

# Bases de Datos

Guía didáctica



## Unidad Académica Técnica y Tecnológica

### Tecnología Superior en Transformación Digital de Empresas

---

## Bases de Datos

### Guía didáctica

Carrera	PAO Nivel
▪ <i>Tecnología Superior en Transformación Digital de Empresas</i>	III

### Autores:

Encalada Encalada Ángel Eduardo  
Morocho Yunga Juan Carlos



Asesoría virtual  
[www.utpl.edu.ec](http://www.utpl.edu.ec)

## **Universidad Técnica Particular de Loja**

### **Bases de Datos**

#### **Guía didáctica**

Encalada Encalada Ángel Eduardo

Morocho Yunga Juan Carlos

#### **Diagramación y diseño digital:**

Ediloja Cía. Ltda.

Telefax: 593-7-2611418.

San Cayetano Alto s/n.

[www.ediloja.com.ec](http://www.ediloja.com.ec)

[edilojacialtda@ediloja.com.ec](mailto:edilojacialtda@ediloja.com.ec)

Loja-Ecuador

ISBN digital - 978-9942-39-621-1



#### **Reconocimiento-NoComercial-CompartirIgual 4.0 Internacional (CC BY-NC-SA 4.0)**

Usted acepta y acuerda estar obligado por los términos y condiciones de esta Licencia, por lo que, si existe el incumplimiento de algunas de estas condiciones, no se autoriza el uso de ningún contenido.

Los contenidos de este trabajo están sujetos a una licencia internacional Creative Commons **Reconocimiento-NoComercial-CompartirIgual 4.0 (CC BY-NC-SA 4.0)**. Usted es libre de **Compartir – copiar y redistribuir el material en cualquier medio o formato. Adaptar – remezclar, transformar y construir a partir del material citando la fuente, bajo los siguientes términos: Reconocimiento- debe dar crédito de manera adecuada, brindar un enlace a la licencia, e indicar si se han realizado cambios.** Puede hacerlo en cualquier forma razonable, pero no de forma tal que sugiera que usted o su uso tienen el apoyo de la licenciante. **No Comercial-no puede hacer uso del material con propósitos comerciales. Compartir igual-Si remezcla, transforma o crea a partir del material, debe distribuir su contribución bajo la misma licencia del original.** No puede aplicar términos legales ni medidas tecnológicas que restrinjan legalmente a otras a hacer cualquier uso permitido por la licencia. <https://creativecommons.org/licenses/by-nc-sa/4.0/>

# Índice

<b>1. Datos de información .....</b>	<b>6</b>
1.1. Presentación de la asignatura .....	6
1.2. Competencias genéricas de la UTPL.....	6
1.3. Competencias específicas de la carrera .....	6
<b>2. Metodología de aprendizaje .....</b>	<b>7</b>
<b>3. Orientaciones didácticas por resultados de aprendizaje .....</b>	<b>8</b>
 <b>Primer bimestre.....</b>	 <b>8</b>
<b>Resultado de aprendizaje 1.....</b>	<b>8</b>
 Semana 1 .....	 8
 Unidad 1. Sistemas y modelos de bases de datos .....	 8
Actividades de aprendizaje recomendadas .....	13
 Semana 2 .....	 13
Actividad de aprendizaje recomendada.....	21
 Semana 3 .....	 21
Actividad de aprendizaje recomendada.....	32
 Semana 4 .....	 33
Actividades de aprendizaje recomendadas.....	39
 Semana 5 .....	 39
Actividades de aprendizaje recomendadas .....	44
 Semana 6 .....	 44
Actividades de aprendizaje recomendadas .....	53
 Semana 7 .....	 54
Actividades de aprendizaje recomendadas .....	59

<b>Semana 8 .....</b>	<b>60</b>
<b>Segundo bimestre .....</b>	<b>61</b>
<b>Resultado de aprendizaje 2.....</b>	<b>61</b>
<b>Semana 9 .....</b>	<b>61</b>
<b>Unidad 2. Diseño de bases de datos.....</b>	<b>61</b>
Actividad de aprendizaje recomendada.....	67
<b>Semana 10 .....</b>	<b>67</b>
Actividad de aprendizaje recomendada.....	70
<b>Semana 11 .....</b>	<b>70</b>
Actividad de aprendizaje recomendada.....	78
<b>Resultado de aprendizaje 3.....</b>	<b>79</b>
<b>Semana 12 .....</b>	<b>79</b>
<b>Unidad 3. Implementación y administración de bases de datos.....</b>	<b>79</b>
Actividades de aprendizaje recomendadas .....	87
<b>Semana 13 .....</b>	<b>87</b>
Actividades de aprendizaje recomendadas .....	91
<b>Semana 14 .....</b>	<b>91</b>
Actividades de aprendizaje recomendadas .....	99
<b>Semana 15 .....</b>	<b>99</b>
Actividades de aprendizaje recomendadas .....	108
<b>Semana 16 .....</b>	<b>109</b>
<b>4. Referencias bibliográficas .....</b>	<b>110</b>



---

## 1. Datos de información

---

### 1.1. Presentación de la asignatura



### 1.2. Competencias genéricas de la UTPL

- Orientación a la innovación y a la investigación.
- Pensamiento crítico y reflexivo.
- Trabajo en equipo.

### 1.3. Competencias específicas de la carrera

- Desarrolla aplicaciones empresariales aplicando enfoques centrados en la nube.
- Diseña modelos de negocio digitales de acuerdo al contexto en que se desenvuelve la organización.
- Diseña modelos arquitectónicos de empresa para gestionar el alineamiento estratégico entre negocio y TI.

- Despliega infraestructura tecnológica para operar digitalmente los dominios arquitectónicos de una organización.



---

## 2. Metodología de aprendizaje

---

En el desarrollo de la asignatura de Bases de Datos emplearemos el método del estudio de caso, pues este método le permitirá al estudiante, a través de un caso propuesto por el docente, entrenarse en la generación de soluciones a una situación dada. El caso describe una situación surgida en la vida real de una persona u organización, para lo que se aporta datos que describan con mucha precisión el entorno del caso. El caso no proporciona al alumno soluciones para ser analizadas, sino que exige que el alumno sea el generador de soluciones encaminadas a resolver el caso propuesto.

A través del aprendizaje autónomo, el estudiante interiorizará el escenario planteado, y reflexionará sobre posibles estrategias de solución; luego, mediante el aprendizaje en contacto con el docente, discutirá las posibles soluciones y seleccionará la más viable, la cual la llevará a ejecución en el marco del aprendizaje práctico experimental. El estudiante tendrá la asesoría y guía del tutor académico durante todo el proceso, tanto en la socialización y comprensión del alcance del caso, en la selección de mejor estrategia de solución, y en la aplicación de métodos y herramientas que permitan efectivizar la solución.



### 3. Orientaciones didácticas por resultados de aprendizaje



#### Primer bimestre

##### Resultado de aprendizaje 1

- Explica los diferentes elementos que conforman un esquema de base de datos.

#### Contenidos, recursos y actividades de aprendizaje



##### Semana 1

#### Unidad 1. Sistemas y modelos de bases de datos



En esta semana iniciamos el estudio de las bases de datos y para ello debemos conocer primero su evolución y qué compone el entorno de un sistema de base de datos.

## Sistemas de base de datos

Actualmente, tanto desde las organizaciones pequeñas hasta las grandes corporaciones y entidades de gobierno hacen uso de bases de datos.

