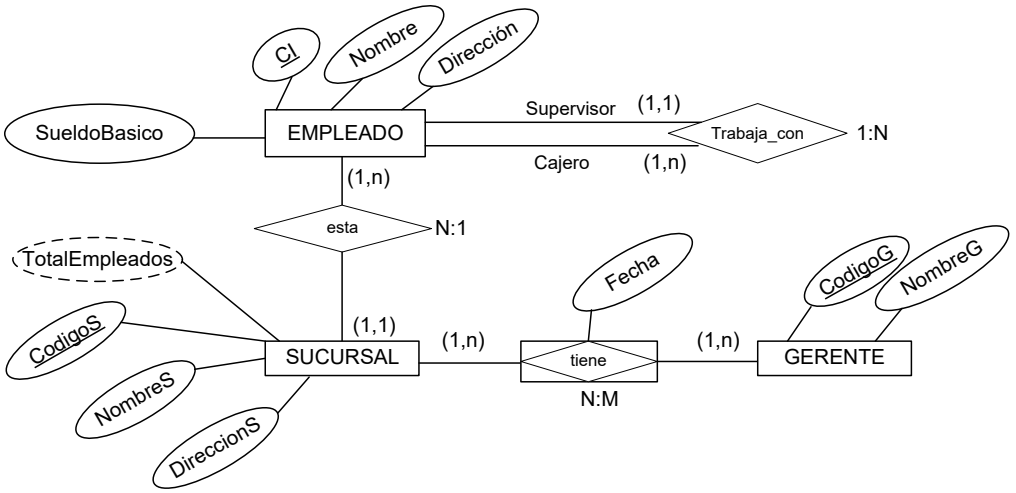
Solución:



6) Se necesita automatizar un centro bancario, donde: Un empleado posee los datos: cédula, nombre, dirección y sueldo básico. El empleado puede ser supervisor o cajero, cada cajero tienen un supervisor y este a su vez tiene un grupo de cajeros bajo su control. Además, es necesario conocer: la sucursal en la que trabaja el empleado, total de empleados que trabajan en la sucursal, y el Gerente actual de cada sucursal. Es importante listar los gerentes que han pasado por una sucursal, sabiendo además que un gerente puede ser asignado a diferentes sucursales en diferentes fechas.

ISBN: 978-631-6557-89-6

Solución:



7) Con base en el escenario del CASO: PROYECTOS DE GRADO, diseñe una base de datos relacional y obtenga el esquema conceptual (notación CHEN).

Ficha del CASO: PROYECTOS DE GRADO

ID_ proyecto: *Titulo:* *Fecha de presentación:* *Duración (meses):*	10 Desarrollo de una aplicación web para ayuda en terapia del habla 2021-10- 08 8		
***Estado*:** Aprobado: Rechazado: Anulado por abandono: Cierre:	 x x	***Fecha:*** 2021-10-15 ……. …….. 2022-07-15	
Tipo de proyecto:	Proyecto Técnico: x	Proyecto Integrador:	Emprendimiento:

Estudiante Proponente:

CODIGO	NOMBRE
1234 9876	Marcia Jara Pedro Ochoa

Tribunal:

DOCENTE	FUNCION
IVONNE ROJAS PATRICIO CEVALLOS	DIR ASE

Línea de Investigación:

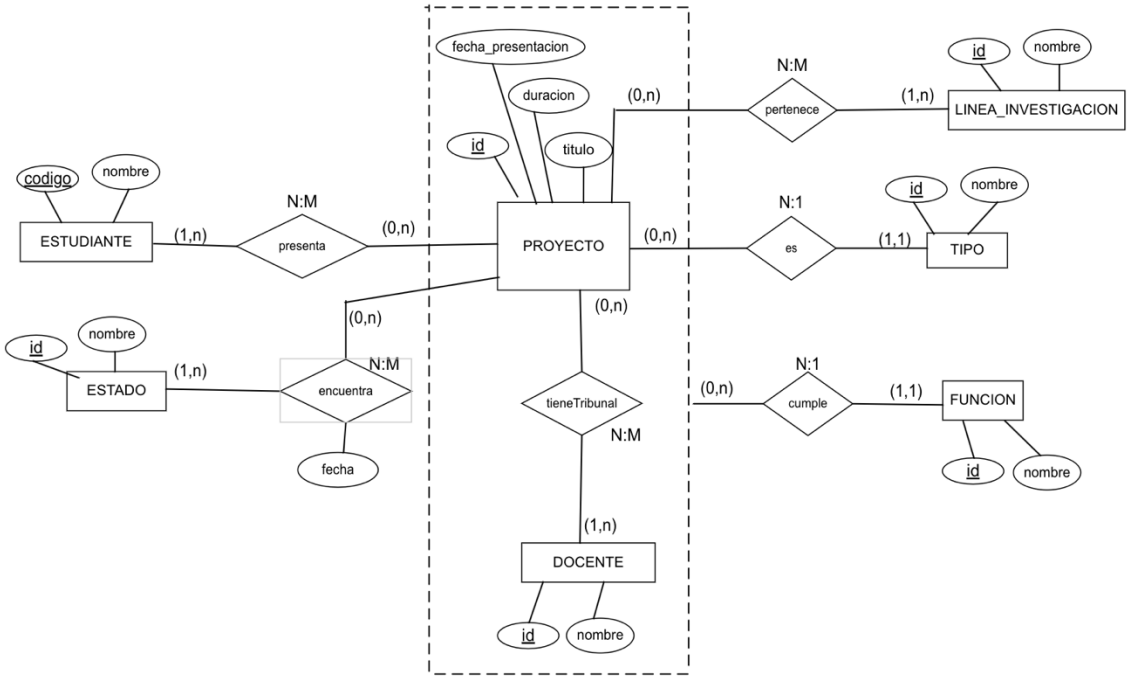
Ingeniería de software	X
Inteligencia Artificial	
Seguridad de Sistemas de Información	
Tecnología Educativa	X

NOTA:

- DIR: Director / ASE: Asesor
- Para un estudiante se puede registrar varios proyectos. Por ejemplo, en el caso de tener un "Anulado por abandono"

ISBN: 978-631-6557-89-6

Solución:



8) Con base en al escenario del CASO, diseñe una base de datos relacional y obtenga el esquema conceptual (notación CHEN).

CASO: ENCUESTA A HOGARES – SUJETOS DE ESTUDIO

IDENTIFICACION DE LA ENCUESTA: 2023004
FECHA DE LA ENCUESTA: 24/05/2023

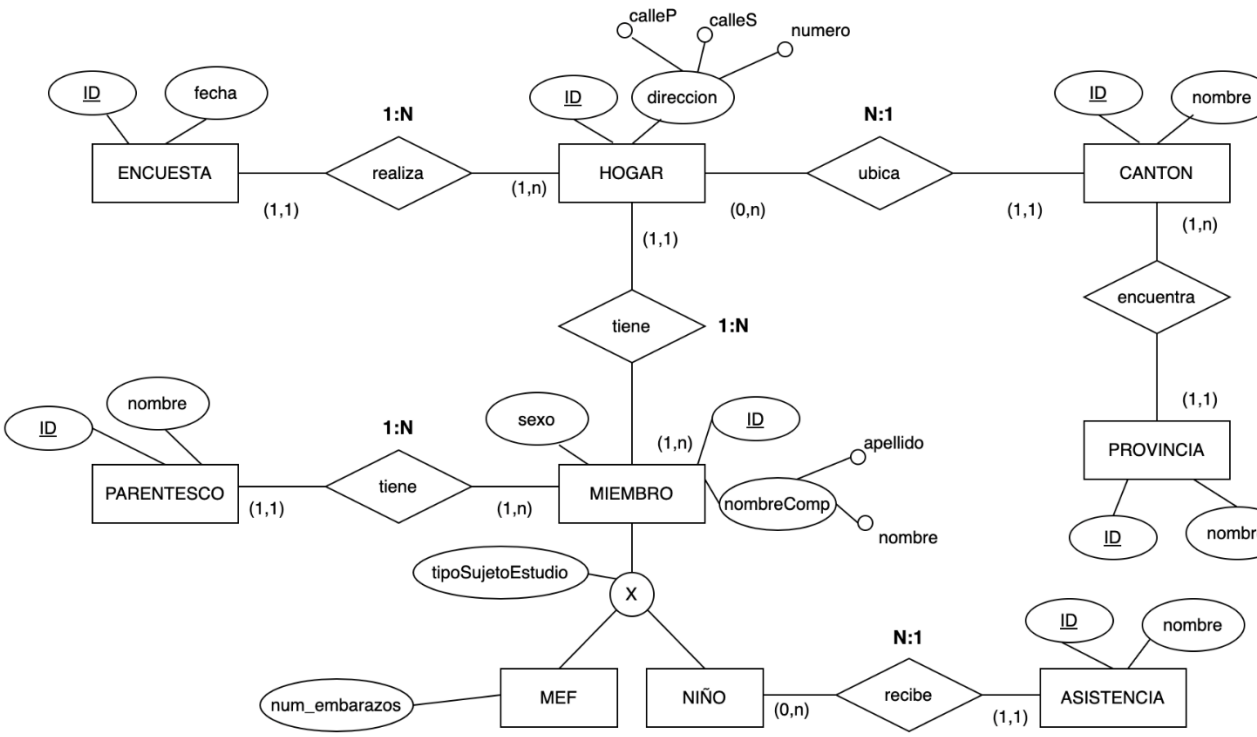
IDENTIFICADOR DE HOGAR:	H1234
PROVINCIA	TUNGURAHUA
CANTON	AMBATO
DIRECCION	
CALLE PRINCIPAL	
NUMERO	
CALLE SECUNDARIA	

MIEMBROS DEL HOGAR

APELLIDOS//NOMBRES	SEXO	PARENTESTO	TIPO DE SUJETO DE ESTUDIO	NUMERO DE EMBARAZOS - MEF (Mujer Estado Fertil)	ASISTENCIA ALIMENTARIA QUE TIENE - NIÑO
Luis Gracia	H: Hombre	1. Jefe			
Ana Villa	M: Mujer	2. Conyuge	MEF - Mujer Edad Fertil	1	
Valentina Gracia	M: Mujer	3. Hijo/hijastro	Niño		0. NINGUNO
Jose Luis Gracia	H: Hombre	3. Hijo/hijastro	Niño		1. DESAYUNO ESCOLAR
María Elena Flores	M: Mujer	4. Padres			

ISBN: 978-631-6557-89-6

Solución:



ISBN: 978-631-6557-89-6

CAPÍTULO 5

5 DISEÑO LÓGICO: TRANSFORMACIONES Y NORMALIZACIÓN

✶ Descripción

El objetivo del diseño lógico es convertir el esquema conceptual obtenido en la fase anterior en un *esquema lógico* que se ajuste a un modelo de datos específico, independientemente del DBMS que se vaya a utilizar para implementar la base de datos e independientemente de cualquier otra consideración física.

Entre los modelos de datos comercialmente más difundidos está el modelo relacional, este Texto se centra en el estudio de este modelo de datos, por lo tanto los esquemas conceptuales obtenidos deben transformarse en esquemas lógicos relacionales. A pesar de existir en el diseño lógico una independencia con respecto al DBMS, este al ser elegido para el diseño físico debe ser del tipo RDBMS (DBMS Relacionales).

En resumen, en el diseño lógico se refinan los esquemas conceptuales creados durante el diseño conceptual, eliminando las estructuras de datos que no se pueden implementar de manera directa sobre el modelo que soporta el DBMS, en el caso que nos ocupa, el *modelo relacional*. Una vez hecho esto, se obtiene un primer esquema lógico que se valida mediante la *normalización* y frente a las transacciones que el sistema debe llevar a cabo, tal y como se refleja en las especificaciones de requisitos de usuario. El esquema lógico ya validado se puede utilizar como base para el desarrollo de prototipos. Una vez finalizada esta fase, se dispone de un esquema lógico correcto, comprensible y sin ambigüedad.

✶ Actores

Diseñador, Usuario.

✶ Entrada

Esquema Conceptual de Bases de Datos.

✶ Salida

ISBN: 978-631-6557-89-6

Esquema Lógico de Bases de Datos.

✶ **Modelos y Técnicas para utilizar**

Los modelos y técnicas aplicables para el diseño lógico (ver Tabla 1.5 y la sección 3.3 del texto), son los indicados a continuación.

M*odelo Relacional*. El esquema lógico esperado como salida de la fase de diseño lógico obedece al modelo relacional.

Modelo Entidad Relación. Aplicable tanto al diseño conceptual como al lógico (ver Tabla 1-5 y sección 3.3). En un proceso de tres niveles, el esquema (DER) derivado del diseño conceptual se utiliza como entrada para el diseño lógico.

Normalización, es una técnica propia del modelo relacional en consecuencia el esquema lógico debe ser validado por ésta técnica.

✶ **Actividades**

Las actividades fundamentales de que consta el diseño lógico se indican a continuación:

a) Paso del MERE al Modelo Relacional
b) Validar cada esquema frente a las transacciones del usuario
c) Definir las restricciones de Integridad
d) Validar cada esquema mediante la normalización
e) Revisar el esquema lógico global con los usuarios

5.1 Transformación de un esquema conceptual a esquema lógico

En este capítulo se realizará la transformación de los esquemas conceptuales obtenidos a esquemas lógicos relacionales. El modelo relacional carece de ciertos rasgos de abstracción que se usan en los modelos conceptuales. Por lo tanto, un primer paso en la fase del diseño lógico consistirá en la conversión de esos mecanismos de representación de alto nivel en términos de las estructuras de bajo nivel disponibles en el modelo relacional.

ISBN: 978-631-6557-89-6

Para la transformación del esquema conceptual es necesario seguir ciertas reglas, las cuales por motivo de facilitar el aprendizaje se han dividido en dos grupos, como se indica a continuación:

- **Reglas de transformación del modelo básico**
  - Transformación de Entidades
  - Transformación de Relaciones N:M
  - Transformación de Relaciones 1:N
  - Transformación de Relaciones 1:1

- **Reglas de transformación del modelo extendido (MERE)**
  - Transformación de atributos compuestos
  - Transformación de atributos multivaluados
  - Transformación de atributos derivados
  - Transformación de una entidad débil
  - Transformación de una relación recursiva
  - Transformación de una relación reflexiva
  - Transformación de Generalización / especialización

5.1.1 Transformación de Entidades

En la transformación de una Entidad del modelo Entidad Relación se aplican los siguientes principios:

- Toda *entidad* **pasa a ser** una TABLA
- El *atributo identificador* de la entidad **pasa a ser** la *clave primaria de la* TABLA
- Los *atributos simples* de la entidad **pasan a ser** *columnas de la* TABLA

NOTA:
Lo correcto sería manejar el término RELACIÓN en vez de Tabla en el modelo relacional, pero esto provocaría confusión en el lector; razón por la cual, en este capítulo se utilizará: entidad, atributo y relación para el modelo E-R, y, tabla, columna, referencia para el modelo relacional.

- **Toda ENTIDAD pasa a ser una TABLA**

  Cada tipo de entidad se convierte en una tabla. Cada tabla que corresponda a un tipo de entidad llevará su mismo nombre. Ver ejemplo 5-1.

ISBN: 978-631-6557-89-6

- **El atributo identificador de la entidad pasa a ser la clave primaria de la TABLA**

  El atributo identificador de cada tipo de entidad pasa a ser la clave de la tabla. Se usa la cláusula PRIMARY KEY. Ver ejemplo 5-1.

  De existir identificadores alternativos o aspirantes, se crearán las columnas candidatas (aspirantes) aplicando la cláusula UNIQUE (valores únicos a ingresarse en la columna)

- **Los atributos simples de la entidad pasan a ser columnas de la TABLA**

  Un atributo de una entidad se transforma en una columna de la tabla, en la cual se ha transformado la entidad.

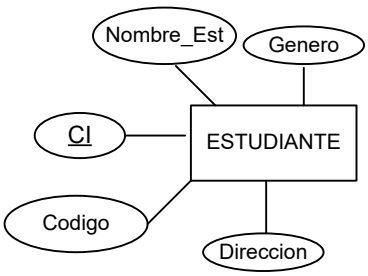
- **Dominio y Null**

  Al tratar los atributos también es importante conocer qué pasa con el dominio y obligatoriedad por Not null. En el modelo relacional estándar un dominio es un objeto más, propio de la estructura del modelo que, como tal, debe tener su definición concreta en el lenguaje de definición de datos DDL.

  En el esquema conceptual de la base de datos ya se define el dominio de cada uno de los atributos de la entidad, mismo que pasará a las columnas transformadas. Es decir, un atributo del tipo char(2), pasará a ser columna char(2). Así también, un atributo obligatorio de la entidad pasa a ser una columna de tabla que no admite nulos (Not null). Una regla asociada a un atributo pasa a ser de la columna transformada (CHECK).

Ejemplo 5-1. Modelo Básico: Transformación de Entidad a Tabla

Supongamos que tenemos la siguiente entidad: ESTUDIANTE



Transformación

ESTUDIANTE (CI, Codigo, Nombre_Est, Genero, Direccion)

ISBN: 978-631-6557-89-6

CI: es la clave primaria (Primary Key) de la tabla estudiante, asume el dominio del atributo CI

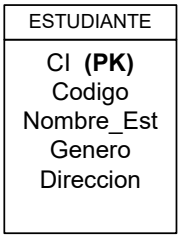
Codigo: de atributo aspirante pasó a ser una columna clave candidata

Nombre_Est: asume el tipo y tamaño de dato del atributo

Genero: asumen tipo y tamaño de dato, además CHECK: "M" o "F"

Direccion: asume el tipo y tamaño de dato del atributo

De manera gráfica la tabla se presenta de la siguiente manera:



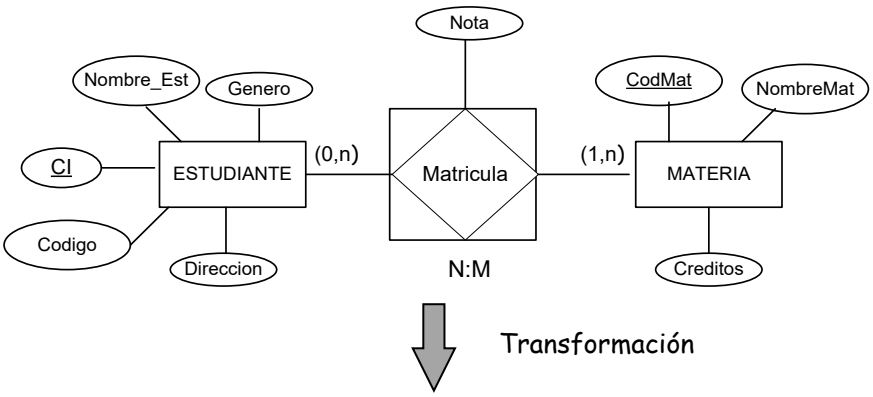
5.1.2 Transformación de Relaciones N:M

En la transformación de una relación con cardinalidad N:M del modelo Entidad Relación se aplican los siguientes principios:

- Se aplica el principio: "*Toda relación N:M pasa a ser una TABLA*".
  Un tipo de relación N:M se transforma en una Tabla, los atributos propios de la relación pasarán como columnas de la tabla. A su vez, la clave primaria (PK) de la tabla se conformará de la concatenación de los identificadores de los tipos de entidad que asocia la relación.
- El nombre de la tabla asumirá el mismo nombre de la relación.
- Cada una de las columnas que forman la clave primaria de la tabla derivada de un tipo de relación N:M, son clave foránea respecto de cada una de las tablas derivadas de los tipos de entidad que asocia. Esto se especifica a través de la cláusula FOREIGN KEY (FK) dentro de la sentencia de creación de la tabla.

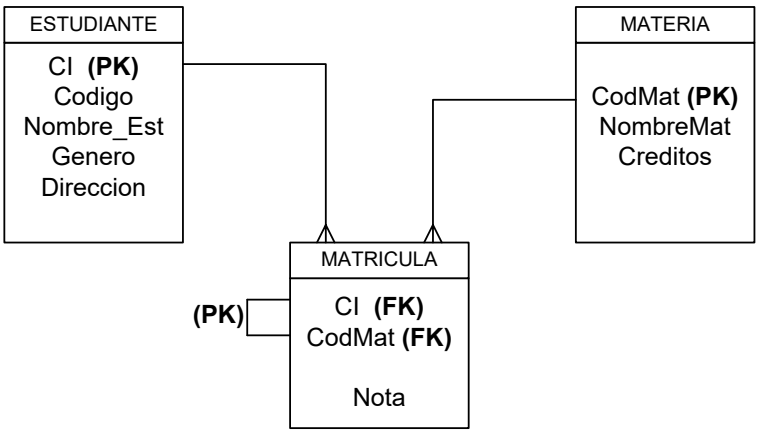
ISBN: 978-631-6557-89-6

Ejemplo 5-2. Modelo Básico: Transformación de relaciones N:M



ESTUDIANTE (CI, Codigo, Nombre_Est, Genero, Direccion)
MATERIA (CodMat, NombreMat, Creditos)
MATRICULA (CI, CodMat, Nota)

Gráficamente la transformación se ilustraría de la siguiente manera:



- **¿Qué ocurre con los mínimos y máximos?**
  - El ***min*** se transforma en la admisión o no de valores nulos (Null o Not null) en la tabla.
  - El ***max*** especifica restricciones. Es decir, representa la *Cardinalidad*.