Muchas veces sin ser conscientes, estamos interactuando con bases de datos, por ejemplo, al obtener un resumen de transacciones de nuestra cuenta bancaria a través de la banca web, cuando usamos una tarjeta de crédito, cuando compramos en un supermercado, al consultar los datos de calificaciones de nuestros estudios, al acceder a una biblioteca y hacer búsqueda de contenido, y estos son solo unos pocos ejemplos de uso de las bases de datos.

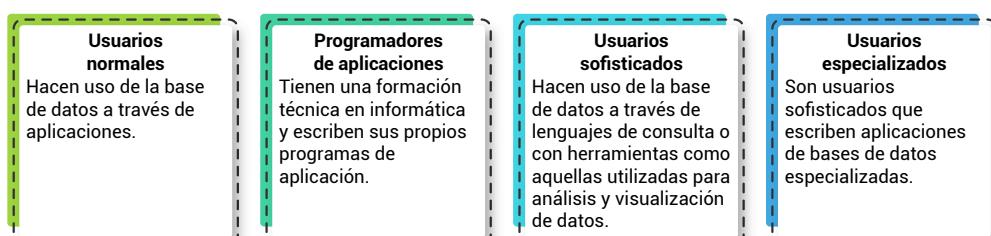
Entonces, una base de datos es una herramienta que nos permite gestionar grandes cantidades de datos de una manera ágil y ordenada, sin embargo, si está mal diseñada puede acarrear problemas al proporcionar datos no actualizados o erróneos.

Dentro de los sistemas de base de datos, encontramos al Sistema Gestor de Base de Datos (SGBD), que actúa como un intérprete de órdenes que los usuarios envían a la base de datos. A través del SGBD se puede acceder a los datos almacenados para actualizar, borrar, consultar o ingresar nuevos datos. También, dentro de la gestión de los datos se incluyen aquellas funciones para definición de las estructuras para almacenar información.

Pero todas las funciones de un SGBD y de la base de datos en sí misma no actúan solas y requieren de personal para que las pongan a funcionar. Dentro de los roles que las personas deben cumplir en un sistema de base de datos están los usuarios que hacen uso de los datos almacenados. La figura 1 muestra los tipos de usuarios:

**Figura 1**

*Tipos de usuarios de un sistema de base de datos*



Nota. Encalada, E. Morocho, J., 2022.

Por otro lado, están los usuarios que tienen control sobre el sistema, conocidos como administrador de datos, y administrador de la base de datos.

### ***Administrador de la Base de Datos (DBA)***

Es el responsable de las actividades relacionadas con el buen funcionamiento de una base de datos. Sus responsabilidades pueden ir desde actividades de diseño de la base de datos, la construcción de sistemas o el mantenimiento y ajuste de los sistemas existentes.

### ***Administrador de datos***

Se encarga de cuidar lo referente al suministro, uso y mantenimiento de los datos de una organización. Además, debe velar por la definición e implementación de procesos que garanticen el buen uso de los datos en la organización.

### **Bases de datos relacionales**

El modelo relacional da lugar a las bases de datos relacionales, que usan un conjunto de tablas para representar tanto los datos como las relaciones existentes entre ellos. Incluyen un Lenguaje de Definición de Datos (DDL) y un Lenguaje de Manipulación de Datos (DML).

### **Tipos de bases de datos**

Con el avance de la tecnología las necesidades de los usuarios fueron cambiando y ampliándose cada vez más. Entonces se fueron creando varios tipos de bases de datos basadas en diferentes criterios: por el tipo de datos que almacenan, por la cantidad de usuarios, entre otras características. A continuación, presentamos distintos tipos de bases de datos.

#### **Tipos de Bases de datos**

En los últimos años, un nuevo enfoque de uso de las bases de datos ha tomado forma debido al requerimiento de manejo de grandes cantidades de información de aplicaciones como *Facebook*, *Twitter*, entre otras. Se conoce como bases de datos NoSQL (*Not Only SQL*) y es una nueva generación de SGBD que no se basa en el modelo tradicional de base de datos relacional. Las bases de datos NoSQL están diseñadas para manejar grandes

volúmenes de información, variedad de tipos y estructuras de datos y velocidad de operación que requieren las aplicaciones de hoy en día.

## **Diseño de base de datos**

El diseño de una base de datos es un proceso de vital importancia para el buen funcionamiento de un negocio. Por lo tanto, las actividades de diseño de base de datos deben tomarse con la seriedad y paciencia necesarias, de forma que se produzca un buen diseño que permita una buena gestión tanto de los datos como el adecuado aprovechamiento de los recursos tecnológicos.

Una base de datos bien diseñada facilita la gestión de los datos y genera información precisa y valiosa. Por el contrario, un mal diseño de base de datos puede producir datos erróneos que harán tomar malas decisiones y por ende causar problemas en una organización.

Un buen diseño de base de datos debe apoyarse en una metodología de diseño que proporciona un conjunto de procesos, procedimientos, herramientas y formas de documentar todo el proyecto de diseño de una base de datos.

## **La evolución desde los sistemas basados en archivos**

Antes de las bases de datos la información se organizaba de forma manual, debido a que las cantidades de información eran pequeñas y, por tanto, manejables por los usuarios que tenían pocas necesidades de información, a estos sistemas se les llamaba sistemas basados en archivos.

Los sistemas basados en archivos manejaban, como su nombre lo dice, archivos de datos, pero estaban en cada departamento ofreciendo soluciones de información a los usuarios de esa dependencia. Podía darse el caso que en los departamentos la información se duplicase debido a que cada departamento manejaba sus propios datos. Además, cada vez que surgían nuevas necesidades de información se generaban nuevas aplicaciones que accedían a los datos de los archivos, lo que provocaba que existiera un gran número de aplicaciones distintas. De igual forma, cuando se requería un nuevo informe se debía invertir una gran cantidad de esfuerzo para lograrlo. Debido a que cada departamento tenía sus propias aplicaciones para administrar sus datos y, por tanto, manejaba sus propios

formatos y lenguajes de programación, el compartir la información con otras dependencias resultaba todo un reto.

Hacía falta desarrollar una estructura de datos más compleja donde la información de múltiples entidades de información se pude almacenar en un solo repositorio unificado e integrado, la base de datos.



Le invitamos a realizar la siguiente lectura, sobre el origen y evolución de las bases de datos.

[Origen de las bases de datos \(Pulido et al., 2019, pp. 8-15\)](#)

En los años 70 es quizá donde se produce el hito más importante, *Edgar Frank Codd*, científico informático inglés conocido por sus aportaciones a la teoría de bases de datos relacionales, definió el modelo relacional a la par que publicó una serie de reglas para los sistemas de datos relacionales a través de su artículo "Un modelo relacional de datos para grandes bancos de datos compartidos". Este trabajo es considerado como el punto de partida de los sistemas gestores de bases de datos relacionales que tenemos hasta la actualidad.



Para complementar detalles de la evolución de las bases de datos vaya a YouTube y busque el video titulado "[History of Databases](#)" publicado por *Computer History Museum*.

Como vemos, los modelos de bases de datos han ido cambiando progresivamente. Luego de los modelos jerárquicos y de red, se dio un salto cualitativo a las bases de datos relacionales, que son las que han venido dominando el mercado desde los años 70. Sin embargo, actualmente han venido surgiendo nuevos modelos de bases de datos, que buscan acoplarse a nuevas necesidades, propósitos, y tendencias, que se las conoce como bases de datos *NoSQL*.



## Actividades de aprendizaje recomendadas

- Ante todo, en esta primera semana acceda al entorno virtual de aprendizaje [EVA](#), utilice sus credenciales facilitadas por la universidad, y explore con detalle el contenido del curso. Luego de familiarizarse con el EVA si tiene alguna duda o no puede ingresar al curso, recuerde que puede solicitar ayuda a su tutor.
- Amplíe la comprensión de lo estudiado, investigando y contrastando con nuevas fuentes, los orígenes, la evolución, y la importancia de los sistemas de bases de datos.



## Semana 2

---

En la presente semana vamos a revisar qué son los modelos de datos y para qué sirven, también revisaremos la propuesta de los niveles de abstracción que debe implementarse en un sistema de base de dato para finalmente conocer qué se propone dentro del modelo relacional de base de datos.

### Modelos de datos

El diseño de la base de datos se concentra en representar, lo más parecido a la realidad, un problema o actividad que cumple una organización. El modelo de datos es una abstracción, generalmente representada de forma gráfica, de un hecho u objeto del mundo real.

El modelo de datos se convierte en una representación de las necesidades de información de los usuarios. El modelo de datos es considerado una guía para la implementación escrita en un lenguaje sencillo y claro y debería contener, al menos, los siguientes componentes (Coronel, 2020):

- La descripción de la estructura de datos que guardará los datos del usuario final.
- Un conjunto de reglas que se utilizan para garantizar la integridad de los datos.

- Una metodología de manipulación de datos para apoyar las transformaciones de los datos reales.

Se debe aclarar también, que se pueden generar varios modelos de datos para un mismo problema, sin embargo, el mejor será aquel que satisfaga las necesidades de datos de los usuarios. Hay que considerar que los usuarios ven los datos de manera diferente, un directivo tendrá una visión universal de los datos porque necesita integrar todas las áreas del negocio, mientras que un director de departamento tendrá una visión más restringida debido a que trabaja solamente con un subconjunto de datos de la empresa.

Pero para construir una base de datos sólida se requiere un buen modelo de datos, haciendo una analogía, para construir una casa el propietario primero debe tener claro cómo se construirán las habitaciones en conjunto y eso lo puede observar en los planos. Cuando se dispone de un buen plano de la base de datos, la gestión de los datos se torna más simple y las aplicaciones se construyen fácilmente. Por el contrario, sin un plano, es poco probable que se cree una buena base de datos.

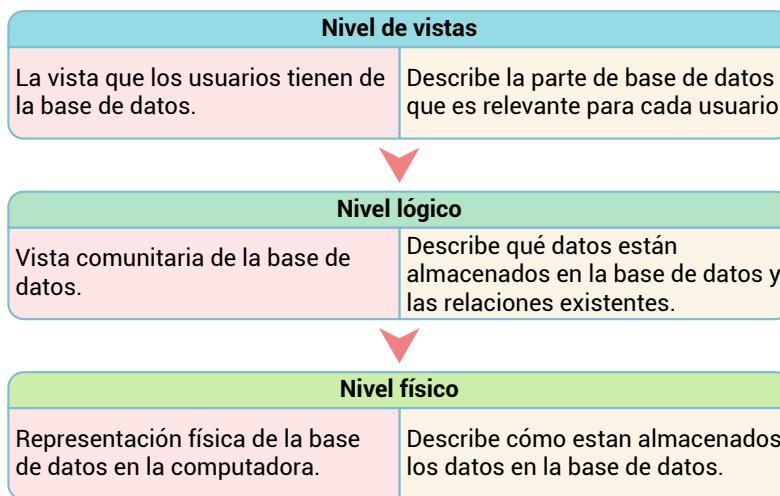
### Niveles de abstracción

Uno de los objetivos principales de los sistemas de bases de datos es ofrecer al usuario una visión abstracta de los datos, lo que significa que el sistema oculta ciertos detalles del almacenamiento y la manipulación de los datos. El objetivo de esta arquitectura es separar la visión que cada usuario tiene de la base de datos de la forma en que está representada físicamente.

Los SGBD comerciales están basados en la denominada arquitectura ANSI-SPARC que comprende tres niveles de abstracción. La figura 2 resume los niveles de abstracción de bases de datos.

**Figura 2**

*Niveles de abstracción de base de datos*

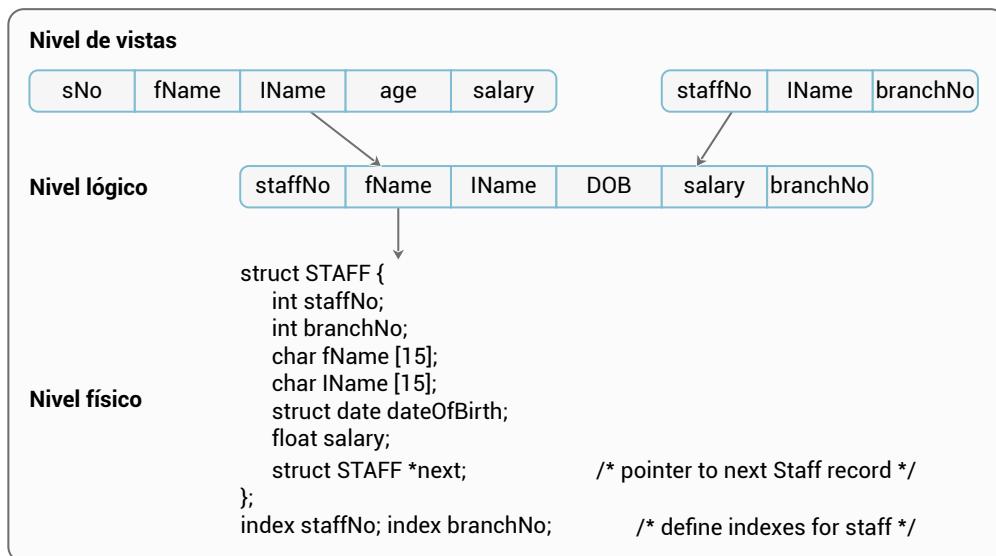


Nota. Tomado de *Guía didáctica de Fundamentos de Bases de Datos* (p. 23), por J. Morocho y A. Romero, 2019, Editorial de la Universidad Técnica Particular de Loja.

La figura 3 presenta un ejemplo donde se esquematizan las diferencias entre los tres niveles de abstracción.

**Figura 3**

Diferencias entre los niveles de abstracción



Nota. Adaptado de *Sistemas de bases de datos. Un enfoque práctico para diseño, implementación y gestión* (p. 35), por T. M. Connolly y C. E. Begg, 2005, Pearson Educación.

Se debe apuntar también que entre los niveles existe lo que se conoce como independencia lógica e independencia física de los datos. La independencia lógica de los datos se da cuando se puede hacer cambios en el nivel conceptual sin afectar el nivel de vistas. La independencia física de los datos se refiere a que los cambios que se efectúen en el nivel físico no deben afectar al nivel conceptual.



En el siguiente recurso encontrará una explicación más amplia del modelo ANSI/SPARC para bases de datos. Revise el apartado [a2.5] Modelo ANSI/X3/SPARC.