ESTUDIANTE

CI	Codigo	Nombre_Est	Genero	Direccion
0602076874	0001	Ana López	F	Guayaquil y Espejo
1708090222	0002	Luis Garzón	M	España y Junín
1807654320	0003	Pedro Barba	M	Boyacá y Espejo

ISBN: 978-631-6557-89-6

MATERIA

CodMat	NombreMat	Creditos
M001	Base Datos I	4
M002	Redes	6
M003	Aplicaciones Web	6
M004	Sistemas operativos	4

MATRICULA

CI	CodMat	Nota
0602076874	M001	15
1708090222	M001	
1708090222	M003	
1807654320	M002	17
1807654320	M003	

Como se aprecia en la tabla con datos MATRICULA, la cardinalidad N:M se expresa cuando:

- Una materia (M001) puede ser tomada por varios estudiantes (0602076874 y 1708090222), pero al ser min=0 (no obligatorio), habrá materias sin estudiantes como el caso de M004.
- Mientras que un estudiante (1807654320) puede tomar varias materias (M002 y M003), necesariamente, es decir con min=1 (obligatorio). En otras palabras, para ser estudiante debe mínimo tomar una materia obligatoriamente.
- En cuanto a la columna Nota, podrá ser Null. Esto si consideramos que en determinada instancia estos valores podrán ser ingresados.

5.1.3 Transformación de Relaciones 1:N

En la transformación de la relación 1:N se aplica uno de dos posibles métodos, que son: propagación de identificador (utiliza integridad referencial) o se crea una nueva tabla.

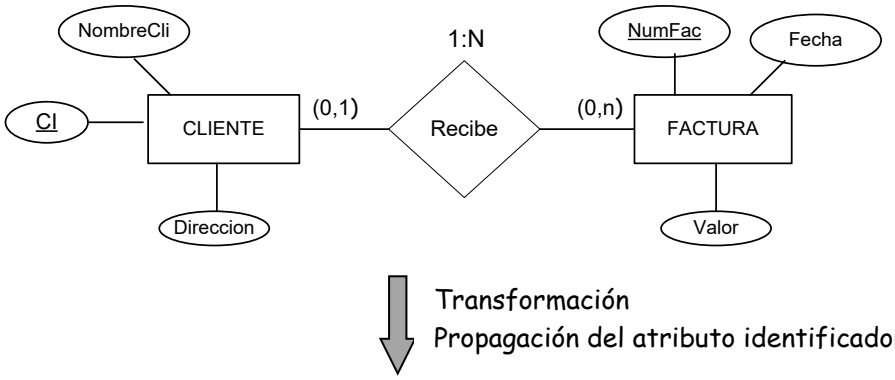
- **Método 1: Propagación del identificador**

La propagación del identificador es el método de transformación más habitual. Las entidades se transforman en tablas, entonces, **propagar el atributo identificador** de la entidad de cardinalidad 1, a la que tiene cardinalidad N. De manera que, en la tabla

ISBN: 978-631-6557-89-6

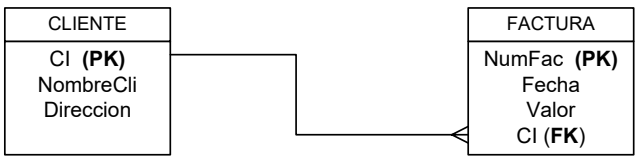
transformada con cardinalidad N contendrá la clave foránea (FOREIGN KEY). En otras palabras, se corresponde a integridad referencial del modelo relacional.

Ejemplo 5-3. Modelo Básico: Transformación de relaciones 1:N – Propagación del identificador



Transformación
Propagación del atributo identificador

CLIENTE (CI, NombreCli, Direccion)
FACTURA (NumFac, Fecha, Valor, CI)



Las tablas obtenidas de la transformación se corresponden a la restricción de integridad referencial padre (CLIENTE) e hija (FACTURA).

- **¿Qué ocurre con los mínimos y máximos en relaciones 1:N aplicando propagación de clave?**

  - **Si min=0**

    (0,1) : La entidad que interviene con ocurrencia (1) es opcional

    Si se propaga, la clave foránea será opcional. Habrá ocurrencias de la otra tabla con la que se referencia, donde la clave foránea sea Null.

    En el Ejemplo 5-3, con min=0 nos indica que: Una factura puede ser recibida *no obligatoriamente por un cliente*; es decir, que habrá facturas que no se asocien con cliente alguno. Por lo tanto en FACTURA la clave foránea (CI) puede ser Null.

    (0,n): La entidad que interviene con varias ocurrencia es opcional.

ISBN: 978-631-6557-89-6

Si se propaga, no se toman medidas al respecto.

En el Ejemplo 5-3, con min=0 nos indica que: Un CLIENTE puede recibir *varias facturas no obligatoriamente*; es decir, habrán clientes que no reciban factura alguna.

CLIENTE

CI	NombreCli	Direccion
0607520401	Jorge Bravo	Guayaquil y Espejo
1802035239	Alcides Reinoso	España y Junín
0606643213	Blanca Pérez	Boyacá y Espejo
1702345670	María Oleas	Argentinos y España
0203746532	Marcelo Galeas	Boyacá y Junín

FACTURA

NumFac	Fecha	Valor	CI
001	01/11/2007	50	0606643213
002	10/01/2008	20	0606643213
003	12/02/2008	15	**NULL**
004	12/02/2008	80	1702345670

- **Si min=1**

(1,1) : La entidad interviene con una (1) y solo una ocurrencia, obligatoriamente.

Si se propaga, la clave foránea será obligatoria (Not null).

En el Ejemplo 5-3, cambiando a min=1 nos indica que: Una FACTURA es recibida *obligatoriamente por un cliente*; es decir, que la factura se asocia necesariamente con un CLIENTE. Por lo tanto en FACTURA la clave foránea (CI) NO puede ser Null.

(1,n): La entidad que interviene con una o varias ocurrencias, obligatoriamente.

Si se propaga, no se toman medidas al respecto.

En el Ejemplo 5-3, cambiando a min=1 nos indica que: Un CLIENTE recibe *varias facturas obligatoriamente*; es decir, los clientes se asocian siempre con facturas, al menos una.

Los datos en la tabla FACTURA quedarían de la siguiente manera:

FACTURA

ISBN: 978-631-6557-89-6

NumFac	Fecha	Valor	CI
001	01/11/2007	50	0606643213
002	10/01/2008	20	0606643213
003	12/02/2008	15	1802035239
004	12/02/2008	80	1702345670

- **Método 2: Crear una nueva tabla de la relación**

El método 2 de transformación se aplica en los siguientes casos:

a) Cuando la relación tiene atributos propios. Primero verificar que los atributos corresponden a la relación.
b) Cuando el número de ocurrencias relacionadas de la entidad que propaga su identificador es muy pequeño, por tanto, hay la posibilidad de que existan muchos valores nulos.

Sea para el caso a) como para el b), la transformación se da de la siguiente manera: Cada entidad se transforma en una tabla con clave principal (PK) que provienen del identificador de la entidad correspondiente. Y, se construye una nueva tabla proveniente de la relación, de la misma forma como se lo realiza en una transformación N:M, donde la nueva tabla se conforma por los identificadores de cada entidad que asocia la relación, más atributos propios de la misma; con la diferencia que, la clave de esta nueva tabla será el identificador de la entidad que participa con cardinalidad N y tendrá como clave foránea el identificador de la otra entidad.

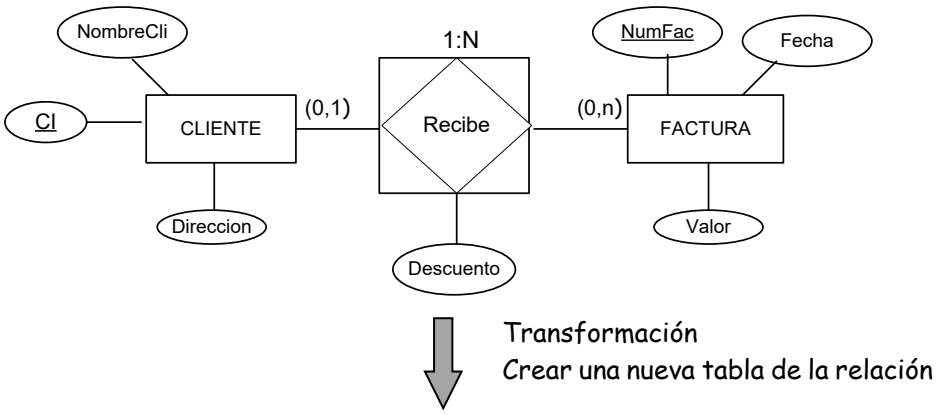
Como se observa en el Ejemplo 5-4, las tablas resultado de la transformación, CLIENTE es padre respecto a la clave foránea CI que se encuentra en RECIBE. Cada CI de CLIENTE se asociará con varios hijos CI en RECIBE.

FACTURA es padre con respecto a la clave foránea NumFac de RECIBE, con la diferencia que cada NumFac de FACTURA tendrá un solo hijo NumFac en RECIBE (*Integridad referencial*: Un padre - un solo hijo; NumFac es PK en FACTURA, y también es PK y FK en RECIBE)

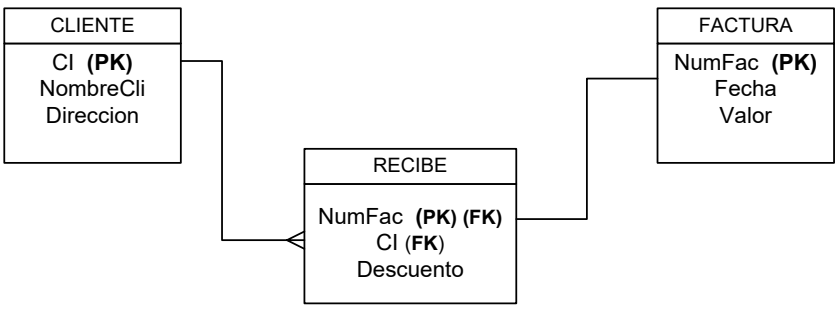
RECIBE contendrá únicamente aquellas tuplas (filas) donde FACTURA tenga CLIENTE, caso contrario, no tiene sentido insertar la fila.

ISBN: 978-631-6557-89-6

Ejemplo 5-4. Modelo Básico: Transformación de relaciones 1:N – Creación de una nueva tabla



CLIENTE (CI, NombreCli, Direccion)
FACTURA (NumFac, Fecha, Valor)
RECIBE (NumFac, CI, Descuento)



- **¿Qué ocurre con los mínimos y máximos en relaciones 1:N creando una nueva tabla?**

  - **Si min=0**

    (0,1): La entidad que interviene con ocurrencia (1) es opcional.

    Si se crea una nueva tabla, habrá identificadores de la entidad opcional que no aparezcan en ninguna ocurrencia de esta tabla.

    En el Ejemplo 5-4, con min=0 nos indica que: Una factura puede ser recibida *no obligatoriamente por un cliente*; es decir, que ciertos NumFac no aparecerán en ninguna ocurrencia de RECIBE.

    (0,n): La entidad que interviene con varias ocurrencia es opcional.

    Si se crea una nueva tabla, habrá identificadores de la entidad opcional que no aparezcan en ninguna ocurrencia de esta tabla.

ISBN: 978-631-6557-89-6

Tomando el mismo Ejemplo 5-4, con min=0, ciertos CI no aparecerán en ninguna ocurrencia de RECIBE.

CLIENTE

CI	NombreCli	Direccion
0607520401	Jorge Bravo	Guayaquil y Espejo
1802035239	Alcides Reinoso	España y Junín
0606643213	Blanca Pérez	Boyacá y Espejo
1702345670	María Oleas	Argentinos y España
0203746532	Marcelo Galeas	Boyacá y Junín

FACTURA

NumFac	Fecha	Valor
001	01/11/2007	50
002	10/01/2008	20
003	12/02/2008	15
004	12/02/2008	80

RECIBE

NumFac	CI	Descuento
001	0606643213	3
002	0606643213	2
004	1702345670	1

Con el ejemplo de tablas de datos, observamos que, RECIBE asocia cliente con factura. Una FACTURA tiene un CLIENTE, pero no necesariamente, como el caso de NumFac=003 que no se encuentra en RECIBE. También tenemos CI que no están en RECIBE.

NOTA:
RECORDAR: El Método 2 se aplica cuando existen muchos valores nulos. Con este ejemplo no se puede precisar, ya que con solo NumFac=003 que no tenga CI, no justificaría el método 2.

- **Si min=1**

  (1,1) : La entidad interviene con una (1) y solo una ocurrencia.

  Si se crea una nueva tabla, en ella debería haber una tupla por cada ocurrencia de la entidad que interviene con cardinalidad N.

ISBN: 978-631-6557-89-6

(1,n): La entidad que interviene con una o varias ocurrencias.

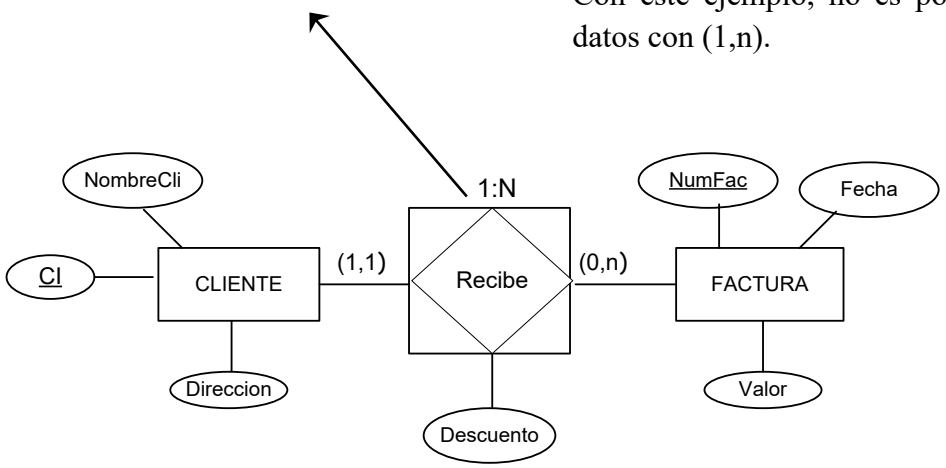
Si se crea una nueva tabla, en ella debería haber al menos una tupla por cada ocurrencia de la entidad que interviene con cardinalidad 1.

Dado el estado del Ejemplo 5-4, y acogiendo min=1, la tabla RECIBE quedaría de la siguiente manera:

RECIBE

NumFac	CI	Descuento
001	0606643213	3
002	1802035239	2
003	0203746532	
004	1702345670	3

De la tabla con datos se puede apreciar que: RECIBE contiene una fila por cada uno de los NumFac de FACTURA, cumpliendo con (1,1). Sin embargo, en RECIBE no están todos los CI, por consiguiente no puede ser (1,n), sino (0,n). Con este ejemplo, no es posible representar datos con (1,n).



NOTA:
En RECIBE estarán solo aquellas ocurrencias que no tengan Null. Para generalizar el concepto se puede decir que: en RECIBE no se encontrarán ocurrencias en los que NumFact no tenga CI asociadas, o CI que no tengan NumFac asociadas.

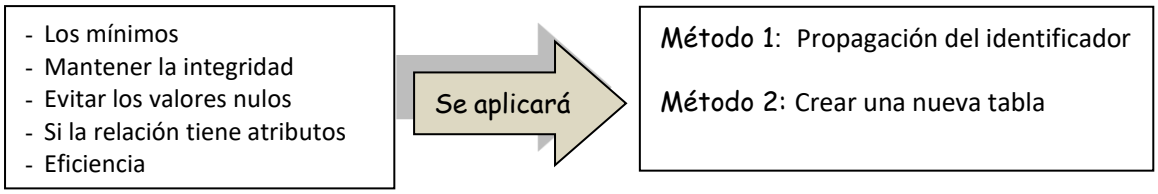
Para garantizar que se cumpla lo indicado, es necesario utilizar algunos mecanismos tales como: validación, procedimientos almacenados y disparadores.

5.1.4 Transformación de Relaciones 1:1

Una relación 1:1 es un caso particular de 1:N y de N:M, por lo que no hay regla fija para su transformación al modelo relacional. Sin embargo, en el texto se propone dos métodos de transformación.

ISBN: 978-631-6557-89-6

Dependiendo de:

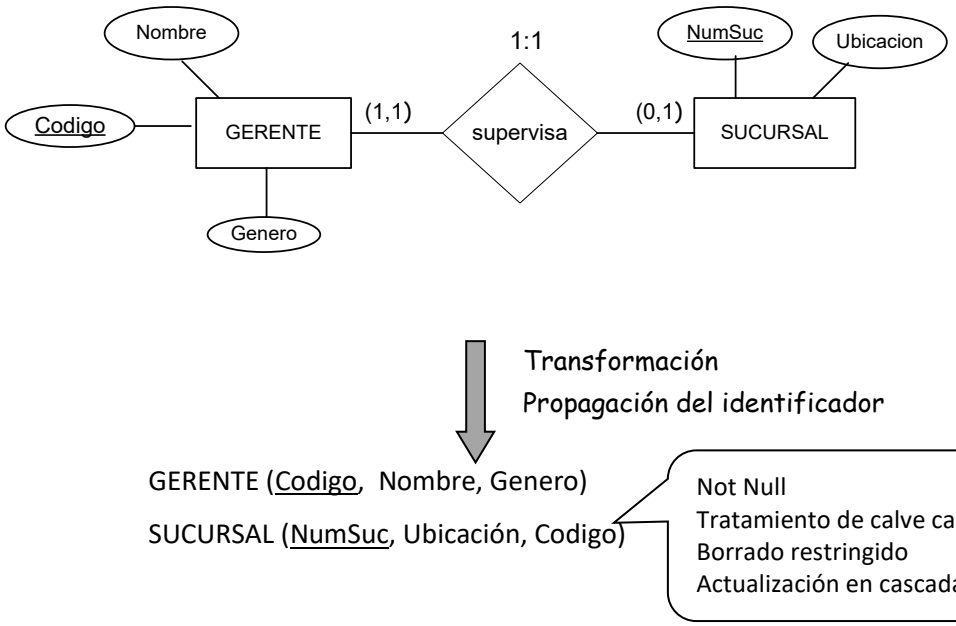


- **Método 1: Propagación del identificador**

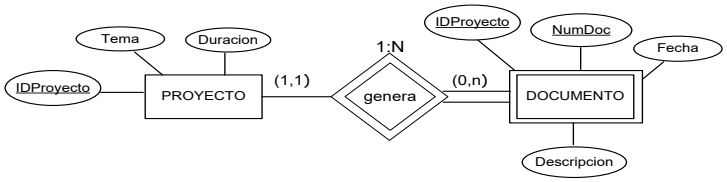
La propagación del identificador es el método de transformación más habitual. Y se aplica en los siguientes casos:

a) Si una de las entidades tiene mínimos y máximos (1,1), mientras que la otra entidad tiene (0,1), es conveniente propagar el identificador de la entidad (1,1) a la tabla resultante de la entidad (0,1).
b) Si una de la entidad tiene mínimos y máximos (1,1), y la otra entidad también tiene (1,1); entonces, se deben elegir el identificador de cualquiera de las dos entidades que será propagado.

Ejemplo 5-5. Modelo Básico: Transformación de relaciones 1:1 – Propagación del identificador



ISBN: 978-631-6557-89-6



GERENTE

Codigo	Nombre	Genero
G1	Danilo Flores	M
G2	Denis Espinoza	M
G3	Blanca Soria	F
G4	María Oleas	F
G5	Marcelo Galeas	M

SUCURSAL

NumSuc	Ubicacion	Codigo
S1	Terminal	G1
S2	Estadio	G3
S3	Plaza Toros	G2
S4	Camal	G5

En el Ejemplo 5-5, el Gerente G4 no tiene sucursal (opcional), toda sucursal tiene obligadamente un Gerente (por cada NumSuc no admite que Codigo sea Null).

En el caso que, el mismo Ejemplo 5-5 de transformación tenga (1,1) para ambas entidades; implica que, una opción sería la propagación de Codigo hacia SUCURSAL. Y, la otra opción es que, NumSuc se propaga a GERENTE, como se indica a continuación:

GERENTE (Codigo, Nombre, Genero, NumSuc)

SUCURSAL (NumSuc, Ubicacion)

- **Método 2: Crear una nueva tabla**

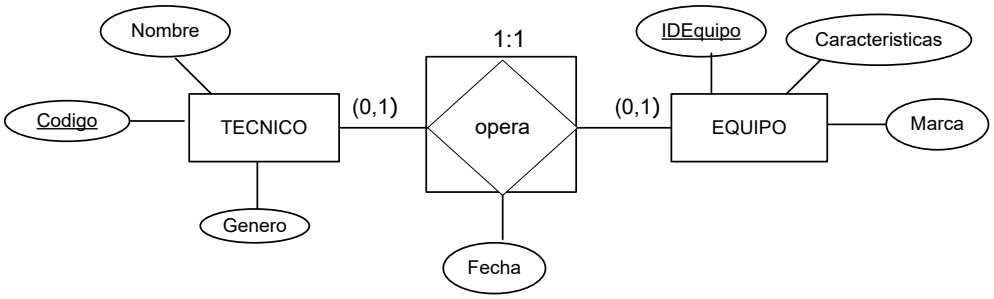
La creación de una nueva tabla es un método que se recomienda aplicar para los siguientes casos:

ISBN: 978-631-6557-89-6

a) Las entidades involucradas tienen min=0 (ambas), esto evitará valores nulos en las claves foráneas. Además, las entidades mantienen su independencia mediante tablas separadas.

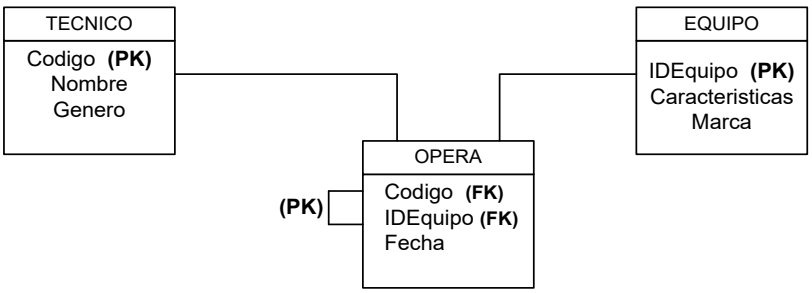
b) La Relación tiene atributos propios.

Ejemplo 5-6. Modelo Básico: Transformación de relaciones 1:1 – Creación de una nueva tabla



Transformación
Crear una nueva tabla de la relación

TECNICO (Codigo, Nombre, Genero)
EQUIPO (IDEquipo, Caracteristicas, Marca)
OPERA (Codigo, IDEquipo, Fecha)



TECNICO

Codigo	Nombre	Genero
T1	Danilo Flores	M
T2	Mauricio Galeas	M
T3	Blanca Soria	F
T4	María Oleas	F
T5	Marcelo Ortiz	M

EQUIPO

ISBN: 978-631-6557-89-6

IDEquipo	Caracteristicas	Marca
E1	Taladro de percusión Gladiador	Dewalt
E2	Lan Tester para redes y telefonía capacidad 300 mts	Lanpro
E3	Tester digital	Sunwa

OPERA

Codigo	IDEquipo	Fecha
T1	E2	01/03/2008
T2	E3	15/04/2008

En OPERA la clave primaria es Codigo + IDEquipo, pero un hijo se corresponde por cada padre, es decir que si E2 fue asignado a un técnico no podrá entregarse a otro técnico. En otras palabras, en la tabla OPERA habrá sólo una fila con E2. Igual ocurre con Codigo de OPERA respecto a su tabla padre TECNICO (ejemplo: habrá una sola fila con T1)

5.1.5 Transformación de atributos compuestos

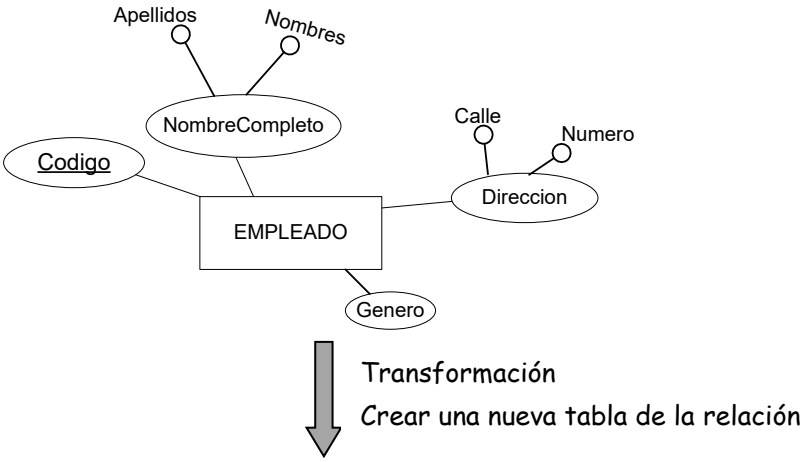
Un atributo compuesto puede provenir de una entidad o de una relación. Pero el modelo relacional admite sólo atributos simples.

Para cualquier atributo compuesto, ya sea que forme parte de una entidad o de una relación, puede ser transformado mediante uno de los métodos que se presentan a continuación:

- **Método 1:** Eliminar el atributo compuesto considerando todos sus componentes como atributos individuales.
- **Método 2:** Eliminar los componentes individuales y considerar el atributo compuesto entero como un sólo atributo.

ISBN: 978-631-6557-89-6

Ejemplo 5-7. Modelo Extendido: Transformación de atributos compuestos



EMPLEADO (Codigo, Apellidos, Nombres, Direccion, Genero)

Qué determina el método se transformación a elegir, definitivamente los requerimientos del usuario. En el mismo Ejemplo 5-7, se ha aplicado los dos métodos tomando en cuenta lo siguiente criterios:

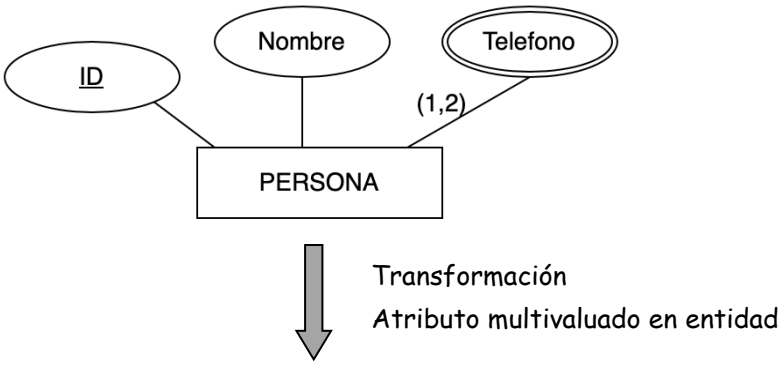
- "*Consultas continuas basadas en los Apellidos del empleado, se recomienda el método 1*". Por tanto, en la transformación no se considera el atributo NombreCompleto; sino que, se crea dos columnas correspondientes a los atributos individuales Apellidos y Nombres.
- En cambio, si "*la dirección no amerita consultas específicas (consultas por calle o por número), entonces se recomienda el método 2*". Por tanto, en la transformación no se considerar los atributos individuales Apellidos y Nombres; sino que, se crea la columna correspondiente al atributo compuesto Direccion.

ISBN: 978-631-6557-89-6

5.1.6 Transformación de atributos multivaluados

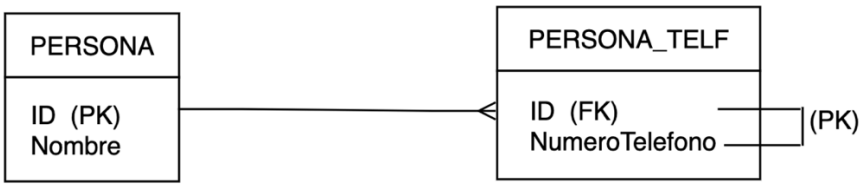
Un atributo multivaluado puede encontrarse en una Entidad o en una Relación. Pero el modelo relacional admite sólo atributos simples. El Ejemplo 5-8 muestra la transformación de un atributo multivaluado en una entidad. Mientras que, el Ejemplo 5-9 presenta la transformación del atributo multivaluado en una relación.

Ejemplo 5-8. Modelo Extendido: Transformación del atributo multivaluado en entidad



PERSONA (ID, Nombre)

PERSONA_TELF (ID, NumeroTelefono)



En un contexto real representado en el DER del Ejemplo 5-8, una persona debe registrar hasta dos (2) números telefónicos móvil, al menos uno que sirva de contacto.

La entidad se transforma en la tabla PERSONA, y para el atributo multivaluado Telefono se crea otra tabla PERSONA_TELF; en la cual, de acuerdo con la regla de participación y cardinalidad (1,2), debe registrarse al menos un número telefónico y como máximo dos. Sin embargo, para que pueda cumplirse con esta regla del negocio, y registrase hasta dos números telefónicos; según la restricción de integridad del modelo relacional, la clave primaria (PK) de la tabla PERSONA_TELF deberá conformarse por las columnas ID + NumeroTelefono

ISBN: 978-631-6557-89-6

En PERSONA_TELF estarán todas las ocurrencias (filas) de PERSONA, ya que todas tienen teléfono al menos uno.

PERSONA

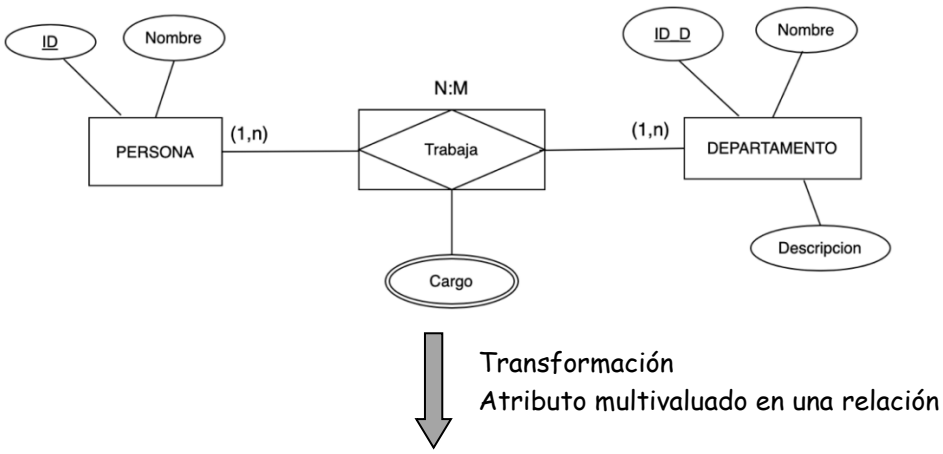
ID	Nombre
P1	Danilo Flores
P2	Mauricio Galeas
P3	Blanca Soria

PERSONA_TELF

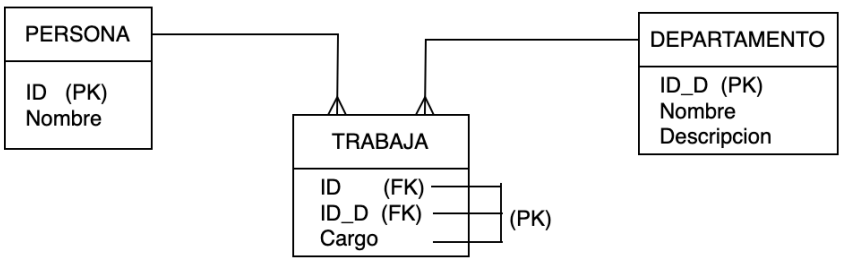
ID	NumeroTelefono
P1	2987656
P1	2431567
P2	3425649
P3	3866728

Ejemplo 5-9. Modelo Extendido: Transformación del atributo multivaluado en relación

En caso del atributo multivaluado en una relación, el tratamiento de transformación es el mismo aplicado en el Ejemplo 5-8.



PERSONA (ID, Nombre)
DEPARTAMENTO (ID_D, Nombre, Descripcion)
TRABAJA (ID, ID_D, Cargo)



NOTA:
Las transformaciones de este apartado se sustentan con la normalización que se revisa más adelante. Además, NO todos los casos de atributos multivaluados necesariamente se transforman de la forma hasta ahora indicada, se requiere conceptos de Normalización.

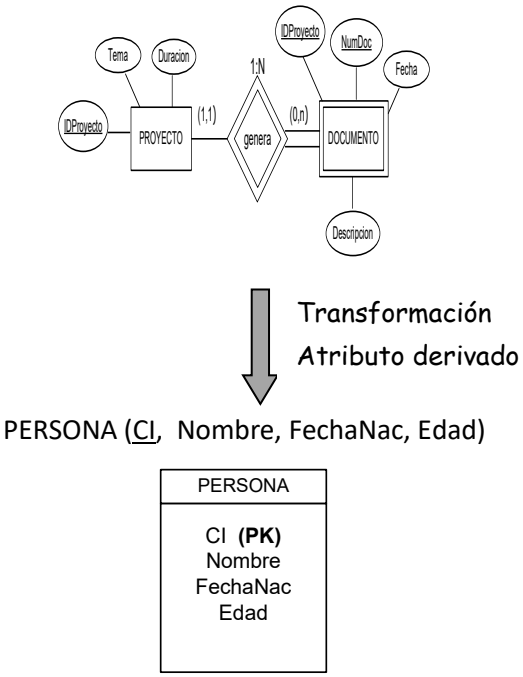
ISBN: 978-631-6557-89-6

5.1.7 Transformación de atributos derivados

Al momento de la transformación de un atributo derivado se tiene dos alternativas:

a) Convertirlo en una columna calculada del modelo relacional. Es decir, la columna no se crea físicamente en la tabla, el dato del cálculo no se almacena solo se utiliza para consultas.
b) Convertirlo en columna que se crea físicamente en la tabla, el dato se almacena y resulta de validaciones CHECK y disparadores.

Ejemplo 5-10. Modelo Extendido: Transformación de atributos derivados



El atributo derivado Edad se ha creado como columna en la tabla PERSONA, cada vez que se ingresa la fecha de nacimiento (FechaNac) se activará el disparador que realizará el cálculo de la edad de la persona y este valor se almacenará en la columna Edad.

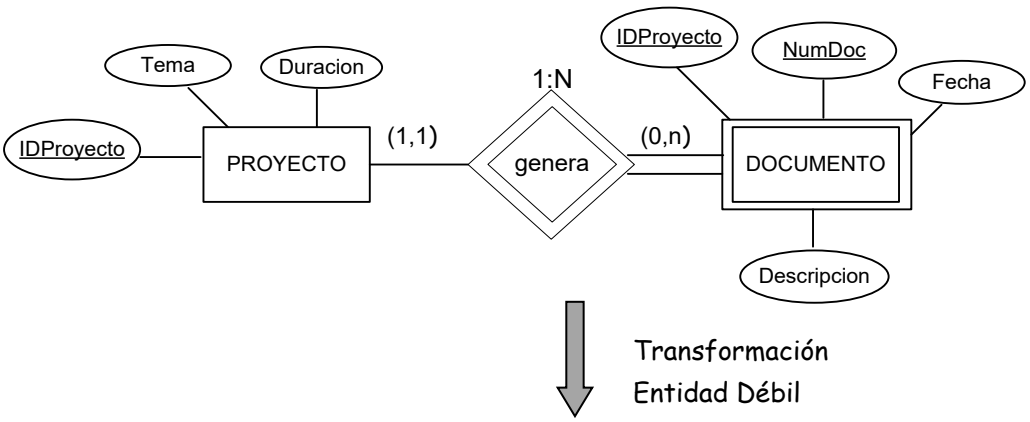
5.1.8 Transformación de una Entidad Débil

La transformación de una entidad débil consiste en la propagación del identificador de la entidad fuerte hacia la entidad débil. Al transformar la entidad débil en tabla, la clave

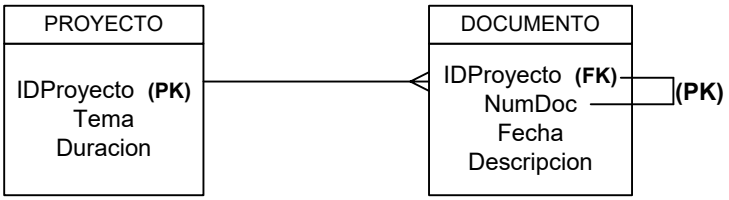
ISBN: 978-631-6557-89-6

primaria de ésta será una compuesta, conformada por el identificador propagado más el identificador propio de la entidad débil.

Ejemplo 5-11. Modelo Extendido: Transformación de una entidad débil



PROYECTO (IDProyecto, Tema, Duracion)
DOCUMENTO (IDProyecto, NumDoc, Fecha, Descripcion)



PROYECTO

IDProyecto	Tema	Duracion
PR1	Estudio comparativo de ETLs	6
PR2	Integración de Base de Datos	8
PR3	Implantación de una LAN en el Municipio de Riobamba	6
PR4	JOLAP para análisis de información	12

DOCUMENTO

IDProyecto	**NumDoc**	Fecha	Descripcion
PR1	1	02/01/2008	Pedido de aprobación
PR1	2	02/01/2008	Nombramiento de Tribunal del proyecto
PR4	1	28/03/2008	Presentación del proyecto

De acuerdo con los mínimos del Ejemplo 5-11, tenemos que, cada documento generado depende necesariamente de un proyecto. Pero también tenemos proyectos que

ISBN: 978-631-6557-89-6

no tienen documentos asociados. El mismo hecho que, IDProyecto y NumDoc conformen la clave primaria de la tabla DOCUMENTO, hace que ninguno de éstos pueda ser nulo.

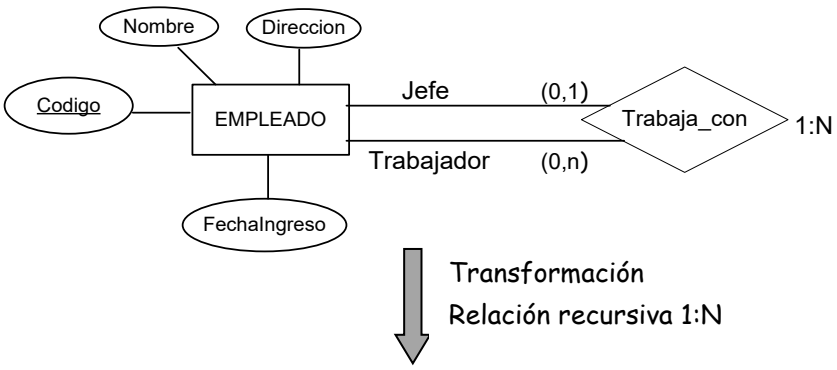
5.1.9 Transformación de una Relación Recursiva

Una relación recursiva, también llamada reflexiva puede ser del tipo 1:N o N:M, y en función de estas cardinalidades se realiza la transformación.

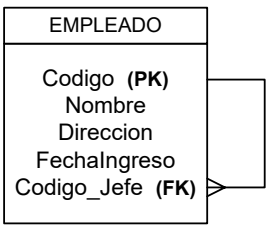
- **Relación Recursiva 1:N**

Para una relación recursiva con cardinalidad 1:N, se obtiene una tabla en la cual el identificador de la entidad pasa a ser clave primaria de la tabla resultante; la cual, se referencia a sí misma a través de una clave foránea.

Ejemplo 5-12. Modelo Extendido: Transformación de una relación recursiva 1:N



EMPLEADO (Codigo, Nombre, Direccion, FechaIngreso, Codigo_Jefe)



EMPLEADO

Codigo	Nombre	Direccion	FechaIngreso	Codigo_Jefe
0607	Jorge Bravo	Guayaquil y Espejo	10/05/2003	0606
1802	Alcides Reinoso	España y Junín	24/02/2001	0606
0606	Blanca Pérez	Boyacá y Espejo	12/07/2000	0203
1702	María Oleas	Argentinos y España	13/08/2004	0606
0203	Marcelo Galeas	Boyacá y Junín	15/09/1998	

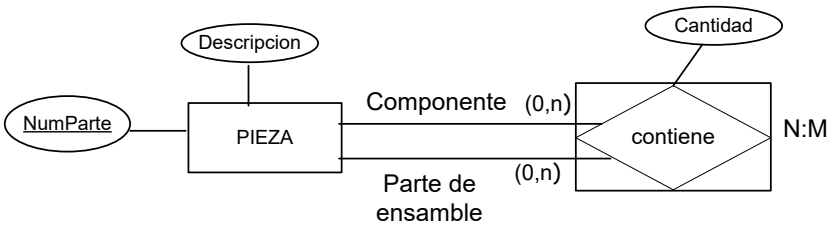
ISBN: 978-631-6557-89-6

Como se aprecia en la tabla con datos del Ejemplo 5-12, cada uno de los empleados tienen un jefe, pero éste también puede tener un jefe, como es el caso de 0606. Así también, 0203 es jefe, pero ya no tiene un superior, por lo tanto, se podría dejar Null, esto siempre y cuando el min=0 como es el caso del ejemplo ((0,1)). Si el caso fuese (1,1), entonces se tendría que hacer que 0203 sea jefe de sí mismo (Codigo_Jefe=0203).

- **Relación Recursiva N:M**

El tratamiento para la transformación de una recursiva con cardinalidad N:M, es similar al de una transformación de una relación N:M. Se obtiene una tabla para la entidad y otra tabla que corresponde a la relación.