[Estándares y modelo ANSI, apartado \[a2.5\]  
Modelo ANSI/X3/SPARC \(Sánchez, s. f.-a\)](#)

El recurso recomendado presenta una descripción del modelo definido por ANSI (*American National Standards Institute*) que es un organismo científico de Estados Unidos que define estándares en el campo de las bases de

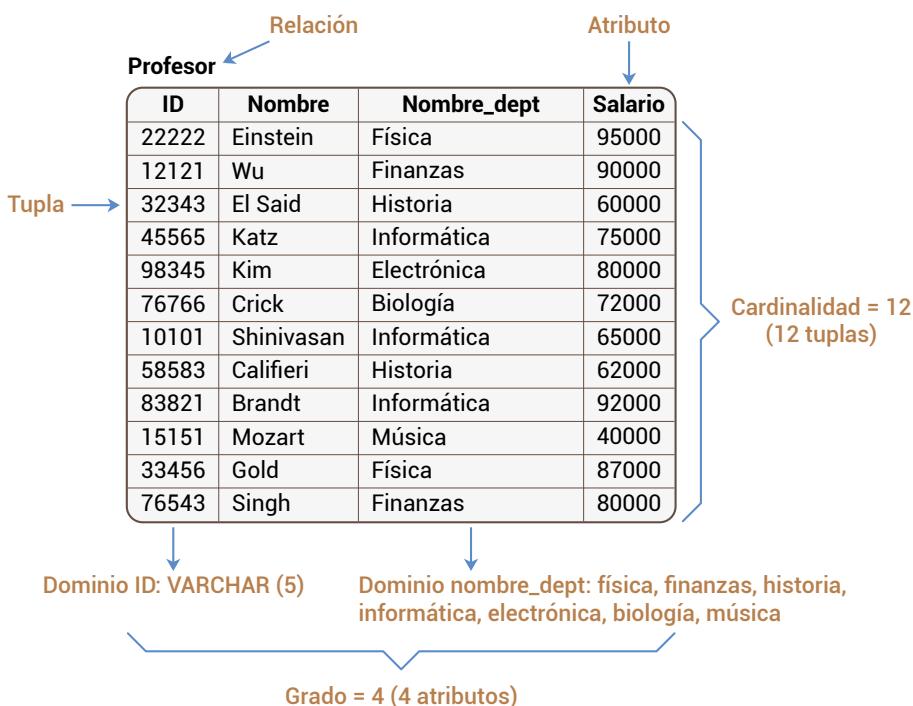
datos. También, muestra con más detalle los niveles definidos por el estándar que los denomina como: externo, conceptual e interno.

## El modelo relacional

El propósito esencial de una base de datos es el almacenamiento de datos. Una base de datos relacional consiste en un conjunto de relaciones, a las que también las podemos denominar tablas, de ahí que se denomine modelo relacional.

Es importante identificar los elementos que son parte de la terminología, lo que permite identificar dichos conceptos en ejercicios prácticos. Con este propósito, la figura 4 presenta los elementos básicos del modelo relacional.

**Figura 4**  
*Elementos básicos del modelo relacional*



Nota. Tomado de *Guía didáctica de Fundamentos de Bases de Datos* (p. 44), por J. Morocho y A. Romero, 2019, Editorial de la Universidad Técnica Particular de Loja.

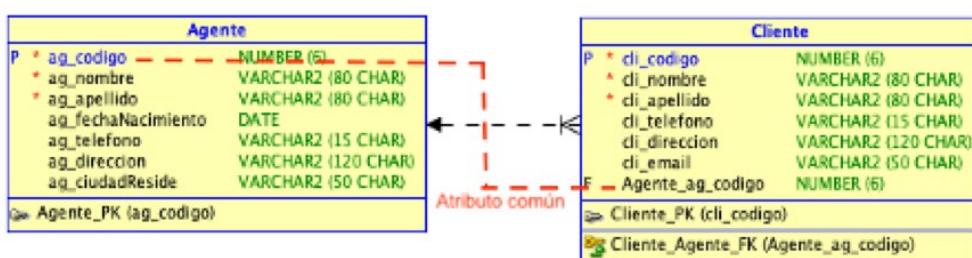
La base del modelo relacional es un concepto matemático conocido como relación. Una **relación** (a veces llamada tabla) puede pensarse como una estructura bidimensional compuesta por filas y columnas que se cruzan.

Cada fila de una relación se denomina tupla. Cada columna representa un **atributo** o característica que define a la relación. Una **tupla** en el modelo relacional es una fila de datos de la relación o tabla y que no se debería duplicar. Un **dominio** lo constituye el conjunto de posibles valores que puede tomar un atributo. El **grado** de una relación lo determina el número de atributos de la relación y suele ser estático (lo que no quiere decir que no se pueda cambiar en algún momento). La **cardinalidad** la determina el número de tuplas de la relación y suele ser cambiante (porque en una tabla se insertan y/o borran filas de datos constantemente).

El modelo de datos relacional se implementa a través de un sistema de gestión de bases de datos relacionales (RDBMS) muy sofisticado. El RDBMS oculta las complejidades del manejo de los datos al usuario gestionando todos los detalles físicos, de forma que el usuario solamente ve tablas con datos.

Las tablas están relacionadas entre sí al compartir un atributo común (un valor en una columna) lo que facilita que se puedan asociar fácilmente los datos entre las tablas, aunque sean independientes entre sí, vea el ejemplo de la figura 5. El modelo relacional proporciona un nivel mínimo de redundancia controlada para eliminar la mayoría de las redundancias que se encuentran habitualmente en los sistemas de archivos.

**Figura 5**  
*Diagrama relacional*



Nota. Elaboración propia realizada en la aplicación Oracle Datamodeler

En el ejemplo de la figura 5, el cliente representa el lado “muchos” porque un agente puede tener muchos clientes. El agente representa el lado “1” porque cada cliente solo tiene un agente.

## Claves

Dentro del modelo relacional se pueden distinguir varios tipos de claves que se utilizan para imponer ciertas restricciones y son las siguientes:

- Superclave.
- Clave candidata.
- Clave primaria.
- Clave alternativa.
- Clave externa o foránea.

Utilice el ejemplo de la figura 5 para identificar cada uno de los tipos de claves en la relación cliente.

- Superclave: cli\_codigo + cli\_nombre + cli\_apellido
- Clave candidata: cli\_codigo, cli\_nombre + cli\_apellido
- Clave primaria: cli\_codigo
- Clave alternativa: cli\_nombre + cli\_apellido
- Clave externa o foránea: Agente\_ag\_codigo



Para complementar el estudio de las claves  
relacionales revise la siguiente información.

[Modelo relacional, apartado 3.2.9 Claves  
\(Sánchez, s. f.-b\)](#)

El recurso muestra una explicación de los principales componentes del modelo relacional y algunos ejemplos que complementan su conocimiento acerca del tema. También presenta lo relacionado a los tipos de claves que se manejan dentro del modelo para asegurar el correcto funcionamiento de la base de datos.

## Restricciones

El modelo relacional impone ciertas reglas que son de cumplimiento obligatorio para el normal funcionamiento de la base de datos:

- Integridad de entidades (primary key)
- Restricción de unicidad (unique)
- Restricción de obligatoriedad (not null)

- Restricción de clave alternativa
- Integridad referencial (foreign key)
- Integridad de dominios

### **Integridad de entidades (primary key)**

Toda tabla debe tener una clave primaria que permita identificar cada fila de manera única y así evitar redundancia. Por lo tanto, al definir un atributo como clave primaria no puede aceptar valores nulos ni repetidos.