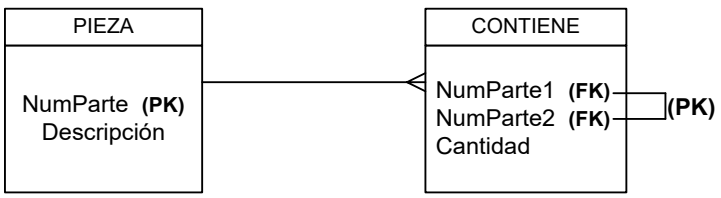
Ejemplo 5-13. Modelo Extendido: Transformación de una relación recursiva N:M



Transformación
Relación recursiva N:M

PIEZA (NumParte, Descripción)

CONTIENE (NumParte1, NumParte2, Cantidad)



PIEZA

NumParte	Descripción
PZ1	Llanta
PZ2	Maza
PZ3	Perno
PZ4	Rueda
PZ5	Suspensión delantera

ISBN: 978-631-6557-89-6

PZ6	Cojinete de bolas

CONTIENE

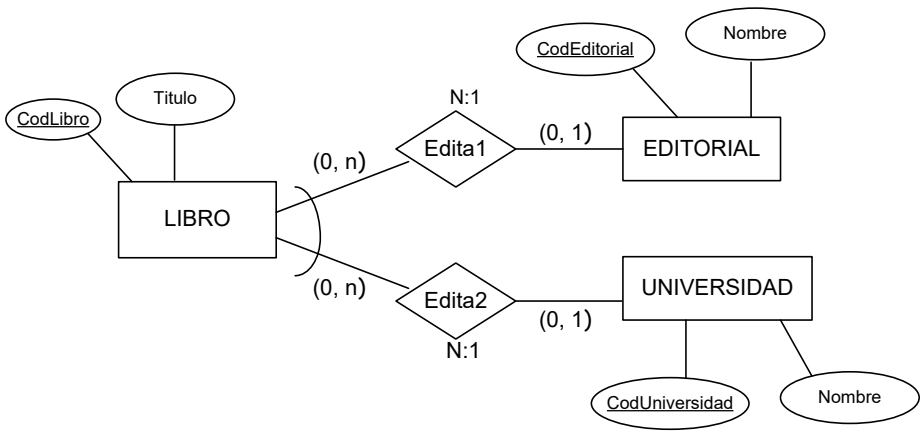
NumParte1	NumParte2	Cantidad
PZ4	PZ1	1
PZ4	PZ2	1
PZ4	PZ6	8
PZ5	PZ3	1
PZ5	PZ4	2

Como se aprecia en la tabla con datos del Ejemplo 5-13, de acuerdo con los mínimos, en la tabla CONTIENE estarán las piezas que tienen componentes

5.1.10 Transformación de una Relación Exclusiva

En la transformación de una relación exclusiva se aplica los criterios y métodos de transformación antes indicados. A continuación, se presenta dos ejemplos, cada uno con diferente cardinalidad.

Ejemplo 5-14. Modelo Extendido: Transformación de una relación exclusiva con cardinalidad 1:N



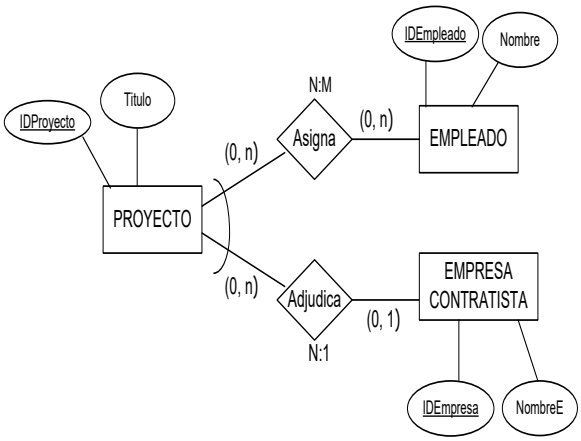
Relación Exclusiva
Transformación con propagación del atributo identificador

EDITORIAL (CodEditorial, Nombre)

UNIVERSIDAD (CodUniversidad, Nombre)

ISBN: 978-631-6557-89-6

LIBRO (CodLibro, Titulo, CodEditorial, CodUniversidad)



En este caso del Ejemplo 5-14, se propagan las claves de UNIVERSIDAD y EDITORIAL a LIBRO. Y, para que se cumpla la exclusividad, es necesario introducir restricciones en una cláusula CHECK, como se indica a continuación:

```
CHECK (
        (CodEditorial IS NULL and CodUniversidad IS NOT NULL)
        OR
        (CodEditorial IS NOT NULL and CodUniversidad IS NULL)
        )
```

EDITORIAL

CodEditorial	Nombre
E1	McGraw-Hill
E2	Grupo Planeta
E3	Crisol
E4	Prentice Hall

UNIVERSIDAD

CodUniversidad	Nombre
U1	Escuela Politécnica de Chimborazo
U2	Universidad Técnica de Ambato

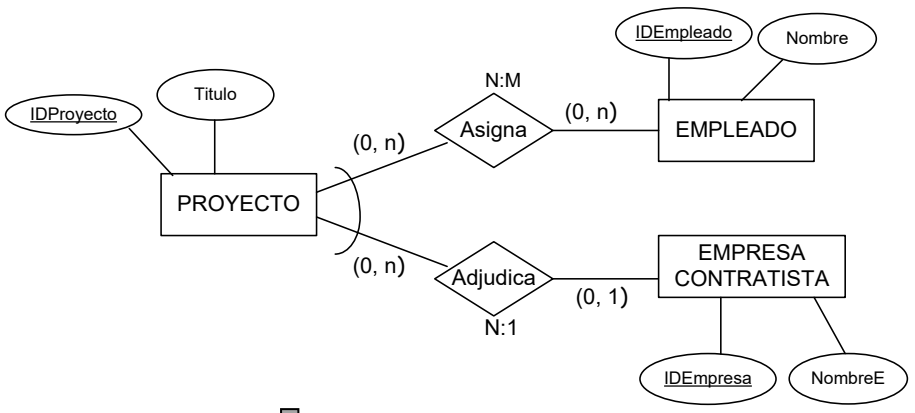
LIBRO

CodLibro	Titulo	CodEditorial	CodUniversidad
L01	Oracle Database 11g DBA Handbook	E1	
L02	Delivering Business Intelligence with Microsoft SQL Server 2005	E1	

L03	Diseño de Base de datos Relacionales		U1
L04	Grandes Civilizaciones	E2	
L05	Programación Java		U2

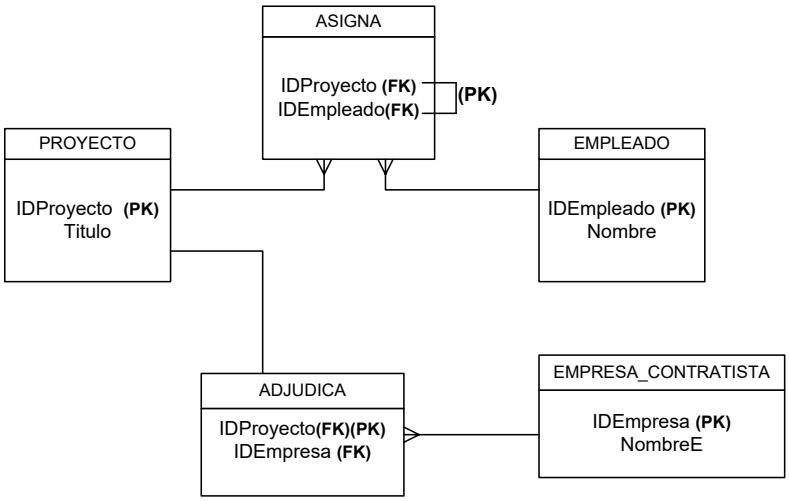
NOTA:
Con la transformación del Ejemplo 5-14 se conserva la semántica, misma que puede ser sacrificada aplicando conceptos de Normalización como se verá en la sección 5.2

Ejemplo 5-15. Modelo Extendido: Transformación de una relación exclusiva con cardinalidad N:M



EMPLEADO (IDEmpleado, Nombre)
PROYECTO (IDProyecto, Titulo)
ASIGNA (IDProyecto, IDEmpleado)
EMPRESA_CONTRATISTA (IDEmpresa, NombreE)
ADJUDICA (IDProyecto, IDEmpresa)

ISBN: 978-631-6557-89-6



En la transformación del Ejemplo 5-15, se ha creado una nueva tabla ASIGNA para la cardinalidad N:M entre PROYECTO y EMPLEADO.

Entre PROYECTO y EMPRESA_CONTRATISTA existe una cardinalidad 1:N. Sin embargo, también se ha creado la tabla ADJUDICA que corresponde a la relación. Esta decisión se lo ha tomado para evitar excesivos nulos en el caso que se aplicara propagación del identificador.

Un proyecto que se encuentra en ASIGNA no puede estar en ADJUDICA, y viceversa. Para este control, se debe utilizar disparadores (Triggers).

5.1.11 Transformación de Generalización/Especialización

El modelo relacional no permite representar tipos y subtipos de entidades. Existen 3 métodos para transformar la generalización al modelo relacional, el método o forma que se elija debe estar acorde con los objetivos que se persigan (en relación con el desempeño de la base de datos).

- **Método 1: Crear una sola tabla a partir de la entidad supertipo**

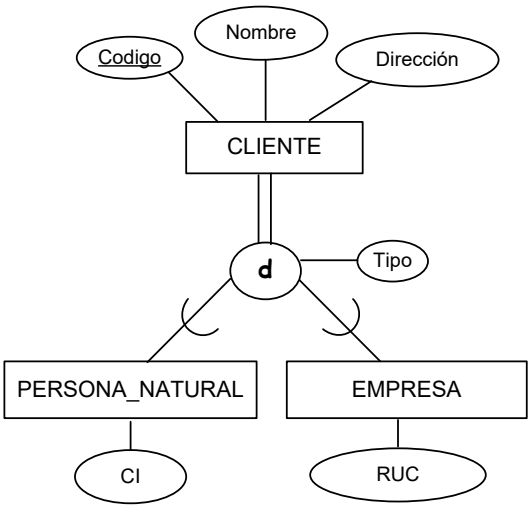
  El método 1 consiste en incluir todos los atributos de la entidad supertipo y sus subtipos en una sola tabla. Además, se añade un atributo que indica a la entidad que se hace referencia (Tipo).

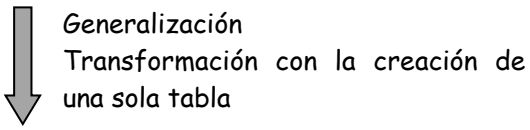
El método 1 se recomienda aplicar para los siguientes casos:

ISBN: 978-631-6557-89-6

- Los subtipos se diferencian en muy pocos atributos (preferiblemente sean compatibles: misma longitud y tipo de dato)
- Las relaciones con los que se asocia con el resto de las entidades son las mismas para todos los subtipos (o no existen). En este sentido, solo la entidad supertipo sería la que se relaciona con otra entidad.

Ejemplo 5-16. Modelo Extendido: Transformación de una Generalización - Método 1





CLIENTE (Codigo, Nombre, Direccion, CI, RUC)

CLIENTE
Codigo **(PK)** Nombre Dirección CI RUC Tipo

En el Ejemplo 5-16 la cobertura es Total - Disjunta. Sin embargo, en la transformación según las restricciones de cobertura que se presente en la jerarquía se debe tener las siguientes consideraciones:

ISBN: 978-631-6557-89-6

Solapada: La columna procedente del atributo discriminante (Tipo) puede tomar varios valores combinados

Disjunta: En función al valor de la columna Tipo (P: Personal natural o E: Empresa), validar que solo las columnas correspondientes tengan valores.

Total: La columna Tipo no puede ser Null

Parcial: La columna Tipo puede aceptar nulos.

- **¿Cómo lograr que las restricciones de cobertura se cumplan?**

En el Ejemplo 5-16, para lograr que las restricciones de cobertura se cumplan se realiza una validación de exclusividad mediante un CHECK o un Disparador; los cuales, **permiten conservar la semántica**.

```
CHECK ((Tipo = 'P' and CI IS NOT NULL and RUC IS NULL)
        or
(Tipo = 'E' and CI IS NULL and RUC IS NOT NULL))
```

NOTA:
- Según la teoría el método 1 de transformación es aplicable a todos los casos, con todas las coberturas (disjunta/solapada, total/parcial).

- La tabla CLIENTE del Ejemplo 5-16, requiere refinarse para que cumpla con reglas de Normalización.

- **Método 2: Crear tablas para entidades supertipo y para las entidades subtipos**

El método 2 de transformación es el más común, y consiste en crear una tabla para la entidad supertipo y tantas tablas como subtipos existan. Además, se añade un atributo que indica a la entidad que se hace referencia (Tipo).

Las entidades subtipos transformadas en tablas, heredan el identificador de la supertipo como clave primaria (PK).

La referencia entre las tablas resultantes es de 1 – 1, es decir un padre con un solo hijo. La tabla padre es la obtenida de la entidad supertipo y las tablas hijas corresponden a las subtipos.

ISBN: 978-631-6557-89-6

El método 2 se recomienda aplicar para los siguientes casos:

- Si se utilizan atributos diferentes para describir cada subtipo, y/o relaciones diferentes
- Mantener la semántica en el modelo relacional

El método 2 tiene ventajas y desventajas, tales como las que se indican a continuación:

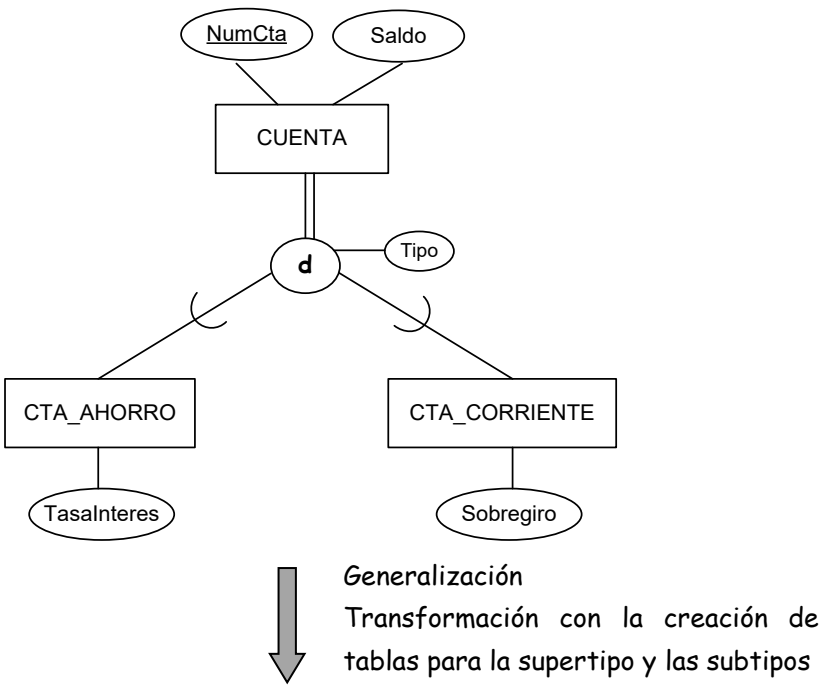
Ventajas

- Mantiene la semántica haciéndola más flexible ante posibles cambios de requerimientos.

Desventajas

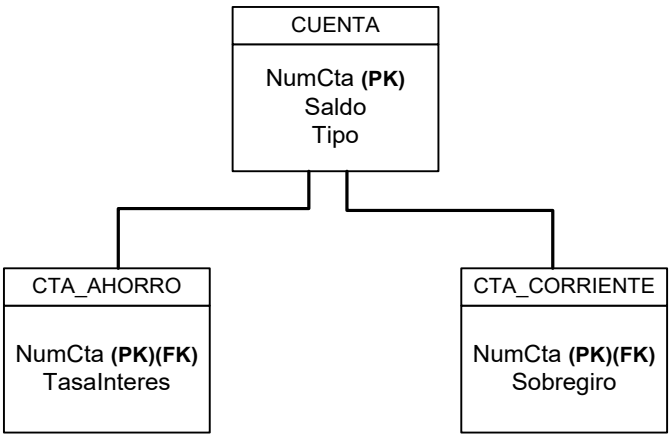
- El esquema resultante que se obtiene es bastante complejo, por ejemplo: por cada nueva ocurrencia en una subtipo, se debe también añadir en la supertipo.
- Existe una redundancia inherente (ocurrencias que se encuentran en la supertipo y también en las subtipos).

Ejemplo 5-17. Modelo Extendido: Transformación de una Generalización – Método 2



ISBN: 978-631-6557-89-6

CTA_AHORRO (NumCta, TasaInteres)
CTA_CORRIENTE(NumCta, Sobregiro)



CUENTA

NumCta	Saldo	Tipo
0001	1500	CT
0002	650	AH
0003	342	AH
0004	3879	CT
0005	768	AH
0006	2786	CT

CTA_AHORRO

NumCta	TasaInteres
0002	3
0003	2
0005	2

CTA_CORRIENTE

NumCta	Sobregiro
0001	100
0004	50
0006	150

En la transformación del método 2, las restricciones de cobertura que se presente en la jerarquía se deben controlar de las formas que a continuación se indican:

Solapada: La columna procedente del atributo discriminante (Tipo) puede tomar varios valores combinados

Disjunta: Verificar que solo se ingrese en la tabla subtipo correspondiente

ISBN: 978-631-6557-89-6

Total: La columna Tipo no puede ser Null, y se debe verificar que todas las filas de la tabla Padre (Supertipo) se encuentren en las tablas hijas (Subtipos).

Parcial: La columna Tipo puede aceptar nulos.

Las restricciones semánticas se implementarán a través de CHECK y Disparadores

- **Método 3: Crea la tabla solo para entidad subtipo**

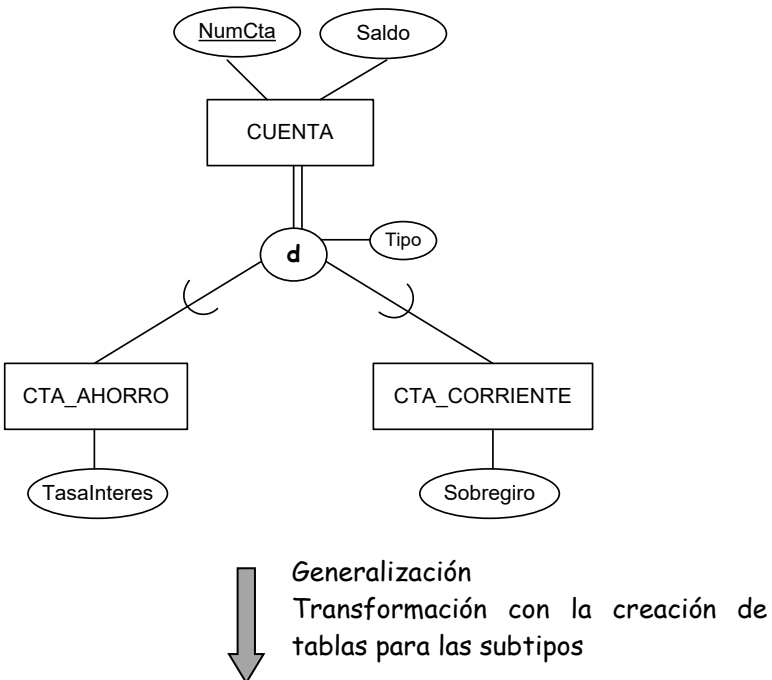
  Se crea una tabla solo para cada entidad subtipo, no para la supertipo. Los atributos de la supertipo se propagan (atributos heredados) a las tablas - subtipos. No existe referencia entre las tablas resultantes, es decir que son independientes.

El método 3 se recomienda aplicar para los siguientes casos:

- Se utilizan atributos diferentes para describir cada subtipo, y/o relaciones diferentes
- El número de atributos de la entidad supertipo (comunes a todas las subtipos) no sean demasiados.

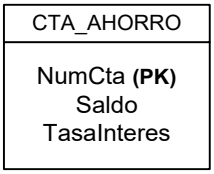
La principal desventaja es que se pierde la semántica. Además, se debe tener cuidado que si el número de atributos de la supertipo es excesivo, su duplicación en cada subtipo no se justifica.

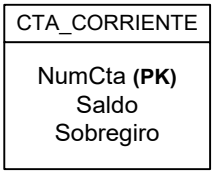
Ejemplo 5-18. Modelo Extendido: Transformación de una Generalización – Método 3



ISBN: 978-631-6557-89-6

CTA_AHORRO (NumCta, Saldo, TasaInteres)
CTA_CORRIENTE(NumCta, Saldo, Sobregiro)





CTA_AHORRO

NumCta	Saldo	TasaInteres
0002	650	3
0003	342	2
0005	768	2

CTA_CORRIENTE

NumCta	Saldo	Sobregiro
0001	1500	100
0004	3879	50
0006	2786	150

El método 3 de transformación no es práctico para generalizaciones solapadas o parciales, sólo lo es para totales y disjuntas. Las restricciones semánticas se implementarán a través de CHECK y Disparadores

Disjunta: Verificar que solo se ingrese en la tabla subtipo correspondiente

Total: Nada que controlar

5.2 Normalización

La normalización es un proceso formal que permite organizar la estructura de una base de datos con la finalidad de reducir la redundancia y optimizar la integridad de los datos. Para entender de mejor manera qué es la normalización se presentan algunas definiciones tomadas de la bibliografía:

> *Normalización* es un enfoque formal que examina datos, y grupos de datos de una manera que mejor pueda acomodar futuros cambios en el negocio y de minimizar el impacto de estos cambios en sistemas de aplicación.
>
> *Normalización* es un conjunto de reglas formales en un modelo de datos relacional llamadas formas normales

ISBN: 978-631-6557-89-6

(Normal Form – FN). Estas reglas se deben revisar cuando se ha creado el modelo lógico relacional, es decir, las tablas.