### **Restricción de unicidad (unique)**

Obliga a que los valores en una columna marcada con este tipo de restricción no se puedan repetir y sean únicos.

### **Restricción de obligatoriedad (not null)**

Impide que un atributo marcado con este tipo de restricción acepte valores nulos (null). Las columnas definidas como clave alternativa también deben aplicar esta restricción.

### **Restricción de clave alternativa**

Al determinar que una o más columnas son clave alternativa no pueden aceptar valores nulos ni repetirse, por lo tanto, debe aplicarse las restricciones de unicidad (unique) y de obligatoriedad (not null). Se debe aclarar que en los SGBD no es posible marcar una columna como clave alternativa.

### **Integridad referencial (foreign key)**

En caso de relaciones que tengan claves externas (foráneas), el valor de la clave externa debe corresponderse con el valor de la clave primaria o clave candidata de alguna tupla de la relación de origen, o el valor de la clave externa debe ser completamente nulo.

### **Integridad de dominios**

Se entiende como valores válidos para un atributo al conjunto de posibles valores determinados por el dominio.



Para complementar el estudio del modelo relacional, revise la siguiente información.

[El modelo relacional \(Sánchez, s. f.-b\)](#)

Una de las grandes virtudes del modelo relacional es que permite mantener una clara independencia de la estructura lógica respecto a modo de almacenamiento y físico de los datos. En general, los sistemas gestores de bases de datos relacionales implementan a nivel lógico todos los principios que definen al modelo relacional, por lo que a ese nivel son muy similares. La diferencia sustancial está a nivel físico, donde cada motor de base de datos está en libertad de manejar sus propios esquemas de almacenamiento.



### Actividad de aprendizaje recomendada

- Con base en lo estudiado en las semanas 1 y 2 diseñe un mapa mental en donde se represente los principales elementos del modelo relacional. Esto le permitirá medir el grado de comprensión de la terminología y conceptualización estudiada hasta ahora.



### Semana 3

---

Todo Sistema Gestor de Base de Datos (SGBD) provee un lenguaje de dominio que permite administrar el servicio de la base de datos, y realizar operaciones de manipulación y exploración de los datos esta almacena. Durante la presente iniciaremos el estudio de SQL que es lenguaje que implementan todos los sistemas de bases de datos relacionales.

### SQL básico

El manejo del lenguaje SQL es una competencia importante en la gestión de base de datos y una de las competencias más requeridas en todo perfil del profesional informático.

Para llevar a la práctica el lenguaje SQL trabajaremos con el SGBD **MySQL**, le recomendamos si aún no lo ha hecho, descargue e instale dicho motor y las herramientas que facilitan la gestión de la base de datos.

## Introducción al lenguaje SQL

SQL es la principal herramienta para la interacción del usuario con una base de datos. A través de este lenguaje se envían solicitudes de datos al servidor, que es el encargado de responder a dichas peticiones.

SQL es un lenguaje relativamente fácil de aprender, en donde el usuario especifica qué se debe hacer, pero no cómo hacerlo.

SQL se concentra en la definición y manipulación de datos, lo que la hace una herramienta poderosa para la gestión de datos.

Los principales comandos del lenguaje SQL se agrupan en 4 sublenguajes:

- Lenguaje de Definición de Datos (DDL)
- Lenguaje de Manipulación de Datos (DML)
- Lenguaje de Control de Transacciones (TCL)
- Lenguaje de Control de Datos (DCL)

**Lenguaje de Definición de Datos (DDL)**, abarca sentencias del tipo ALTER, DROP y CREATE para crear/eliminar o modificar las estructuras de almacenamiento requeridas. La definición de vistas, se especifican mediante el comando CREATE VIEW. La Tabla 1 muestra un listado de las principales sentencias DDL.

**Tabla 1.**

*Listado de sentencias de definición de datos (DDL) en SQL*

COMMAND OR OPTION	DESCRIPTION
CREATE SCHEMA AUTHORIZATION	Creates a database schema
CREATE TABLE	Creates a new table in the user's database schema
NOT NULL	Ensures that a column will not have null values
UNIQUE	Ensures that a column will not have duplicate values
PRIMARY KEY	Defines a primary key for a table
FOREIGN KEY	Defines a foreign key for a table

COMMAND OR OPTION	DESCRIPTION
DEFAULT	Defines a default value for a column (when no value is given)
CHECK	Validates data in an attribute
CREATE INDEX	<b>Creates an index for a table</b>
CREATE VIEW	<b>Creates a dynamic subset of rows and columns from one or more tables</b>
ALTER TABLE	<b>Modifies a table's definition (adds, modifies, or deletes attributes or constraints)</b>
CREATE TABLE AS	<b>Creates a new table based on a query in the user's database schema</b>
DROP TABLE	<b>Permanently deletes a table (and its data)</b>
DROP INDEX	<b>Permanently deletes an index</b>
DROP VIEW	<b>Permanently deletes a view</b>

Nota. Tomado de *Database Systems: Design, Implementation, & Management* (p. 247), por C. Coronel y S. Morris, 2019, Cengage Learning.

Restricciones de integridad, se expresan mediante los comandos *PRIMARY KEY*, *FOREIGN KEY* y *NOT NULL* que permiten asegurar la integridad de los datos.



Para conocer más detalles sobre los comandos DDL, le invitamos a revisar el siguiente material de apoyo:

[Comandos para definición de datos \(Coronel, 2011\), pp. 223-237.](#)

En (Coronel, 2011) se muestra el funcionamiento de los comandos de definición de datos de SQL, los ejemplos ahí desarrollados ayudan a una mayor comprensión de la sintaxis de la sentencia y su comportamiento.

## Lenguaje de Manipulación de Datos (DML)

Contiene las sentencias *SELECT*, *INSERT*, *UPDATE* y *DELETE*, que manipulan los datos en una base de datos. La Tabla 2 presenta una lista de los principales comandos DML.

**Tabla 2.**

*Listado de sentencias de manipulación de datos (DML) en SQL*

COMMAND, OPTION, OR OPERATOR	DESCRIPTION
<b>SELECT</b>	<b>Selects attributes from rows in one or more tables or views</b>
<b>FROM</b>	Specifies the tables from which data should be retrieved
<b>WHERE</b>	Restricts the selection of rows based on a conditional expression
<b>GROUP BY</b>	Groups the selected rows based on one or more attributes
<b>HAVING</b>	Restricts the selection of grouped rows based on a condition
<b>ORDER BY</b>	Orders the selected rows based on one or more attributes
<b>INSERT</b>	<b>Inserts row(s) into a table</b>
<b>UPDATE</b>	<b>Modifies an attribute's values in one or more table's rows</b>
<b>DELETE</b>	<b>Deletes one or more rows from a table</b>
<b>Comparison operators</b>	
=, <, >, <=, >=, <>, !=	Used in conditional expressions
<b>Logical operators</b>	
AND/OR/NOT	Used in conditional expressions
<b>Special operators</b>	<b>Used in conditional expressions</b>
BETWEEN	Checks whether an attribute value is within a range
IN	Checks whether an attribute value matches any value within a value list
LIKE	Checks whether an attribute value matches a given string pattern
IS NULL	Checks whether an attribute value is null
EXISTS	Checks whether a subquery returns any rows
DISTINCT	Limits values to unique values
<b>Aggregate functions</b>	<b>Used with SELECT to return mathematical summaries on columns</b>
COUNT	Returns the number of rows with non-null values for a given column
MIN	Returns the minimum attribute value found in a given column
MAX	Returns the maximum attribute value found in a given column
SUM	Returns the sum of all values for a given column
AVG	Returns the average of all values for a given column

Nota. Tomado de *Database Systems: Design, Implementation, & Management* (p. 246), por C. Coronel y S. Morris, 2019, Cengage Learning.

Una sentencia *SELECT* básica se compone de:

```
SELECT lista de columnas  
FROM lista de tablas;
```

La lista de columnas representa el listado de atributos separado por comas. Si se desea consultar todos los atributos en lugar de la lista de atributos se coloca un asterisco (\*).

```
SELECT *  
FROM lista de tablas;
```

Cuando se presentan los datos que recupera una consulta el encabezado de la columna suele ser el nombre del atributo, pero si se desea personalizar el encabezado se debe utilizar los “alias”. Un alias es un nombre alternativo para la columna.

```
-- Incluye la palabra reservada AS  
SELECT p_code as codigo, p_descript AS descripcion, p_price AS "Precio unitario"  
FROM product AS p;  
  
-- Sin incluir la palabra reservada AS  
SELECT p_code codigo, p_descript descripcion, p_price "Precio unitario"  
FROM product p;
```

La palabra reservada *AS* es opcional, si se la utiliza o no, el resultado es el mismo. Si el alias contiene espacios intermedios se lo debe encerrar entre comillas.



### Alias

Un nombre alternativo para una columna o tabla en una sentencia SQL.

Una columna calculada representa un atributo derivado y se puede o no almacenar en la base de datos. Si no se almacena se debe calcular cada vez que se requiera.

A continuación, se muestra un ejemplo de una consulta que tiene entre sus atributos un atributo derivado.

```
SELECT p_descript, p_qoh, p_price, p_qoh * p_price  
FROM product;
```

También es posible hacer cálculos con columnas de fecha, debido a que las fechas se guardan como un número de días. Por ejemplo, si la fecha de hoy en algún SGBD es el número de día 50000, el siguiente día será 50001 ( $50000 + 1$ ). En lenguaje SQL sería algo como:

```
SELECT p_descript, p_qoh, p_price, p_qoh * p_price,  
p_date + 30 as expdate  
FROM product;
```

En algunas ocasiones es necesario mostrar valores únicos, sin repeticiones, para ello utilizamos la cláusula *DISTINCT* de la siguiente manera:

```
SELECT DISTINCT v_code  
FROM product;
```

En la mayoría de los casos la información a obtener desde la base de datos se encuentra en varias tablas, por lo que se debe aplicar la operación de *JOIN* entre las tablas involucradas.

### Join Natural

Este tipo de JOIN devuelve todas las filas con valores coincidentes en las columnas coincidentes y elimina las columnas duplicadas. El natural join realiza las siguientes funciones:

- Determina los atributos comunes, que son aquellos atributos con nombres idénticos y tipos de datos compatibles.
- Selecciona solo las filas con valores comunes en los atributos comunes.
- Si no hay atributos comunes, devuelve el producto relacional de las dos tablas.

```
SELECT CUS_CODE, CUS_LNAME, INV_NUMBER, INV_DATE  
FROM CUSTOMER NATURAL JOIN INVOICE;
```

En el ejemplo anterior, la tabla Customer y la tabla Invoice tienen un atributo en común cus\_code que es utilizado por el Natural Join para realizar la operación.

### Join Using

La consulta devuelve solo las filas con valores coincidentes en la columna indicada en la cláusula *USING*, y esa columna debe ser común a las dos tablas. A continuación se muestra un ejemplo en la que la columna v\_code (solo aparece una vez) es la columna común por la cual se realiza el *JOIN*.

```
SELECT P_CODE, P_DESCRPT, V_CODE, V_NAME, V_AREACODE, V_PHONE  
FROM PRODUCT JOIN VENDOR USING (V_CODE);
```

### Join On

Este tipo de *JOIN* a diferencia de los anteriores, utiliza una condición en donde se debe especificar una igualdad de atributos para que se pueda realizar el *JOIN* que pueden o no tener el mismo nombre, pero obviamente deben tener tipos de datos comparables. A continuación un ejemplo en donde se realiza un *JOIN* entre Invoice y Line y luego entre Line y Product.

```
SELECT INVOICE.INV_NUMBER, PRODUCT.P_CODE, P_DESCRPT, LINE_UNITS, LINE_PRICE  
FROM INVOICE JOIN LINE ON INVOICE.INV_NUMBER = LINE.INV_NUMBER  
JOIN PRODUCT ON LINE.P_CODE = PRODUCT.P_CODE;
```

#### Número de condiciones de *JOIN*



Una forma de asegurarnos que la sentencia tiene los *JOIN* bien especificados es considerar el número de tablas ( $N$ ) y de acuerdo a ello la sentencia debe tener  $N-1$  condiciones de *JOIN*.

#### Cualificar las columnas con el nombre similar

Recuerde que una de las propiedades del modelo relacional se refiere a que una tabla debe tener columnas/atributos con nombre únicos. Debido a ello, cuando tenemos nombres de

columna similares, como en el ejemplo anterior los atributos inv\_number y p\_code, se los debe cualificar, es decir, anteponer el nombre de la tabla seguido de un punto. Ej. Line.p\_code

## Outer Joins

Existen tres tipos de combinación externa, izquierda (Left Outer Join), derecha (Right Outer Join) y completa (Full Outer Join). Este tipo de *Join* devuelve a más de las tuplas coincidentes aquellas que no coinciden de la parte izquierda, derecha o de ambos extremos, según se utilice el tipo de combinación externa.

En el ejemplo de la figura 6 se aplica una combinación externa a la izquierda, se muestra el código del producto, el código del proveedor y el nombre del proveedor de todos los productos e incluye los proveedores que no tienen productos coincidentes.

**Figura 6**

Ejemplo Left outer join

```
SELECT P_CODE, VENDOR.V_CODE, V_NAME  
FROM VENDOR LEFT JOIN PRODUCT  
ON VENDOR. V_CODE = PRODUCT.V_CODE;
```



P_CODE	V_CODE	V_NAME
23109-HB	21225	Bryson, Inc.
SM-18277	21225	Bryson, Inc.
null	21226	SuperLoo, Inc.
SW-23116	21231	D&E Supply
13-Q2/P2	21344	Gomez Bros.
14-Q1/L3	21344	Gomez Bros.
54778-2T	21344	Gomez Bros.
null	22567	Dome Supply
1546-QQ2	23119	Randsets Ltd.
1558-QW1	23119	Randsets Ltd.
null	24004	Brackman Bros.
2232/QTY	24288	ORDVA, Inc.
2232/QWE	24288	ORDVA, Inc.
89-WRE-Q	24288	ORDVA, Inc.
null	25443	B&K, Inc.
null	25501	Damal Supplies
11QER/31	25595	Rubicon Systems
2238/QPD	25595	Rubicon Systems
WR3/TT3	25595	Rubicon Systems

Nota. Adaptado de *Database Systems: Design, Implementation, & Management* (p. 262), por C. Coronel y S. Morris, 2019, Cengage Learning.