La *Normalización* es una técnica que se utiliza para comprobar la validez de los esquemas lógicos basados en el modelo relacional, ya que asegura que las relaciones (tablas) obtenidas no tienen datos redundantes.

Tanto el diseño conceptual, como el diseño lógico, son procesos iterativos, tienen un punto de inicio y se van refinando continuamente. Pero, si el esquema lógico obtenido no cumple con los requerimientos del usuario (mundo real) y/o tienen problemas de normalización (exigencia del modelo relacional), será difícil, sino imposible, mantener la integridad de la base de datos; así como también, puede ser difícil definir la implementación física o el mantener unas prestaciones aceptables del sistema.

En las bases de datos que no han sido normalizadas es común la aparición de diferentes anomalías que afectan la integridad y consistencia de la información. Una de ellas es la anomalía de inserción, que impide registrar nuevos datos cuando no se dispone de información adicional considerada obligatoria. También se tiene la anomalía de actualización, la cual obliga a modificar un mismo dato en múltiples registros. Además, está la anomalía de eliminación, que ocurre cuando al borrar un registro se pierde información relevante relacionada.

La teoría de normalización es una ayuda que proporciona un procedimiento riguroso para el diseño de bases de datos relacionales. Esta teoría se centra en que los esquemas relacionales cumplan con las formas normales. La Formas Normales son reglas que están encaminadas a eliminar redundancias e inconsistencias de dependencia en el diseño de las tablas.

Para determinar la FORMA NORMAL que tiene una relación, previamente se debe definir las DEPENDENCIAS DE LOS DATOS (limitantes) de dicha relación. Existen dependencias, como la funcional y la dependencia de valores múltiples; las cuales, conducen a varias formas normales; como las que se indican a continuación:

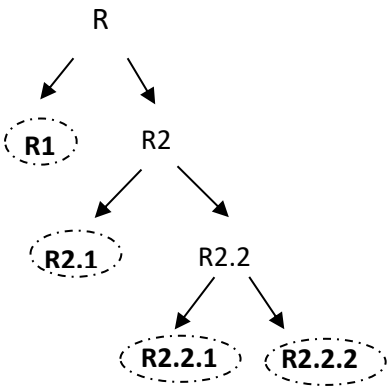
ISBN: 978-631-6557-89-6

- ✓ **Dependencias funcionales** ⟶
  - Primera Forma Normal (1FN)
  - Segunda Forma Normal (2FN)
  - Tercera Forma Normal (3FN)
  - Forma Normal Boyce Codd (BCNF: Boyce Codd Normal Form)
- ✓ **Dependencias de valores múltiples** ⟶⟶
  - Cuarta Forma Normal (4FN)

5.2.1 Proceso de Normalización

El proceso de normalización implica una descomposición sin pérdida de una relación universal inicial (agrupa todos los atributos de la base de datos por diseñar) en varias relaciones más pequeñas. Se efectúan descomposiciones sucesivas de las relaciones intermedias hasta que cumpla con una **FORMA NORMAL** (Normal Form – FN) **adecuada** como se ilustra en la **¡Error! No se encuentra el origen de la referencia.¡Error! No se encuentra el origen de la referencia.**.

Figura 5-1. Descomposición de una relación



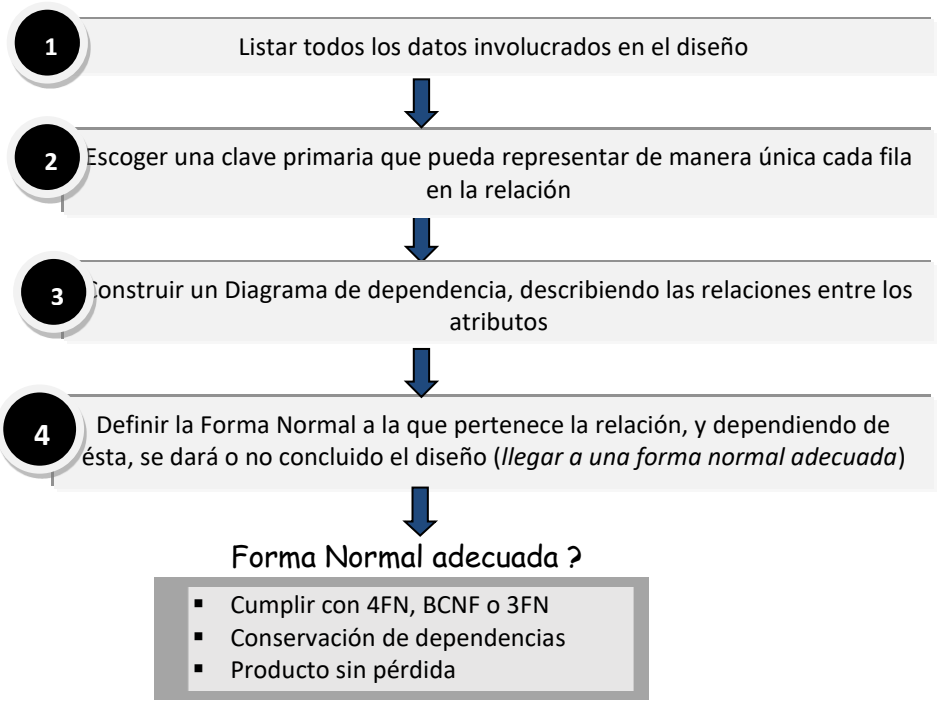
Elaborado por los Autores

El proceso de normalización se desarrolla mediante una serie de pasos que permiten estructurar los datos de manera eficiente y consistente. Primero se identifican y listan todos los datos que intervendrán en el diseño, después se selecciona una clave primaria capaz de identificar de manera única cada registro, luego se establecen las dependencias

ISBN: 978-631-6557-89-6

funcionales entre los atributos para comprender cómo se relacionan entre sí y, finalmente, se determina la forma normal alcanzada con el propósito de evaluar si la estructura es adecuada o requiere continuar avanzando hacia un nivel superior de normalización; ver Figura 5-2.

Figura 5-2. Pasos del Proceso de Normalización



Elaborado por los Autores

5.2.2 Normalización por medio de Dependencias Funcionales (DF)

Una dependencia funcional (DF) es una restricción o limitante sobre un conjunto de tuplas (filas) que pueden aparecer en una relación R (Mendelzon, 2000).

Sea R un esquema de relación con los atributos (columnas) A y B

R (A, B)

Se dice que A dependen funcionalmente de B, si cada valor de A tiene asociado un solo valor de B. Es decir, que A está determinado por el valor B; por tal razón a B se le denomina determinante de A, y se simboliza:

$$B \longrightarrow A$$

ISBN: 978-631-6557-89-6

Ejemplo 5-19. Representación de una dependencia funcional (DF)

COD_PRO	NOMB_PRO	GENERO_PRO
P1	Lorena Saltos	Femenino
P2	Ana Vega	Femenino
P3	Fausto Galeas	Masculino
P4	Rita Flores	Femenino
....	...	...
Pn	...	...

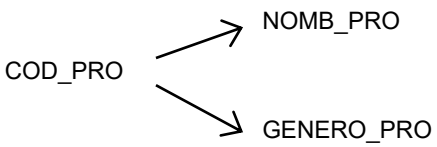
Considere la relación PROFESOR, que consta de tres atributos: Código del profesor (COD_PRO), nombre del profesor (NOMB_PRO) y género del profesor (GENERO_PRO).

PROFESOR (COD_PRO, NOMB_PRO, GENERO_PRO)

Para un valor dado de COD_PRO, digamos P1, sólo puede haber un nombre (Lorena Saltos). Así, el atributo NOMB_PRO es funcionalmente dependiente del atributo COD_PRO.

De igual manera, el atributo GENERO_PRO es funcionalmente dependiente del atributo COD_PRO porque para un valor dado de COD_PRO puede haber sólo un GENERO_PRO. Sin embargo, no se puede decir lo contrario; que COD_PRO es funcionalmente dependiente de GENERO_PRO, porque más de un COD_PRO (P1, P2, P4) se puede asociar con un GENERO_PRO (Femenino).

Los atributos NOMB_PRO y GENERO_PRO de la relación PROFESOR son funcionalmente dependientes de su clave principal, COD_PRO:



Se dice **COD_PRO → NOMB_PRO, GENERO_PRO**

COD_PRO es determinante de Nombre del profesor (NOMB_PRO) y también lo es de Género (GENERO_PRO)

O

NOMB_PRO y GENERO_PRO dependen funcionalmente de COD_PRO (este atributo es su llave de acceso)

ISBN: 978-631-6557-89-6

NOTA:
Para mejor comprensión, también se puede dar la siguiente interpretación:
Un profesor (representado por COD_PRO) tiene un solo nombre (NOMB_PRO) y un solo género (GENERO_PRO), eso simboliza la flecha simple ⟶

Las dependencias funcionales son muy importantes porque conducen a varias formas normales de base de datos muy valiosas, como son:

5.2.2.1 Primera Forma Normal (1FN)

Una relación está en la primera forma normal (1FN) si todos los campos en cada fila contienen un solo valor tomado de sus dominios respectivos.

El dominio de un atributo es el rango de valores continuos o discretos permitidos para el atributo (o columna). Del Ejemplo 5-20, si los valores del atributo CANTIDAD son enteros entre 1 y 99, el dominio es el conjunto de enteros 1, 2, 3, ... 99.

Ejemplo 5-20. Normalización con dependencias funcionales

Una empresa necesita un subsistema computarizado que le permita manejar y controlar sus CUENTAS POR COBRAR.

Partimos de una relación universal inicial CLIENTE_COBROS que contienen todos los datos relacionados a los cobros (CUENTAS POR COBRAR) que la empresa tiene que realizar a sus clientes.

Datos:

Código del cliente	COD_CLI
Nombre del cliente	NOMB_CLI
Ciudad del cliente	CIU_CLI
Precio unitario del artículo	PRE_UNI
Retribución (ganancia) por artículo	GAN_ENT
Artículo de venta o entrega	ARTICULO
Cantidad de artículos de entrega	CANTIDAD

ISBN: 978-631-6557-89-6

Fecha de entrega	FEC_ENT

(a) CLIENTE_COBROS

COD_CLI	NOMB_CLI	CIU_CLI	GAN_ENT	PRE_UNI	ARTICULO	CANTIDAD	FEC_ENT
C1	JUAN	QUITO	0.75	8.20	TUERCAS	1	05/05/93
C2	JANET	AMBATO	1.95	4.00 8.20 2.00	RODELAS TUERCAS CLAVOS	2 1 3	10/12/93
C3	BOBY	QUITO	0.75	4.00 2.00	RODELAS CLAVOS	1 2	08/10/93 05/05/93
C4	RITA	GUAYAQUIL	1.05	10.50	TORNILLOS	1	10/10/93

La tabla con los datos está sin normalizar, o dicho en otras palabras la relación o tabla (a) CLIENTE_COBROS es un archivo no plano, por lo tanto, es una relación (tabla) sin normalizar.

La tabla CLIENTE_COBROS muestra una manera típica y con sentido común de guardar los datos de los clientes en forma tabular y sin datos duplicados (C2 y C3). Pero, para que una base de datos funcione uniformemente, se deben crear nuevas filas para que reemplacen los espacios en blanco, como muestra la siguiente tabla (b) CLIENTE_COBROS rectificada.

Esta última relación (b) CLIENTE_COBROS **está en 1FN** porque cada atributo (o campo) de la fila contiene un valor, sin embargo, existe repetición de los datos por lo que puede presentarse anomalías en las operaciones de inserción, eliminación o actualización, lo que nos obliga a seguir normalizando y cumplir otras formas normales.

(b) CLIENTE_COBROS

COD_CLI	NOMB_CLI	CIU_CLI	GAN_ENT	PRE_UNI	ARTICULO	CANTIDAD	FEC_ENT
C1	JUAN	QUITO	0.75	8.20	TUERCAS	1	05/05/93

ISBN: 978-631-6557-89-6

C2	JANET	AMBATO	1.95	4.00	RODELAS	2	10/12/93
C2	JANET	AMBATO	1.95	8.20	TUERCAS	1	10/12/93
C2	JANET	AMBATO	1.95	2.00	CLAVOS	3	10/12/93
C3	BOBY	QUITO	0.75	4.00	RODELAS	1	08/10/93
C3	BOBY	QUITO	0.75	2.00	CLAVOS	2	05/05/93
C4	RITA	GUAYAQUIL	1.05	10.50	TORNILLOS	1	10/10/93

- *Anomalías de inserción*
  - No se puede introducir un nuevo artículo sin que se haya vendido al menos a un cliente.
  - Análogamente, no se puede introducir información sobre un cliente nuevo hasta que el cliente haya comprado un artículo.
- *Anomalías en la eliminación*
  - Si un artículo se hace obsoleto y se decide no venderlo, se debe eliminar los registros (o filas) que contengan dicho artículo; pero el borrado de las filas no sólo quitaría la información del inventario acerca del artículo sino también acerca del cliente. Del mismo modo, el eliminar la información del cliente provoca la eliminación de la información del artículo.
- *Anomalías de actualización*
  - La duplicación de datos almacenados causa problemas de actualización. Por ejemplo, el cliente JANET se muda de AMBATO a QUITO, se tendrá que modificar cada registro que contenga JANET; por lo tanto, se registrarán datos inconsistentes.

Para evitar las anomalías descritas, conviene descomponer la relación universal en relaciones más pequeñas, es decir, NORMALIZAR la relación (b) CLIENTE_COBROS.

- **Normalizar:**

2 Escoger una clave primaria que pueda representar de manera única a cada fila de la tabla o relación

ISBN: 978-631-6557-89-6

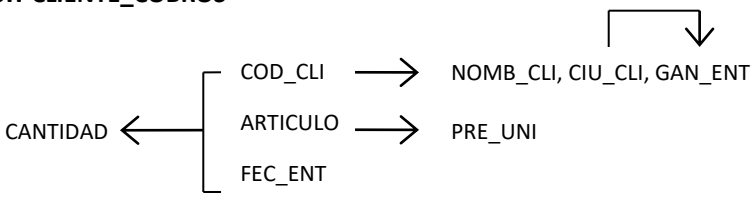
CLIENTE_COBROS (**COD_CLI, ARTICULO,FEC_ENT**, NOMB_CLI, CIU_CLI, GANT_ENT, PRE_UNI, CANTIDAD)

La clave primaria de una relación puede ser una clave compuesta (concatenada, múltiple) que consta de más de un atributo. Por lo tanto, un atributo puede ser funcionalmente dependiente de un grupo de atributos en vez de un solo atributo.

- Un atributo que forma parte de la clave compuesta se llama **atributo principal.**
- Cualquier atributo que no forma parte de la clave principal se llama **no-clave.**
- El atributo clave primaria no puede contener un valor nulo o sin definir.

3 Construir un diagrama de dependencia funcional describiendo las relaciones entre los atributos.

Relación CLIENTE_COBROS



4 Definir la Forma Normal a la que pertenece la relación y dependiendo de éste se dará o no concluido el diseño.

NOTA:
Hasta este punto la relación ya está en 1FN, pero presenta otros problemas que se expresan en las anomalías indicadas. Así que, al llegar al paso 4 es evidente que no se ha llegado a una forma normal adecuada, por lo tanto, hay que seguir descomponiendo.

5.2.2.2 Segunda Forma Normal (2FN)

Una relación es o pertenece a la segunda forma normal (2FN) si es 1FN y cada atributo no-clave de la relación es total y funcionalmente dependiente de su clave principal.

Continuando con el Ejemplo 5-20, la relación CLIENTE_COBROS NO cumple con la 2FN por lo que se requiere descomponer en las relaciones CLIENTE, INVENTARIO y COBROS. De

Relación CLIENTE_COBROS

ISBN: 978-631-6557-89-6

esta manera, las relaciones (Tablas) más pequeñas obtenidas si cumple con esta forma normal.

Las relaciones CLIENTE, INVENTARIO y COBROS cumplen la 2FN. El contenido de estas tres relaciones derivadas se muestra en las siguientes tablas con datos:

Relación CLIENTE

COD_CLI	NOMB_CLI	CIU_CLI	GAN_ENT
C1	JUAN	QUITO	0.75
C2	JANET	AMBATO	1.95
C3	BOBY	QUITO	0.75
C4	RITA	GUAYAQUIL	1.05

Relación INVENTARIO

ARTICULO	PRE_UNI
TUERCAS	8.20
RODELAS	4.00
CLAVOS	2.00
TORNILLOS	10.50

Relación COBROS

COD_CLI	**ARTICULO**	**FEC_ENT**	CANTIDAD
C1	TUERCAS	05/05/93	1
C2	RODELAS	10/12/93	2
C2	TUERCAS	10/12/93	1
C2	CLAVOS	10/12/93	3
C3	RODELAS	08/10/93	1
C3	CLAVOS	05/05/93	2
C4	TORNILLOS	10/10/93	1

Como se aprecia en las tablas con datos, CLIENTE, INVENTARIO y COBROS cumplen con la 2FN, pero la relación CLIENTE sigue presentando las siguientes anomalías de almacenamiento:

- *Anomalías de inserción*
  - La ganancia de entrega GAN_ENT para una ciudad en particular no se puede registrar en la relación CLIENTE hasta que haya al menos un cliente que resida en esa ciudad. De otra manera la clave principal COD_CLI quedará sin definir. Esta anomalía se debe a la dependencia funcional del atributo no clave GAN_ENT de otro atributo no clave CIU_CLI.

- *Anomalías en la eliminación*
  - Si se desea cerrar la cuenta del cliente RITA y se elimina el registro correspondiente de la relación CLIENTE, esto provocaría que se pierda la información referente a GAN_ENT de Guayaquil.

ISBN: 978-631-6557-89-6

- *Anomalías de actualización*
  - Si se desea cambiar el precio de ganancia de entrega GAN_ENT para Quito, entonces se debe actualizar GAN_ENT en todos los registros de la ciudad Quito.

NOTA:

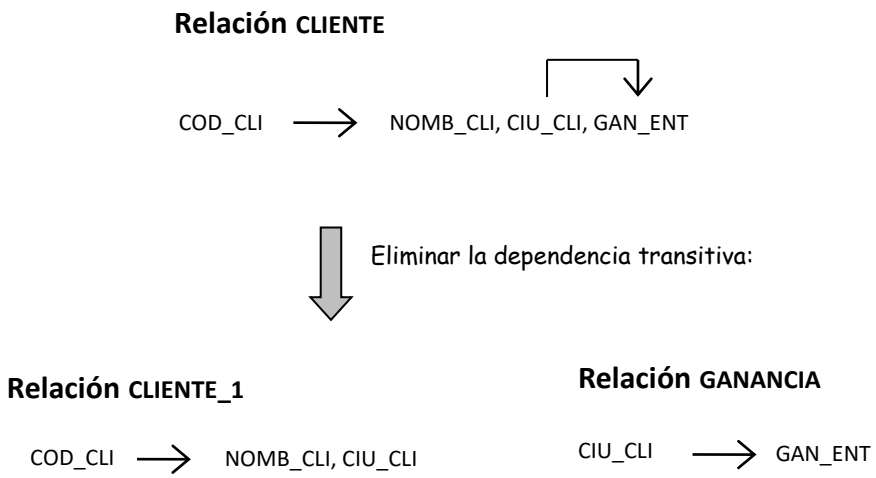
Hasta este punto las tres relaciones están en 2FN, pero una de ellas (CLIENTE) sigue presentando otros problemas que se expresan en las anomalías indicadas. Por lo tanto, hay que seguir descomponiendo

5.2.2.3 Tercera Forma Normal (3FN)

Una relación pertenece a la tercera forma normal (3FN) si es 2FN y ningún atributo no-clave es dependiente transitivamente de algún otro atributo no-clave (candidata).

Del Ejemplo 5-20, CLIENTE no es 3FN, ya que el atributo no-clave GAN_ENT es funcionalmente dependiente del atributo no-clave CIU_CLI. Sin embargo, CIU_CLI a su vez es dependiente de la clave principal COD_CLI. Mientras que las otras relaciones INVENTARIO y COBROS, son 2FN y dan cumplimiento también a regla 3FN, por lo tanto, terminan como relaciones 3FN.