En el resultado se puede observar algunas tuplas que contienen en la columna p\_code valores nulos y son aquellas tuplas que no tienen coincidencias con las tuplas de la tabla Vendor.

En la *figura 7* se muestra un ejemplo donde se aplica una combinación externa a la derecha, muestra el código del producto, el código del proveedor y el nombre del proveedor para todos los productos e incluye los productos que no tienen un código de proveedor coincidente.

**Figura 7**

Ejemplo Right outer join

```
SELECT P_CODE, VENDOR.V_CODE, V_NAME  
FROM VENDOR RIGHT JOIN PRODUCT  
ON VENDOR. V_CODE = PRODUCT.V_CODE;
```



P_CODE	V_CODE	V_NAME
23114-AA	null	null
PVC23DRT	null	null
23109-HB	21225	Bryson, Inc.
SM-18277	21225	Bryson, Inc.
SW-23116	21231	D&E Supply
13-Q2/P2	21344	Gomez Bros.
14-Q1/L3	21344	Gomez Bros.
54778-2T	21344	Gomez Bros.
1546-QQ2	23119	Randsets Ltd.
1558-QW1	23119	Randsets Ltd.
2232/QTY	24288	ORDVA, Inc.
2232/QWE	24288	ORDVA, Inc.
89-WRE-Q	24288	ORDVA, Inc.
11QER/31	25595	Rubicon Systems
2238/QPD	25595	Rubicon Systems
WR3/TT3	25595	Rubicon Systems

Nota. Adaptado de *Database Systems: Design, Implementation, & Management* (p. 263), por C. Coronel y S. Morris, 2019, Cengage Learning.

En el resultado se puede observar algunas tuplas que contienen en las columnas v\_code y v\_name valores nulos y son aquellas tuplas que no tienen coincidencias con las tuplas de la tabla Product.

Finalmente, la combinación externa completa devuelve las tuplas coincidentes entre las tablas y además todas las filas con valores no coincidentes en la tabla de ambos lados. Por ejemplo, la consulta de la *Figura 8* muestra el código de producto, el código de proveedor y el nombre de proveedor de todos los productos e incluye todas las filas de productos (productos sin valores coincidentes) y todas las filas de proveedores (proveedores sin productos coincidentes).

## Figura 8

### Ejemplo Full join

```
SELECT P_CODE, VENDOR.V_CODE, V_NAME  
FROM VENDOR FULL JOIN PRODUCT  
ON VENDOR.V_CODE = PRODUCT.V_CODE;
```



P_CODE	V_CODE	V_NAME
null	21226	SuperLoo, Inc.
null	22567	Dome Supply
null	24004	Brackman Bros.
null	25443	B&K, Inc.
null	25501	Damal Supplies
11QER/31	25595	Rubicon Systems
13-Q2/P2	21344	Gomez Bros.
14-Q1/L3	21344	Gomez Bros.
1546-QQ2	23119	Randsets Ltd.
1558-QW1	23119	Randsets Ltd.
2232/QTY	24288	ORDVA, Inc.
2232/QWE	24288	ORDVA, Inc.
2238/QPD	25595	Rubicon Systems
23109-HB	21225	Bryson, Inc.
23114-AA	null	null
54778-2T	21344	Gomez Bros.
89-WRE-Q	24288	ORDVA, Inc.
PVC23DRT	null	null
SM-18277	21225	Bryson, Inc.
SW-23116	21231	D&E Supply
WR3/TT3	25595	Rubicon Systems

Nota. Adaptado de *Database Systems: Design, Implementation, & Management* (p. 263), por C. Coronel y S. Morris, 2019, Cengage Learning.

En el resultado se puede observar aquellas tuplas coincidentes entre las tablas, pero además aquellas que no coinciden tanto del lado izquierdo como del lado derecho y por ello tienen valores nulos.

## Uniones recursivas

Este tipo de uniones se presenta cuando una tabla debe unirse a sí misma en una consulta. Para hacer esto posible se debe emplear alias para las tablas, ya que, no se puede hacer referencia a una misma tabla más de una vez en una misma consulta.

Suponga la tabla que se muestra en la figura 9 que almacena datos de Empleados.

**Figura 9**  
*Tabla Empleados*

Empleados
emp_num
emp_title
emp_lname
emp_fname
emp_initial
emp_dob
emp_hire_date
emp_areacode
emp_phone
emp_mgr

Nota. Encalada, E. Morocho, J., 2022.

Con base en la tabla antes descrita se pide generar un reporte de los nombres de empleados con el nombre de sus respectivos supervisores. Para ello se debe emplear una unión recursiva, en donde debe utilizarse alias para que la tabla Empleados aparezca dos veces en la sección del *From*. La consulta quedaría de la siguiente forma:

```
SELECT E.EMP_NUM, E.EMP_LNAME, E.EMP_MGR, M.EMP_LNAME  
FROM EMP E JOIN EMP M ON E.EMP_MGR = M.EMP_NUM;
```

Al utilizar alias para la tabla Emp (E y M) el SGBD interpreta como dos tablas distintas con las cuales realizar la unión. También se debe considerar que el jefe de un empleado es también otro empleado y, por lo tanto, está registrado en la misma tabla, de ahí la utilidad de la unión recursiva.

### Lenguaje de Control de Transacciones (TCL)

Incluye las sentencias *COMMIT* y *ROLLBACK*. Permite ejecutar varias operaciones como una sola de forma indivisible y si se ejecuta correctamente se confirma a través del comando *COMMIT*, caso contrario, si algo sale mal se pueden deshacer los pasos previos aplicando el comando *ROLLBACK*.

SQL incorporado y SQL dinámico, abarcan las definiciones para incorporarlas en el código SQL en lenguajes de programación de propósito general.

## Lenguaje de Control de Datos (DCL)

Abarca las sentencias *GRANT* y *REVOKE* y permiten al administrador gestionar el acceso a los datos en la base de datos. La *tabla 3* muestra los comandos de control de transacciones y de control de datos usados en SQL.

**Tabla 3.**

Comandos de control de transacciones y de control de datos en SQL

COMMAND OR OPTION	DESCRIPTION
<b>Transaction Control Language</b>	
COMMIT	Permanently saves data changes
ROLLBACK	Restores data to its original values
<b>Data Control Language</b>	
GRANT	Gives a user permission to take a system action or access a data object
REVOKE	Removes a previously granted permission from a user

Nota. Tomado de *Database Systems: Design, Implementation, & Management* (p. 247), por C. Coronel y S. Morris, 2019, Cengage Learning.

Muy importante estos comandos de cara a garantizar la integridad y la seguridad de la base de datos.



### Actividad de aprendizaje recomendada

- Para reforzar lo aprendido, practique los principales comandos SQL. Para ello vaya a [SQL fiddle](#) y seleccione la base de datos en la que desee practicar: MySQL, Oracle o PostgreSQL. Primero escriba comandos DDL para definir la estructura de la base de datos y luego ejecute comandos DML para probar la gestión de los datos.



## Semana 4

---

Durante la presente semana ampliaremos la experiencia del uso y manejo del lenguaje SQL hacia la implementación de consultas complejas, y la generación reportes estadísticos, mediante el uso de subconsultas y agrupamiento de datos.