Una de las soluciones al problema es eliminar la dependencia transitiva (una no-clave depende transitivamente de la clave primaria si es funcionalmente dependiente de otra no clave) de la relación CLIENTE almacenando las no-claves del problema en una nueva relación. Como se indica a continuación:



ISBN: 978-631-6557-89-6

Las relaciones CLIENTE_1 y GANANCIA son 3FN. El contenido de estas relaciones derivadas se muestra en las siguientes tablas con datos:

Relación CLIENTE_1

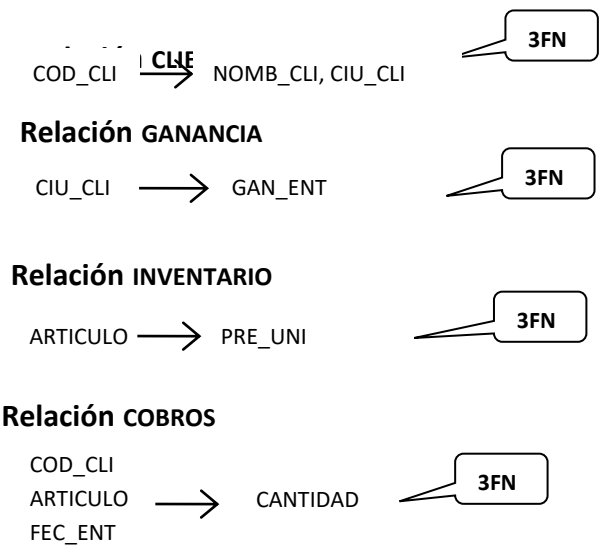
COD_CLI	NOMB_CLI	CIU_CLI
C1	JUAN	QUITO
C2	JANET	AMBATO
C3	BOBY	QUITO
C4	RITA	GUAYAQUIL

Relación GANANCIA

CIU_CLI	GAN_ENT
QUITO	0.75
AMBATO	1.95
GUAYAQUIL	1.05

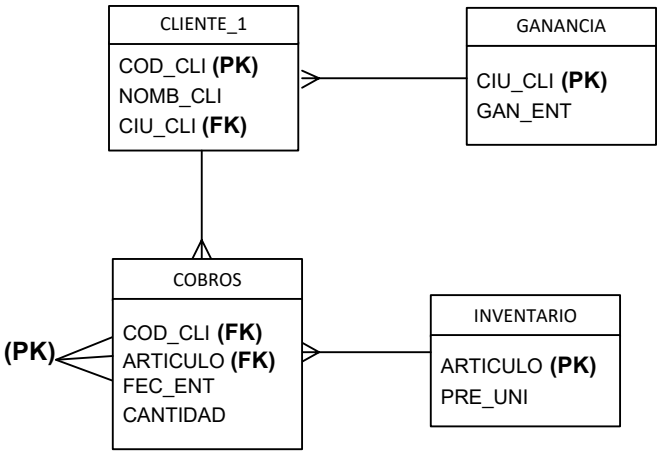
- **Resultado de la normalización con dependencias funcionales**

Al aplicar la Normalización en el Ejemplo 5-20, las relaciones resultantes están en 3FN, siendo esta una forma normal adecuada, dándose por concluida la descomposición.



El esquema relacional gráfico equivalente, sería:

ISBN: 978-631-6557-89-6



5.2.2.4 Forma Normal Boyce Codd (BCNF)

En la mayoría de los casos, el proceso de normalización se considera concluido cuando todas las relaciones resultantes alcanzan la 3FN. La definición original de la 3FN fue propuesta por Edgar F. Codd en 1972; sin embargo, con el tiempo fue revisada y mejorada, dando origen a una versión más estricta conocida como Forma Normal de Boyce-Codd (BCNF).

> *Una relación es BCNF si cada determinante en la relación es una clave aspirante*.

Si existe algún atributo que resulte total y funcionalmente dependiente de otro se llama **determinante**. En la mayoría de los casos cuando una relación es 3FN también es BCNF.

Una **clave aspirante** (Candidate Key) es un atributo o un grupo de atributos cuyo contenido puede representar de manera única cada fila de la relación. Cuando en una relación hay más de una clave aspirante, una de las claves aspirantes se designa como clave primaria.

Las relaciones CLIENTE y PRODCUTO del Ejemplo 5-21 están en 3FN, porque ningún no-clave depende transitivamente de otra clave. También están en BCNF ya que cada determinante es clave aspirante y sólo uno será clave.

Ejemplo 5-21. Relaciones que son 3FN y también BCNF

ISBN: 978-631-6557-89-6

Relación CLIENTE

CLIENTE (COD_CLI, NOMB_CLI, CIU_CLI)

COD_CLI ⟶ NOMB_CLI, CIU_CLI

Relación PRODUCTO

PRODUCTO (COD_PRO, NOMB_PRO, PRE_UNI, STOCK)

COD_PRO ⟶ NOMB_PRO, PRE_UNI, STOCK

Ejemplo 5-22. Normalización con BCNF. Caso: Proyecto - Programador

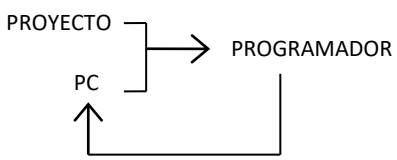
El problema consiste en un pequeño subsistema computarizado que maneje y controle los proyectos de una empresa. Se considera la relación PROY-PC-PROG, cuyo contenido se ha dado bajo las siguientes consideraciones:

- Para cada proyecto, una PC dada es usada por sólo un programador, aun cuando el programador trabaje en más de un proyecto.
- Un proyecto puede usar distintos tipos de PC
- Un programador se especializa sólo en un tipo de PC, pero una PC se puede compartir entre distintos programadores para distintos proyectos.

PROY-PC-PROG

PROYECTO	PC	PROGRAMADOR
P01	IBM PC	JANETH
P01	APPLE	BOBY
P15	DELL PORTATIL	TONY
P20	IBM PC	JANETH
P30	IBM PC	GALO

Relación PROY-PC-PROG



La relación PROY-PC-PROG está en 3FN porque ninguna no-clave es transitivamente dependientes de otras no-clave, y la no-clave es totalmente dependiente de la clave primaria. Sin embargo, la relación no es BCNF ya que el PROGRAMADOR es un determinante de PC. Como resultado, la relación sufre ciertas anomalías:

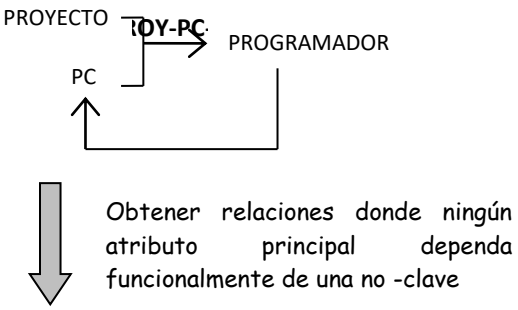
- *Anomalías de inserción*

ISBN: 978-631-6557-89-6

- Si se contrata un nuevo programador para trabajar con el APPLE, no es posible insertar el registro hasta que se dé un proyecto al programador.

- *Anomalías en la eliminación*
  - Si se desea eliminar información sobre el proyecto P15. Tal operación también eliminaría la información que TONY es especialista en DELL PORTATIL porque P15 es el único proyecto que TONY trabaja.

Las anomalías 3FN de la relación PROY-PC-PROG suceden cuando hay dos claves sobrepuestas y cuando uno de los dos atributos principales depende funcionalmente de una no-clave. El problema puede superarse proyectando la relación en dos relaciones BCNF.



Relación PROY-PC

PROGRAMADOR ⟶ PC

PROGRAMADOR	PC
JANETH	IBM PC
BOBY	APPLE
TONY	DELL
GALO	IBM PC

Relación PROY-PROG

PROYECTO ⟶ PROGRAMADOR

PROYECTO	PROGRAMADOR
P01	JANETH
P01	BOBY
P15	TONY
P20	JANETH
P30	GALO

- **Comparación de BCNF y 3FN**

- Se han visto ya dos formas normales de los esquemas de bases de datos: 3FN y BCNF. La 3FN tiene la ventaja de que se sabe que es posible obtener un diseño 3FN sin sacrificar el producto sin pérdida o la conversación de las dependencias. No obstante, la 3FN tiene una desventaja: si ni se elimina todas las dependencias transitivas, es posible que sea preciso utilizar valores vacíos para representar algunas de las relaciones posibles entre los datos.

ISBN: 978-631-6557-89-6

- Si el diseñador se ve obligado a escoger entre la BCNF o conservar las dependencias de 3FN, es posible generalmente optar por 3FN. Si no hay una forma suficiente de comprobar si se conserva las dependencias, se reducirá drásticamente el rendimiento del sistema o bien se arriesgará la integridad de la información en la base de datos. Ninguna de estas alternativas es atractiva. Cuando se presentan estas alternativas, el grado limitado de redundancia que producen las dependencias transitivas permitidas en 3FN es el menor de los males. Por lo tanto, generalmente se escoge asegurar la conservación de las dependencias y sacrificar la BCNF.

NOTA:
La relación obtenida en BCNF: PROY-PROG no respeta la Dependencia Funcional (DF), por lo que se debería representarse como un multivalor, tema que se lo estudia en la siguiente sección.

PROYECTO $\longrightarrow\!\!\!\!\!\rightarrow$ PROGRAMADOR

5.2.3 Normalización por medio de Dependencias de Valor Múltiples (DMV)

Hasta ahora, la única limitante del conjunto de relaciones que se ha permitido es la Dependencia Funcional. A continuación, se definirá otro tipo de limitante, llamado Dependencia de Valores Múltiples. Como se hizo en el caso de las dependencias funcionales, se emplearán las de valores múltiples para definir una forma normal de los esquemas de relaciones, y ésta forma normal es conocida como cuarta forma normal (4FN).

> *Dada una relación R, el atributo A de esta relación se dice ser dependiente de multivalores (DMV) del atributo B si un rango específico de valores del atributo A está determinado por un valor particular de B. La flecha doble indica que B define varios valores múltiples de A.*

R (A, B)

B $\longrightarrow\!\!\!\!\!\rightarrow$ A

Una dependencia multivaluada expresa que dos conjuntos de atributos en un mismo esquema son mutuamente independientes (Mendelzon, 2000).

ISBN: 978-631-6557-89-6

Ejemplo 5-23. Dependencia Multivalor (DMV)

Sea R una realación con atributos A,B y C, ocurre una DMV cuando B es dependiente multivalor de A, y C tambien es dependiente multivalor de A. Se expresa de la siguiente manera:

R (A, B, C)

Tener ciudado y no confundir un DMV con DF con clave primaria compuesta.

$$A, B \longrightarrow C$$

NOTA:
En DMV y DF con clave compuesta, se presenta un varios a varios (desde el punto de vista de cardinalidad). Por ejemplo, entre los atributos A y B

Ejemplo 5-24. Normalización DMV, caso: Inventario

Considere la relación INVENTARIO que contiene especificaciones de los artículos: precio unitario (PRE_UNI), color (COLOR) y tamaño (TALLA).

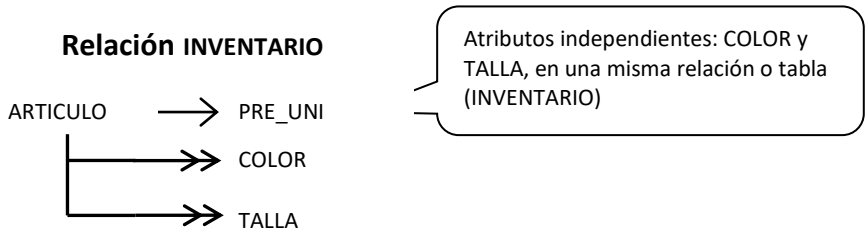
INVENTARIO (ARTICULO, PRE_UNI, COLOR, TALLA)

ARTICULO	PRE_UNI	COLOR	TALLA
A1	15	AZUL	28
A1	15	AZUL	30
A1	15	AZUL	32
A1	15	AZUL	40
A1	15	CAFE	28
A1	15	CAFE	30
A1	15	CAFE	32
A1	15	CAFE	40
A2	20	NEGRO	12
A2	20	NEGRO	16
A2	20	VERDE	12
A2	20	VERDE	16

ISBN: 978-631-6557-89-6

A2	20	BLANCO	12
A2	20	BLANCO	16

Para cada artículo del inventario (Prendas de vestir) hay una selección de distintos colores y tamaños. Los atributos COLOR y TALLA son dependientes multivalores de ARTICULO. En otras palabras, cualquier valor de ARTICULO determina un rango de valores en los atributos COLOR y TALLA. El PRE_UNI es total y funcionalmente dependiente de ARTICULO, para las filas con el mismo artículo permanece constante el precio unitario, sin importar la talla y el color. Por ejemplo, los pantalones (A2) indiferente de la talla 32 y 40 tienen el mismo precio ($15). Los pantalones hay en colores azul y café, y en las tallas: 28,30,32 y 40.



En la relación INVENTARIO existe DMV para COLOR y para TALLA, lo que indica que la relación NO esta en 4FN

5.2.3.1 Cuarta Forma Normal (4FN)

> *Una relación es 4FN si es BCNF y si no contiene dependencias multivalores*

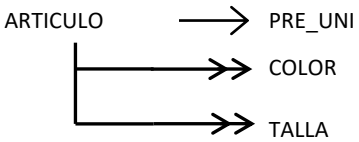
Una relación con una dependencia multivalor es aquella donde existe relaciones independientes varios a varios causando redundancia; y es esta redundancia la que es removida por la cuarta forma normal.

Dando continuidad el Ejemplo 5-24, la clave principal de la relación INVENTARIO debería ser una compuesta por: ARTICULO, COLOR, TALLA, ya que la combinación de los tres atributos es la única manera de identificar a una fila.

ISBN: 978-631-6557-89-6

Si se considera a la Relación INVENTARIO que no es 4FN, porque tiene una dependencia multivalor (COLOR y TALLA).

Relación INVENTARIO



El problema evidente con la dependencia multivalor es la *redundancia de datos*, como la tabla de datos INVENTARIO. La redundancia de los datos causada por la dependencia multivalor se puede eliminar siguiendo uno de los dos siguientes métodos:

Método A: Crear una nueva relación para cada atributo DMV

Se crea una nueva relación para el atributo DMV y su clave principal.

Este método se recomienda para los siguientes casos:

- El número de valores repetidos en un DMV es muy grande
- Se desconoce o puede variar el rango de posibles valores que puede tener el atributo DMV

En el método A, la manera en que se almacenan los distintos valores para un atributo DMV es **verticalmente**, se agregan nuevas filas.

Método B: Reemplazar un atributo DMV con atributos funcionalmente dependientes (DF)

Para cada uno de los valores de atributo DMV se puede representar por un atributo dentro de la misma fila

Este método se recomienda para los siguientes casos:

- Si la cantidad de valores distintos en un atributo DMV es un número específico y pequeño.
- En el futuro no va a variar.

En método B, la manera en que se almacenan los distintos valores para un atributo DMV es **horizontalmente**, y se asegura la conservación de dependencia.

ISBN: 978-631-6557-89-6

Relación INVENTARIO

ARTICULO → PRE_UNI
ARTICULO →→ COLOR
ARTICULO →→ TALLA

No es 4FN, necesita normalizarse

Separación de atributos no-clave que no son funcionalmente dependientes de la clave primaria.

Relación COLOR_TALLA

ARTICULO →→ TALLA
ARTICULO →→ COLOR

Eliminar DMV

Relación PRECIO

3FN

ARTICULO → PRE_UNI

Se considera, que:
- El número de tallas para un artículo es muy grande.
- Si la cantidad de colores para cualquier artículo dado, no exede de tres.

Pueden agregarse tallas nuevas en el futuro

Método A

El atributo COLOR se puede representar por COLOR1, COLOR2, COLOR3

Método B

Relación TALLA

Definitiva 4FN

ARTICULO →→ TALLA

ARTICULO	TALLA
A1	28
A1	30
A1	32
A1	40
A2	12
A2	16

En almacenamiento vertical se pierde dependencia, por lo tanto, la PK de la tabla será una compuesta:
ARTICULO + TALLA.

Relación COLOR

ARTICULO → COLOR1, COLOR2, COLOR3

Única DF

Relación PRECIO_COLOR

Definitiva 3FN

ARTICULO → PRE_UNI, COLOR1, COLOR2, COLOR3

ARTICULO	PRE_UNI	COLOR1	COLOR2	COLOR3
A1	15	AZUL	CAFE	
A2	20	NEGRO	VERDE	BLANCO

En almacenamiento horizontal se mantiene dependencia (DF), por lo tanto, la PK de la tabla será la que indica la DF: ARTICULO.

ISBN: 978-631-6557-89-6

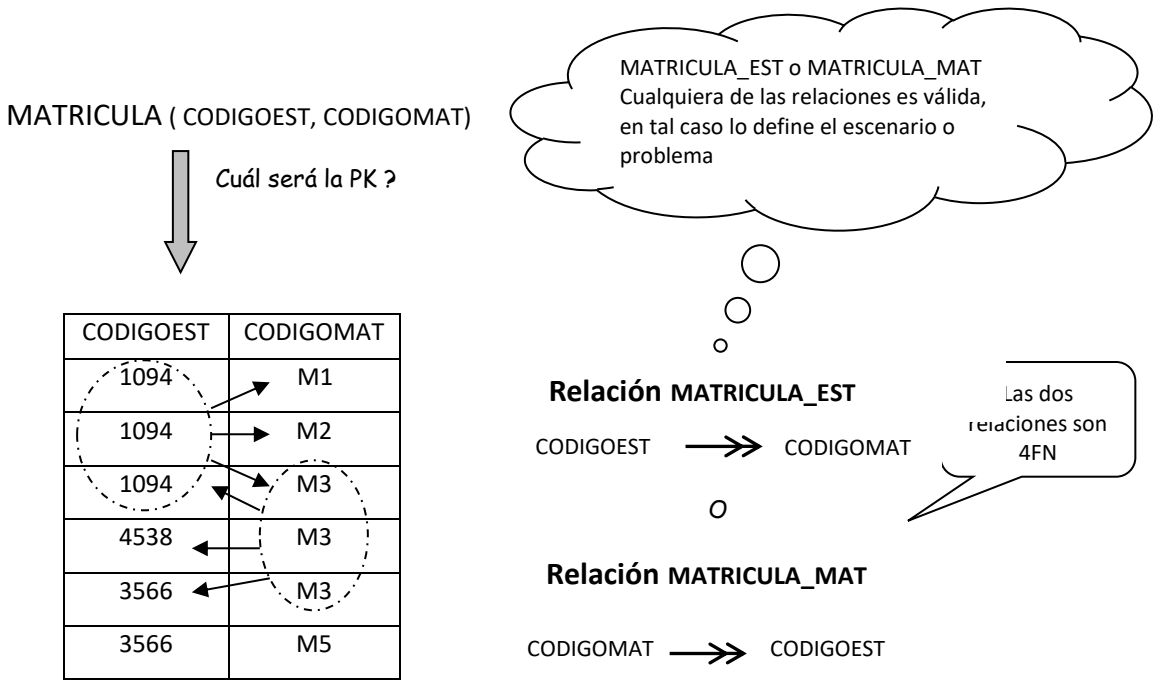
NOTA:

- Según algunos autores, fieles a la teoría de **Date**, el método B no estaría en armonía con el espíritu de 1FN, ya que provoca valores nulos; además, que la columna COLOR1, COLOR2 y COLOR3 comparten exactamente el mismo dominio y exactamente el mismo significado. El dividir en tres encabezados es artificial y puede causar problemas lógicos. Sin embargo, es una muy buena alternativa de optimización (menos tablas a menos join).
- En circunstancias normales no se requiere más normalización después de 3FN, porque ya se han tomado en cuenta los atributos DMV durante la etapa inicial de normalización para derivar en un archivo plano 1FN.

5.2.4 Clave primaria compuesta para un DF (3FN) vs DMV (4FN)

Este apartado pretende aclarar la diferencia y similitudes que existe al definir una clave primaria compuesta para un DF que llega a una 3FN, en relación una DMV que requiere normalizar a un 4FN.

Ejemplo 5-25 . Determinar la clave en una DMV



ISBN: 978-631-6557-89-6

Ejemplo 5-26 .- Clave compuesta en un DF

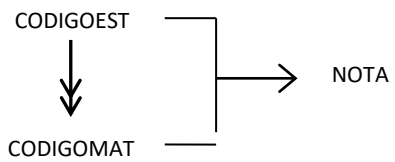
EVALUACION (CODIGOEST, CODIGOMAT, NOTA)

CODIGOEST	CODIGOMAT	NOTA
1094	M1	14
1094	M2	16
1094	M3	14
4538	M3	18
3566	M3	17
3566	M5	15

Como se aprecia en el Ejemplo 5-26, es similar al 5-25 con la diferencia que se ha añadido el atributo NOTA. De la misma forma, se presenta un DMV entre CODIGOEST y CODIGOMAT.

Cuál será la PK ?
Para conocer la NOTA, de qué datos requiero; en otras palabras la nota de qué clave depende ?

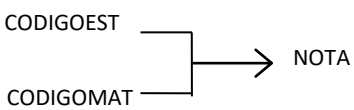
Relación EVALUACION



El atributo NOTA Depende Funcionalmente (DF) de:
CODIGOEST + CODIGOMAT

Por lo tanto, no necesita especificarse el DMV implícito que existe entre los atributos claves de la relación (es decir los que son parte de la PK)

Relación EVALUACION 3FN



ISBN: 978-631-6557-89-6

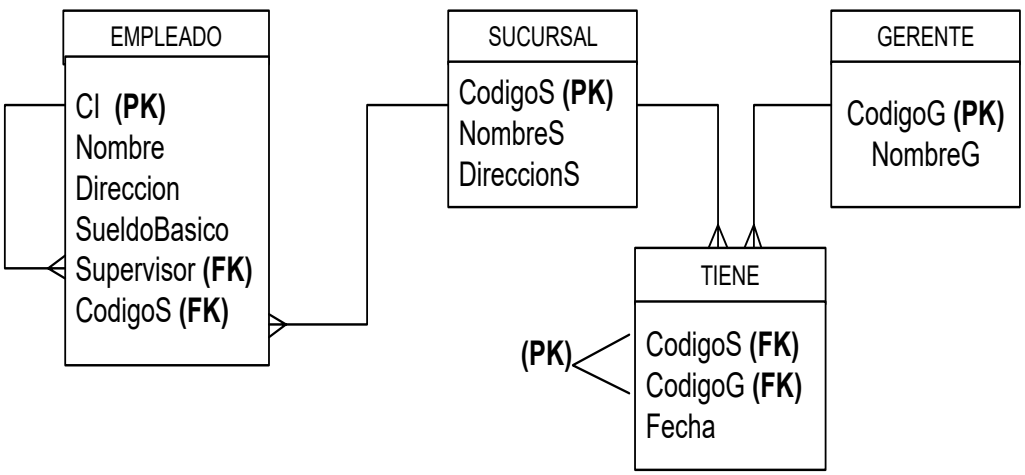
NOTA:

- La normalización es una técnica que se utiliza para comprobar la validez de los esquemas lógicos basados en el modelo relacional, ya que asegura que las relaciones (tablas) obtenidas no tienen datos redundantes.
- Al separar o descomponer una relación para eliminar anomalías de almacenamiento puede tener el impacto negativo en el desempeño de la base de datos porque se puede requerir operaciones extras de entrada y salida para recuperar datos de distintos archivos. Así, para decidir el grado de normalización deseado, el diseñador debe considerar cuidadosamente los tipos de operaciones que se efectúan con más frecuencia.
- La creación de las tablas de un sistema puede crearse a partir de un modelo conceptual, como el modelo entidad relación (mencionado en el artículo anterior) o a partir de cero. La ventaja de utilizar el modelo entidad relación es que es más sencillo generar los grupos de datos y además muchos de los problemas resueltos por la normalización, se identifican antes en el modelo conceptual. Esto en palabras más sencillas quiere decir que sería mejor iniciar generando el diagrama entidad relación del sistema, que empezar con las tablas en un DBMS. De todos modos, después de generar las tablas, es necesario hacer una revisión de las formas normales, pues el modelo entidad relación no resuelve todos los problemas que resuelve la normalización, debido a que la normalización es parte del diseño lógico relacional de la base de datos.

5.3 Ejercicios: Diseño Lógico

1) El DER del ejercicio 1) de Diseño Conceptual, transforme al esquema lógico relacional correspondiente.

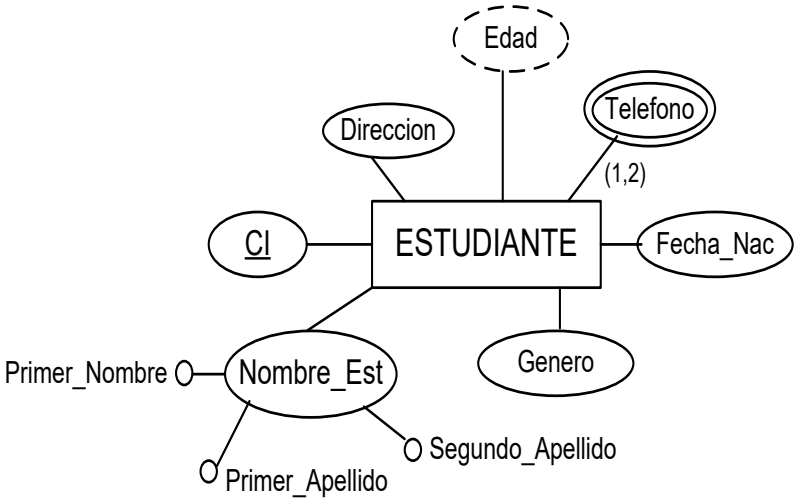
Solución:



ISBN: 978-631-6557-89-6

2) El DER del ejercicio 4) de Diseño Conceptual, transforme al esquema lógico relacional correspondiente.

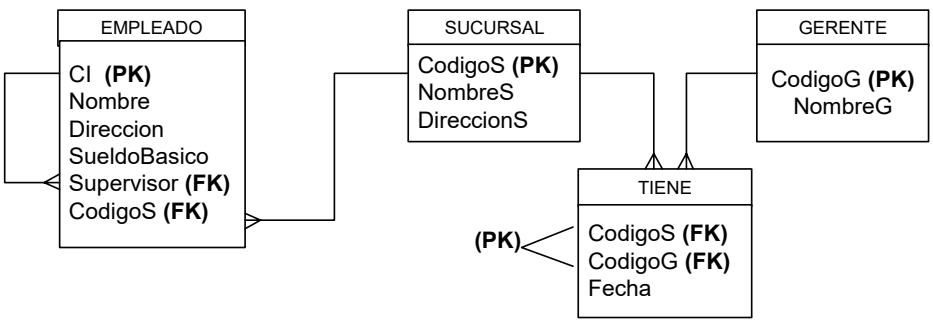
Solución:



3) El DER del ejercicio 6) de Diseño Conceptual, transforme al esquema lógico relacional correspondiente, e indique el diccionario de datos.

Solución:

Se ha considerado que TotalEmpleados sea calculado y no se almacene en la base de datos



ISBN: 978-631-6557-89-6

Diccionario de datos:

TABLA	COLUMNA	TIPO - TAMAÑO	VALIDACIÓN
SUCURSAL	CodigoS	Char(5)	Not null
	NombreS	Varchar(80)	Not null
PK : CodigoS	DireccionS	Varchar(80)	Null

TABLA	COLUMNA	TIPO - TAMAÑO	VALIDACIÓN
GERENTE	CodigoG	Char(5)	Not null
PK : CodigoG	NombreG	Varchar(80)	Not null

TABLA	COLUMNA	TIPO - TAMAÑO	VALIDACIÓN
EMPLEADO	CI	Char(10)	Not null CHECK: CI LIKE '[0-9] [0-9] [0-9] [0-9] [0-9] [0-9] [0-9] [0-9] [0-9] [0-9]'
	Nombre	Varchar(80)	Not null
PK : CI	Direccion	Varchar(80)	Null
	SueldoBasico	Float	Not null
FK : Supervisor	Supervisor	Char(10)	Not null CHECK: CI LIKE '[0-9] [0-9] [0-9] [0-9] [0-9] [0-9] [0-9] [0-9] [0-9] [0-9]'
FK : CodigoS			
	CodigoS	Char(5)	Not null

TABLA	COLUMNA	TIPO - TAMAÑO	VALIDACIÓN
TIENE	CodigoS	Char(5)	Not null
	CodigoG	Char(5)	Not null
PK : CodigoS + CodigoG FK : CodigoS FK : CodigoG	Fecha	Datetime	Not null CHECK: Fecha <= GETDATE()

4) Si tengo las siguientes relaciones (TABLAS):

Cliente (CodigoCli, CI, Nombre, Dirección, Genero, FechaNac)

Factura (NumFac, CodigoCli, FechaFac, CodigoVen)

Vendedor (NombreVen, CodigoVen)

a) Definir claves primarias y foráneas, y realizar el esquema lógico correspondiente

b) Definir el diccionario de datos

c) Representar las tablas con datos, fielmente al esquema lógico definido

ISBN: 978-631-6557-89-6

Solución:

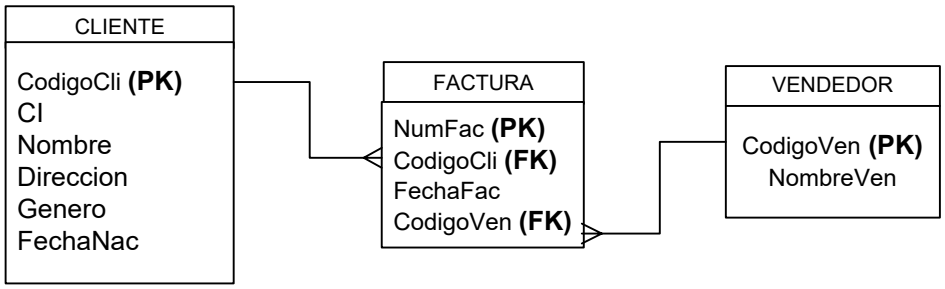


TABLA	COLUMNA	TIPO - TAMAÑO	VALIDACIÓN
CLIENTE PK : CodigoCli	CodigoCli	int	Not null
	CI	Char(10)	Not null CHECK: CI LIKE '[0-9] [0-9] [0-9] [0-9] [0-9] [0-9] [0-9] [0-9] [0-9] [0-9]'
	Nombre	Varchar(80)	Not null
	Direccion	Varchar(80)	Null
	Genero	Char(1)	Not null M: masculino / F: femenino
	FechaNac	Datetime	CHECK: FechaNac < GETDATE()

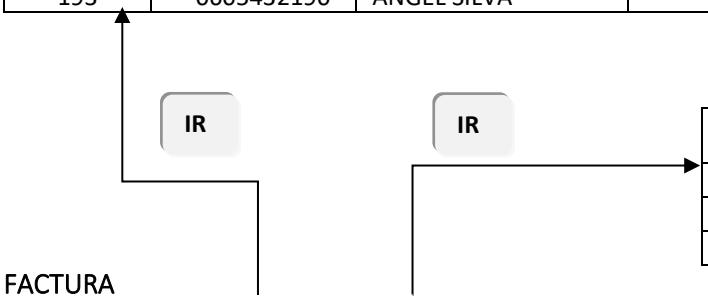
TABLA	COLUMNA	TIPO - TAMAÑO	VALIDACIÓN
VENDEDOR PK : CodigoVen	CodigoVen	int	Not null
	NombreVen	Varchar(80)	Not null

TABLA	COLUMNA	TIPO - TAMAÑO	VALIDACIÓN
FACTURA PK : NumFac FK : CodigoCli FK : CodigoVen	NumFac	int	Not null
	CodigoCli	int	Not null
	CodigoVen	int	Not null
	FechaFac	Datetime	CHECK: FechaFac <= GETDATE()

ISBN: 978-631-6557-89-6

CLIENTE

CodigoCli	CI	Nombre	Direccion	Genero	FechaNac
115	1800000091	ANGELICA RIVERA	DIR-ABC	F	24/11/1980
525	1701111987	LUIS MALAN	DIR-XYZ	M	13/09/1991
193	0605432190	ANGEL SILVA		M	09/07/1976



VENDEDOR

CodigoVen	NombreVen
1	SOFIA LAYEDRA
2	CARMEN SANTOS
3	FEDERICO SANI

FACTURA

NumFac	CodigoCli	CodigoVen	FechaFac
18	115	1	17/01/2011
200	115	1	15/02/2011
309	115	1	07/02/2011
496	115	3	18/02/2011
534	525	3	21/02/2011
551	193	2	11/03/2011
619	193	3	29/03/2011

IR: Integridad Referencial
Referencia Padre - Hijo

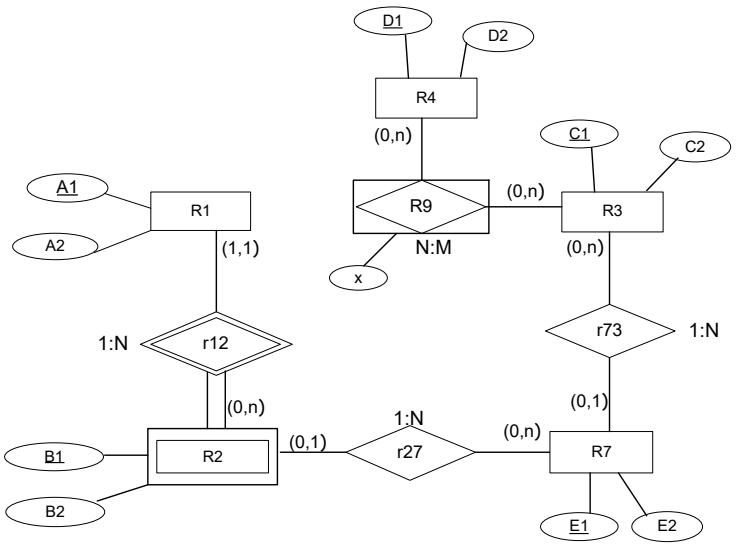
5) Ejercicio de ingeniería inversa: Dado el siguiente esquema de base de datos relacional, obtener el DER del cual se haya podido derivar.

R1(A1,A2) R2(A1,B1,B2) R3(C1,C2,E1) R4(D1,D2)

R7(A1,B1,E1,E2) R9(C1,D1,x)

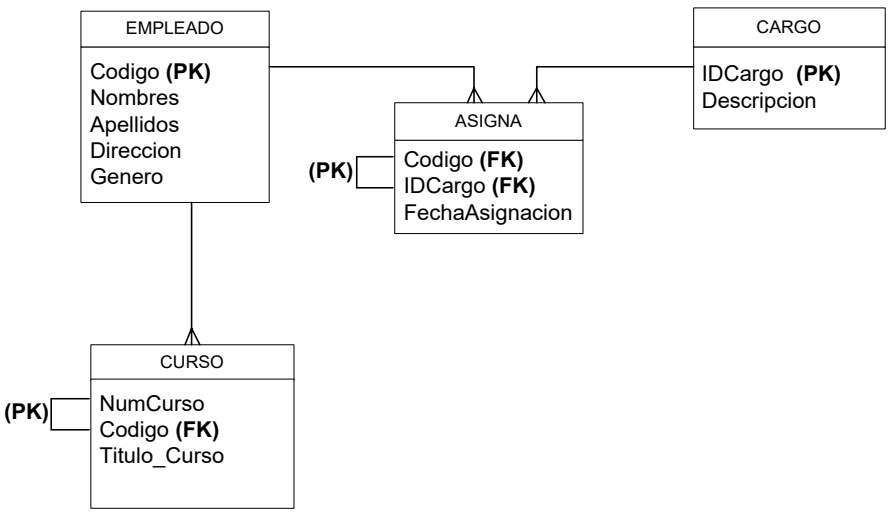
Solución:

ISBN: 978-631-6557-89-6



6) El DER del ejercicio 9) de Diseño Conceptual, transforme al esquema lógico relacional correspondiente.

Solución:



7) Con base en la tabla PRESTAMO, se desea que reconozca redundancias y diseñe una base de datos en la cual se registre los datos indicados.

a) Utilizando la Normalización, realice diseño lógico correspondiente

b) Realice el esquema lógico relacional (gráfico)

ISBN: 978-631-6557-89-6

Solución:

1 **PRESTAMO**

NUMPRESTAMO	CI_ CLIENTE	NOMB_CLIENTE	CODIGO_ LIBRO	TITULO_LIBRO	FECHA_ PRESTAMO	FECHA_ ENTREGA
P001	060205837-0	ACBDFGH	001	INTRODUCCION A LAS BASES DE DATOS	01/02/2005	04/02/2000
P001	060205837-0	ACBDFGH	005	CLIENTES/SERVIDOR	01/02/2005	04/02/2000
P002	170089768-2	JJJJJJJJJJ	006	PROCESAMIENTOS DE BASE DE DATOS	05/02/2007	06/02/2000
P003	060205837-0	ACBDFGH	008	VISUAL BASIC	07/02/2008	
P004	060987654-0	XXXXXX	006	PROCESAMIENTOS DE BASE DE DATOS	07/02/2008	15/02/2008
P004	060987654-0	XXXXXX	010	REDES	07/02/2008	15/02/2008

2 PRESTAMO (NUMPRESTAMO, CI_ CLIENTE, NOMB_CLIENTE, CODIGO_ LIBRO, TITULO_LIBRO, FECHA_ PRESTAMO, FECHA_ ENTREGA)

3 **Relación PRESTAMO**

NUMPRESTAMO → CI_ CLIENTE, NOMB_CLIENTE, FECHA_ PRESTAMO, FECHA_ ENTREGA

NUMPRESTAMO → CODIGO_ LIBRO, TITULO_LIBRO

4 Separar (Descomponer)

Relación PRESTAMO_CLIENTE — 3FN

NUMPRESTAMO → CI_ CLIENTE, FECHA_ PRESTAMO, FECHA_ ENTREGA

Relación CLIENTE — 3FN

CI_ CLIENTE → NOMB_CLIENTE

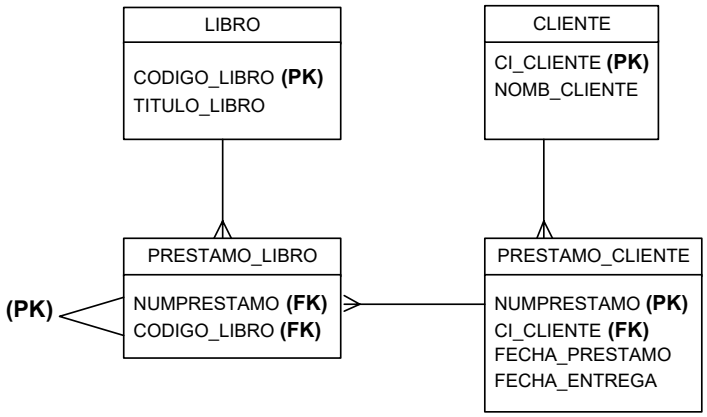
Relación LIBRO — 3FN

CODIGO_ LIBRO → TITULO_LIBRO

Relación PRESTAMO_LIBRO — 4FN

NUMPRESTAMO →→ CODIGO_LIBRO

ISBN: 978-631-6557-89-6



8) Con base en tabla ESTUDIANTE, se desea que reconozca redundancias y diseñe una base de datos en la cual se registre los datos indicados.

a) Utilizando la Normalización, realice diseño lógico correspondiente

b) Realice el esquema lógico relacional (gráfico)

c) Represente los datos en tablas (relaciones) normalizadas

ESTUDIANTE

COD_EST	NOMBRE	ESPECIALIDAD	ACTIVIDAD
100	ANA	MUSICA	NATACIÓN
100	ANA	CONTABILIDAD	TENIS
100	ANA		INDOR
150	DANILO	MUSICA	
150	DANILO	MATEMATICAS	TENIS

Solución:

ESTUDIANTE (COD_EST, NOMBRE, ESPECIALIDAD, ACTIVIDAD)

Relación ESTUDIANTE

COD_EST → NOMBRE

→ ESPECIALIDAD

→ ACTIVIDAD

Relación ESTUDIANTE_1 — 3FN

ISBN: 978-631-6557-89-6

COD_EST → NOMBRE

COD_EST	NOMBRE
100	ANA
150	DANILO

Relación ESTUD_ESP (4FN)

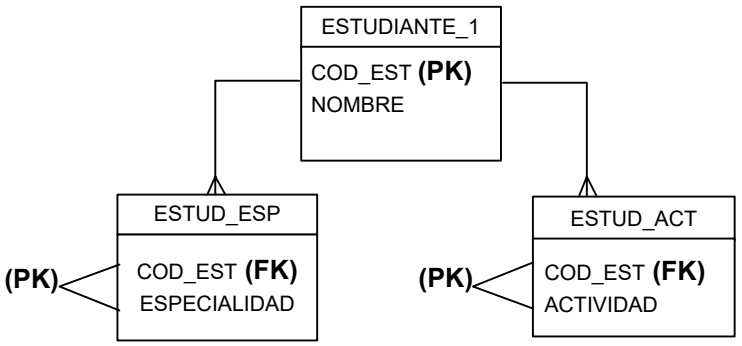
COD_EST →→ ESPECIALIDAD

COD_EST	ESPECIALIDAD
100	MUSICA
100	CONTABILIDAD
150	MUSICA
150	MATEMATICAS

Relación ESTUD_ACT (4FN)

COD_EST →→ ACTIVIDAD

COD_EST	ACTIVIDAD
100	NATACIÓN
100	TENIS
100	INDOR
150	TENIS



9) Realice el diseño conceptual de la base de datos del escenario que se expone a continuación

CASO: FICHA DE RECOLECCIÓN DE INFORMACIÓN – ATRACTIVOS TURÍSTICOS

I. DATOS GENERALES
FECHA: …….
ENCUESTADOR: ……..
ID ATRACTIVO: ……
NOMBRE DEL ATRACTIVO: Artesanías de Guano
CATEGORIA: Manifestaciones culturales
TIPO : Etnografía
SUBTIPO: Cuero y tejidos
2. UBICACIÓN
Provincia: Chimborazo
Cantón: Guano
Parroquia: La Matriz
Centro Urbanístico cercano: Riobamba

ISBN: 978-631-6557-89-6

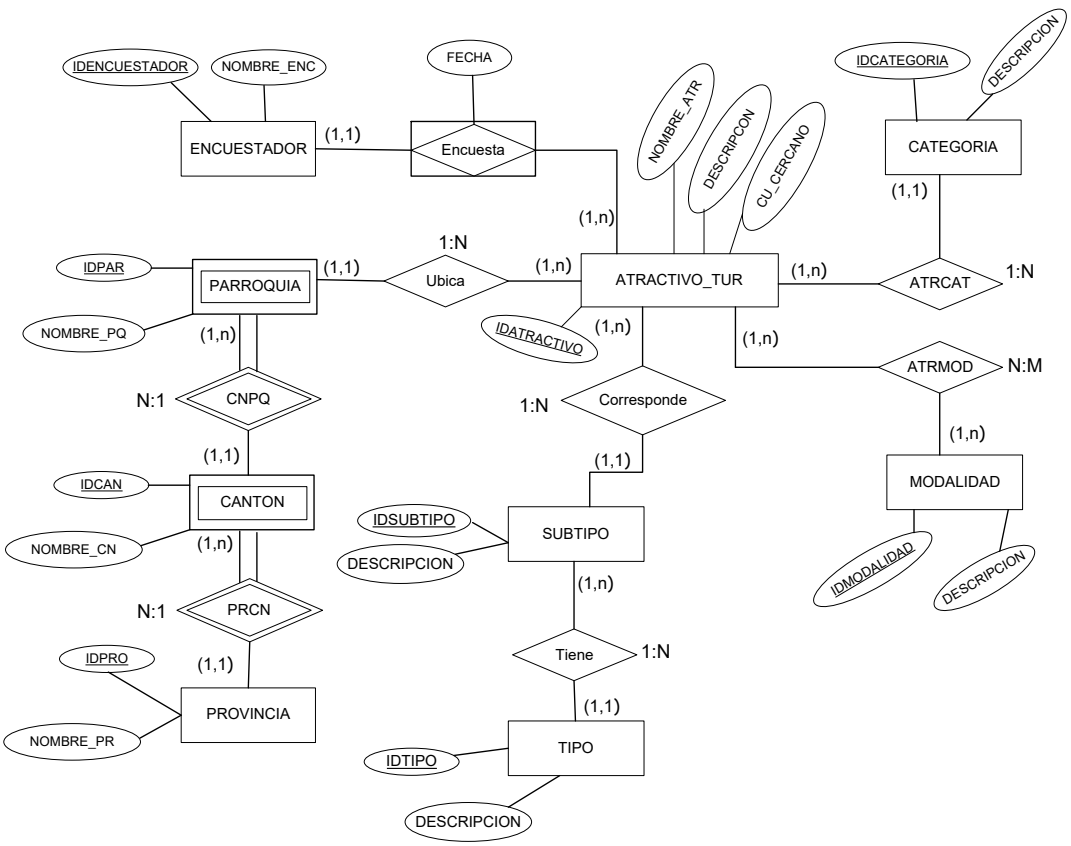
3. DESCRIPCION

Guano llamada "La Capital Artesanal del Ecuador" posee talleres para …….

4. MODALIDAD DE TURISMO A LA QUE PERTENECE EL ATRACTIVO

AGROTURISMO:		TURISMO RURAL:	
TURISMO DE AVENTURA:		TURISMO DE FERIAS Y CONVENCIONES:	X
TURISMO CIENTIFICO:	X	TURISMO DE NEGOCIOS:	
TURISMO HISTORICO:		TURISMO DE COMPRAS:	X
TURISMO DE SALUD:		TURISMO CULTURAL:	X

Solución:



10) Utilizando la Normalización, realice diseño lógico del CASO: FICHA DE RECOLECCIÓN DE INFORMACIÓN – ATRACTIVOS TURÍSTICOS que se expone en el ejercicio 9).

a) Utilizando la Normalización, realice diseño lógico correspondiente

b) Realice el esquema lógico relacional (gráfico)

ISBN: 978-631-6557-89-6

Solución:

Relación CATEGORIA 3FN

IDCATEGORIA → DESCRIPCION

Relación MODALIDAD 3FN

IDMODALIDAD → DESCRIPCION

Relación TIPO 3FN

IDTIPO → DESCRIPCION

3FN

Relación SUBTIPO

IDSUBTIPO → DESCRIPCION, IDTIPO

Relación PROVINCIA 3FN

IDPRO → NOMBRE_PR

Relación CANTON 3FN

IDCAN, IDPRO → NOMBRE_CN

Relación PARROQUIA 3FN

IDPRO, IDCAN, IDPAR → NOMBRE_PQ

Relación ENCUESTADOR 3FN

IDENCUESTADOR → NOMBRE_ENC

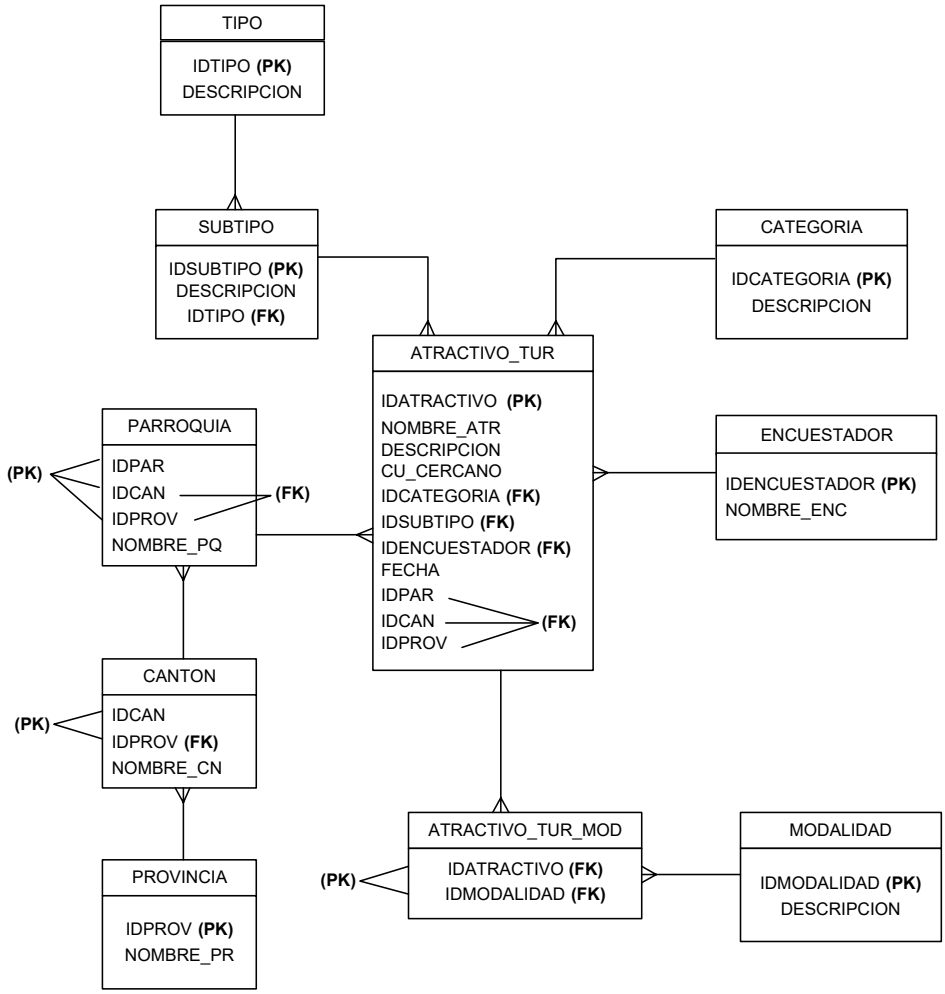
Relación ATRACTIVO_TUR 3FN

IDATRACTIVO → NOMBRE_ATR, DESCRIPCION, CIU_CERCANO, IDSUBTIPO, IDCATEGORIA, IDPRO, IDCAN, IDPAR ID_ENCUESTADOR, FECHA

Relación ATRACTIVO_TUR_MOD 4FN

IDATRACTIVO ↠ IDMODALIDAD

ISBN: 978-631-6557-89-6



11) Con base en la tabla del caso, utilizando la Normalización realice diseño lógico correspondiente

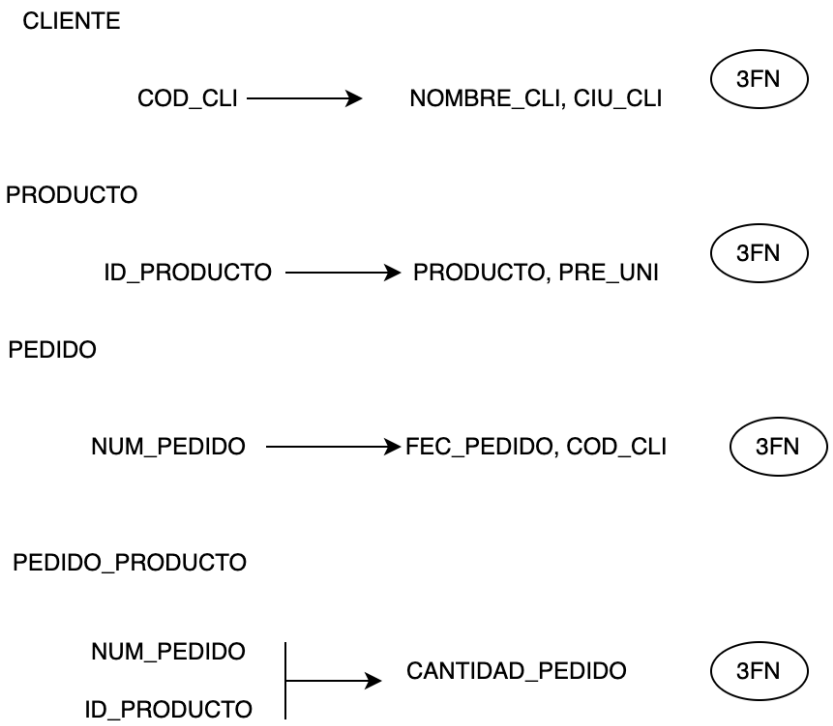
CASO: PEDIDOS DE PRODUCTOS

COD_CLI	NOMBRE _CLIENTE	CIU_CLI	PRE_UNI	ID_ PRODUCTO	PRODUCTO	NUM_ PEDIDO	CANTIDAD_ PEDIDO	FEC_PEDIDO
C1	JUAN	QUITO	8.20	T1	TUERCAS	100	1	05/05/2020
C2	JANET	AMBATO	4.00	R2	RODELAS	150	2	10/12/2020
C2	JANET	AMBATO	8.20	T1	TUERCAS	150	1	10/12/2020
C2	JANET	AMBATO	2.00	C4	CLAVOS	150	3	10/12/2020
C3	BOBY	QUITO	4.00	R2	RODELAS	207	1	08/10/2020
C3	BOBY	QUITO	2.00	C4	CLAVOS	203	2	05/05/2020

ISBN: 978-631-6557-89-6

C4	RITA	GUAYAQUIL	0.50	T6	TORNILLOS	210	1	10/10/2020

Solución:

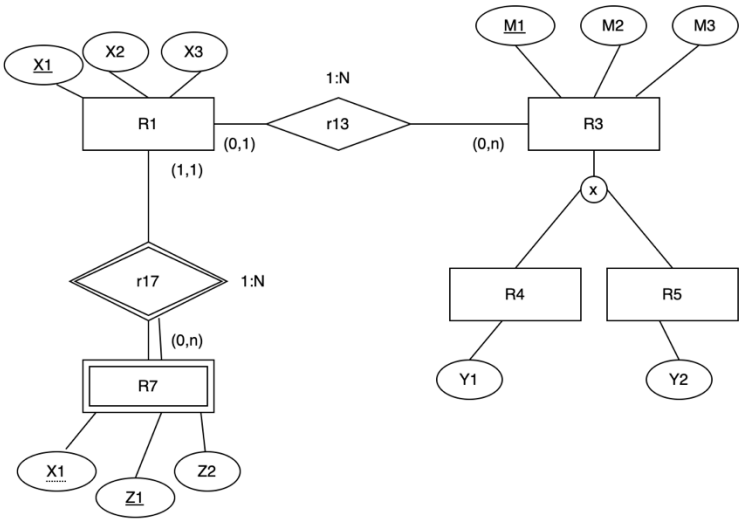


12) Dado el siguiente esquema de base de datos relacional, obtener el DER del cual se haya podido derivar. (1,5)

R1(X1, X2, X3) R3(M1, M2, M3, X1) R4(M1, Y1) R5(M1, Y2) R7(X1, Z1, Z2)

Solución:

ISBN: 978-631-6557-89-6



ISBN: 978-631-6557-89-6

BIBLIOGRAFÍA

Acharya, B., Jena, A. K., Chatterjee, J. M., Kumar, R., & Le, D.-N. (2019). NoSQL Database Classification: New Era of Databases for Big Data. *International Journal of Knowledge-Based Organizations*, *9*(1), 50–65. https://doi.org/10.4018/IJKBO.2019010105

Aguilar Vera, R., Naal Jácome, A., Díaz Mendoza, J., Gómez Gómez, O., Aguilar Vera, R., Naal Jácome, A., Díaz Mendoza, J., & Gómez Gómez, O. (2023). NoSQL Database Modeling and Management: A Systematic Literature Review. *Revista Facultad de Ingeniería*, *32*(65), e16519. https://doi.org/10.19053/01211129.V32.N65.2023.16519

Connolly, T., & Begg, C. (2015). *Database Systems* (6th ed.). Pearson.

Date, C. J. (2004). *An Introduction to Database Systems* (8th ed.). Pearson Educación.

DB-Engines. (n.d.). *DB-Engines Ranking*. Retrieved August 22, 2025, from https://db-engines.com/en/ranking

De Miguel, A., Piattini, M., & Marcos, E. (2000). *Diseño de bases de datos relacionales*. Ra-Ma.

Elmasri, R., & Navathe, S. B. (2011). *Fundamentals of database systems* (6th ed.). Pearson Education.

Elmasri, R., & Navathe, S. B. (2016). *Fundamentals of Database Systems* (7th editio). Pearson.

Garba, M., & Abubakar, H. (2020). A Comparison of NoSQL and Relational Database Management Systems (RDBMS). *KASU JOURNAL OF MATHEMATICAL SCIENCE (Maths Access)*, *1*(2), 61–69.

Gobert, M., Meurice, L., & Cleve, A. (2021). Conceptual Modeling of Hybrid Polystores. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial*

Intelligence and Lecture Notes in Bioinformatics), *13011 LNCS*, 113–122. https://doi.org/10.1007/978-3-030-89022-3_10/COVER

Huawei Technologies Co., Ltd. (2022). *Database Principles and Technologies – Based on Huawei GaussDB* (1st ed.). Springer Singapore. https://doi.org/https://doi.org/10.1007/978-981-19-3032-4

IAIDQ. (2016). *IAIDQ-glossary*. http://iaidq.org/main/glossary.shtml

ISO/IEC 2382-1. (1993). *ISO/IEC 2382-1:1993, Information technology — Vocabulary — Part 1: Fundamental terms*. https://www.iso.org/obp/ui/#iso:std:iso-iec:2382:-1:ed-3:v1:en

Jatana, N., Puri, S., Ahuja, M., Kathuria, I., & Gosain, D. (2012). A Survey and Comparison of Relational and Non-Relational Database. *International Journal of Engineering Research & Technology (IJERT)*, *1*(6).

Kroenke, D. M. ., Auer, D. J. ., Vandenberg, S. L. ., & Yoder, R. C. . (2018). *Database processing : fundamentals, design, and implementation*. Pearson. https://unidel.edu.ng/focelibrary/books/Database Processing Fundamentals, Design, and Implementation by David M. Kroenke, David J. Auer, Robert C. Yoder, Scott L. Vandenberg (z-lib.org).pdf

Lemahieu, W., Broucke, S. vanden, & Baesens, B. (2018). Principles of Database Management: The Practical Guide to Storing, Managing and Analyzing Big and Small Data. *Principles of Database Management*. https://doi.org/10.1017/9781316888773

Lucas Gomez, A. (1993). *Diseño y gestión de sistemas de bases de datos* (1st ed.). Editorial Paraninfo.

Marqués, M. (2011). *Bases de datos* (1st ed.). Universitat Jaume I.

Martinez-Mosquera, D., Navarrete, R., & Lujan-Mora, S. (2020). Modeling and management big data in databases-A systematic literature review. *Sustainability (Switzerland)*, *12*(2). https://doi.org/10.3390/SU12020634

Mendelzon, A. (2000). *Introducción a las Bases de datos relacionales* (1st ed.). Pearson Educación.

Piñeiro Gómez, J. M. (2013). *Bases de datos relacionales y modelado de datos* (1st ed.). Paraninfo.

Rodriguez, I. E. (2020). *Metadatos estadísticos para el aseguramiento de la calidad de los datos*. Universidad Nacional de Colombia - Sede Medellín.

Rowley, J. (2007). The wisdom hierarchy: Representations of the DIKW hierarchy. *Journal of Information Science*, *33*(2), 163–180. https://doi.org/10.1177/0165551506070706;CTYPE:STRING:JOURNAL

Shin, K., Hwang, C., & Jung, H. (2017). NoSQL database design using UML conceptual data model based on peter chen's framework. *International Journal of Applied Engineering Research*, *12*(5), 632–636.

Silberschatz, A., Korth, H. F., & Sudarshan, S. (2014). *Fundamentos de bases de datos*. McGraw-Hill Interamericana de España.

Tripathi, N. (2025). NoSQL database education: A review of models, tools and teaching methods. *Journal of Systems and Software*, *226*, 112391. https://doi.org/10.1016/J.JSS.2025.112391

Veerappampalayam, S., Anilkumar, C., Rajendran, R., Priyanka, P., Prabhakar, A. V., & Moorthy, U. (2025). Next-Generation Databases: A Comparative Insight into NoSQL and NewSQL Systems. *2025 International Conference on Emerging Technologies in Engineering Applications, ICETEA 2025 - Proceedings*. https://doi.org/10.1109/ICETEA64585.2025.11099997

ISBN: 978-631-6557-89-6

DE LOS AUTORES

IVONNE ELIZABETH RODRÍGUEZ FLORES

Doctora en Ingeniería - Sistemas (PhD) de la Universidad Nacional de Colombia, Becaria Senescyt - Ecuador. Máster en Planificación, Evaluación y Acreditación de la Educación Superior y Máster en Informática Aplicada, títulos obtenidos en la Escuela Superior Politécnica de Chimborazo. Ingeniera de Sistemas e Informática de la Escuela Politécnica del Ejercito.

Profesora titular principal en la Facultad de Informática y Electrónica de la ESPOCH. Investigadora senior del Grupo de Investigación en Tecnologías de la Información para la Gestión del Conocimiento (TIGECON) - ESPOCH. El desarrollo de proyectos académicos, profesionales y de investigación se relacionan con el descubrimiento de conocimiento a través de datos, la Ingeniería de Datos y Bases de Datos, con énfasis en la Integración de datos y Data warehousing. Además, los trabajos realizados tienen relación directa con el Análisis y gestión de datos e información, enfocados en el BigData, la Calidad de datos y Modelado de datos. Con experiencia profesional en el sector tecnológico, desarrollo de proyectos tanto en entidades gubernamentales a nivel nacional como en la empresa privada.

Correo electrónico: ivonne.rodriguez@espoch.edu.ec

GISEL KATERINE BASTIDAS GUACHO

Docente e investigadora en el área de Ciencias de la Computación. Doctoranda en Ciencias Computacionales Aplicadas en la Escuela Superior Politécnica del Litoral. Posee una Maestría en Ciencias de la Computación por la University of California, Riverside y el título de Ingeniera en Sistemas Informáticos por la Escuela Superior Politécnica de Chimborazo. Se desempeña como docente universitaria en la Escuela Superior Politécnica de Chimborazo. Cuenta con experiencia profesional como Ingeniera Senior de Business Intelligence en TopTech Advisors y Produbanco, participando en proyectos de analítica de datos para el sector financiero. Ha trabajado como analista y administradora de bases de datos en el Ministerio del Interior y en CNT EP. Sus líneas de investigación incluyen minería de datos, aprendizaje automático y visión por computador.

Correo electrónico: gis.bastidas@espoch.edu.ec

Este libro ofrece una introducción integral al diseño de bases de datos, abordando de manera progresiva los fundamentos conceptuales, técnicos y metodológicos necesarios para su correcta implementación. A lo largo de cinco capítulos, desarrolla los conceptos básicos de las bases de datos y los principales modelos de datos, el modelo relacional con su estructura y restricciones, y los principios del diseño orientados a garantizar coherencia, integridad y eficiencia en entornos reales.

Asimismo, presenta el diseño conceptual mediante el modelo entidad–relación y el diseño lógico, incluyendo la transformación de esquemas conceptuales a esquemas relacionales y la aplicación de la normalización como herramienta clave para reducir redundancias y evitar anomalías en los datos. Con ejercicios prácticos en cada capítulo, esta obra constituye una guía clara para estudiantes, docentes y profesionales que buscan fortalecer sus competencias en el diseño de bases de datos relacionales.

PUERTO MADERO EDITORIAL

ISBN 978-631-6557-89-6