### SQL avanzado

Vamos a potenciar el manejo de SQL y concretamente el uso del comando *SELECT*, que es la operación que conlleva una mayor dificultad dada la variedad y complejidad de consultas que se pueden llegar a generar.

Es importante diferenciar entre dos tipos de reportes que se pueden obtener con el comando *SELECT*:

- **Listados:** aquellos donde se obtiene una línea por cada fila de la tabla o combinación de tablas fuente.
- **Informes de resumen:** también conocidos como agregaciones, informes agrupados, o reportes estadísticos. Conlleva agrupar varias filas en uno o más grupos y aplicar funciones de agregación de manera que al final se obtiene un solo valor por cada grupo.

En primera instancia nos enfocaremos justamente en los informes de resumen (agrupamientos), y luego complementaremos y potenciaremos ambos tipos de reporte con el uso de subconsultas.

### Agrupamientos

Los agrupamientos permiten generar informes de resumen, donde las filas de los datos fuente -especificados en la cláusula *FROM*- se agrupan en uno o varios conjuntos, y por cada conjunto de filas se aplica un cálculo estadístico (contar, sumar, promediar, etc.) mediante las denominadas funciones de agregación o funciones de columna.



Le invitamos a revisar el siguiente recurso, en el cual se explican las funciones de columna, que

sirven de base para realizar agregaciones en consultas SQL.

[Funciones de columna \(Quintana, 2014, pp. 45-50\)](#)

El recurso presenta una explicación con ejemplos de las funciones de columna, que se denominan así porque reciben como parámetro valores de una columna y devuelven un solo resultado y se pueden aplicar cuando se requiere calcular totales ya sea de sumatoria, conteo, promedio, entre otras.

Para realizar agrupamientos se usa la cláusula *GROUP BY* y complementariamente la cláusula *HAVING* con la cual se pueden filtrar los grupos a mostrar.



En el siguiente recurso encontrará una explicación detallada sobre el uso de la cláusula *GROUP BY* para realizar agrupamientos con SQL.

[Agrupación \(Quintana, 2014, pp. 53-63\)](#)

Como se puede evidenciar, pueden existir escenarios en los cuales no se requiera el uso de la cláusula *GROUP BY*, pero sí de funciones de agregación. Por ejemplo, suponga que deseamos obtener el salario promedio de aquellos empleados que tienen como oficio vendedor, la consulta sería similar a la que se muestra en la figura 10. Se debe tener claro que en este escenario igual estamos hablando de agrupamiento, solo que en lugar de varios grupos tenemos un solo grupo con todas las tuplas correspondientes a empleados con cargo vendedor. El resultado sería un único valor.

## Figura 10

Ejemplo de agregación SQL sin cláusula GROUP BY

```
SELECT AVG(salario)
FROM empleados
WHERE oficio = 'VENDEDOR';
```

Nota. Encalada, E. Morocho, J., 2022.

Con las cláusulas *GROUP BY* y *HAVING* se completaría la revisión de todas las cláusulas de la sentencia *SELECT*. En la figura 11 se muestra la estructura completa de dichas cláusulas y el orden en el que se deben especificar.

## Figura 11

Cláusulas de la sentencia SELECT

```
SELECT      columna(s), función de agregación
FROM        tabla o combinación de tablas
[WHERE      condición de selección de filas]
[GROUP BY  expresión de agrupamiento]
[HAVING     condición de selección de grupos]
[ORDER BY  expresión de ordenamiento];
```

Nota. Encalada, E. Morocho, J., 2022.

Recuerde que la condición que se especifica en la cláusula WHERE se aplica para filtrar los datos antes de agrupar, mientras que la condición que se especifica en la cláusula HAVING se aplica luego de agrupar, para restringir el resultado solo a los grupos que cumplan dicho predicado. El ordenamiento se realiza siempre al final, una vez generados y filtrados los grupos.



Revise el siguiente video en el cual se realiza una explicación práctica del uso de SQL para la generación de reportes estadísticos, mediante el agrupamiento de datos.

[Uso de SQL para la obtención de reportes estadísticos \(Encalada, 2022b\)](#)

Para complementar lo estudiado, en la *figura 12* se muestra un ejemplo de consulta SQL de agrupamiento con todas sus cláusulas. Específicamente, dicha consulta permite obtener la estadística de los supervisores junto al salario mínimo de los empleados que cada supervisor tiene a su cargo, y únicamente en aquellos casos donde el salario mínimo es mayor a 1500 dólares.

## Figura 12

Ejemplo consulta SQL con agrupamiento

```
SELECT supervisor, MIN(salario)
FROM empleados
WHERE supervisor IS NOT NULL
GROUP BY supervisor
HAVING MIN(salario) > 1500
ORDER BY MIN(salario) DESC;
```

Nota. Encalada, E. Morocho, J., 2022.

Aprovechemos la consulta mostrada en la *figura 12* para resumir el proceso que realiza el motor al ejecutarla:

1. Primero obtiene los datos base, recuperando en este caso las filas de la tabla empleados (cláusula FROM).
2. De los datos base filtra aquellas filas que correspondan a empleados que tienen supervisor (cláusula WHERE).
3. Sobre las filas resultantes ordena y agrupa con base en el supervisor. Es decir, arma un grupo de filas por cada valor distinto de supervisor (cláusula GROUP BY).
4. Por cada grupo determina el salario más bajo (función de agregación).
5. Luego, selecciona únicamente aquellos grupos cuyo salario mínimo es superior a 1500 (cláusula HAVING).

6. Ordena los grupos resultantes según el salario mínimo de mayor a menor (cláusula ORDER BY).
7. Finalmente, como resultado de la consulta, por cada grupo proyecta el valor del supervisor y del salario mínimo asociado (cláusula SELECT).

Es muy importante comprender este proceso para poder especificar y hacer un uso correcto de las distintas cláusulas que conforman la operación SELECT de SQL.

### Subconsultas

En una consulta SQL es posible embeber otra consulta SQL, la cual se denomina *subconsulta*. Su propósito es extraer desde otras tablas información que se requiere dentro de la consulta principal, y que normalmente no se puede obtener mediante combinaciones o JOINs, o resulta complejo hacerlo.

Por ejemplo: Supongamos que necesitamos listar los empleados cuyo salario supera el promedio.

Para dicho ejemplo, dado que el valor a comparar es desconocido, necesitaremos de una subconsulta para obtenerlo. La figura 13 nos muestra cómo sería esta consulta SQL. En este caso el motor primero ejecutará la subconsulta que calcula el salario promedio, y luego utilizará el resultado en la consulta principal.

### Figura 13

*Ejemplo de uso de subconsultas*

```
SELECT apellidos, nombres, oficio, salario
FROM empleados
WHERE salario > (SELECT AVG(salario)
                  FROM empleados);
```

Nota. Encalada, E. Morocho, J.. 2022.

Las subconsultas comúnmente se las utiliza dentro de las cláusulas WHERE o HAVING, dado que normalmente se las requiere para obtener o calcular valores a comparar en las condiciones de selección de filas o

grupos. Sin embargo, también se pueden utilizar dentro de las cláusulas SELECT o FROM.

La figura 14 nos muestra otra manera de resolver el ejemplo anterior; en este caso usando una subconsulta en la cláusula FROM. Al hacerlo de esta manera lo que el motor hará primero es ejecutar la subconsulta y luego combinar cada fila de la tabla *empleada* con el valor resultante de la subconsulta (salario promedio) agregando con en ello una columna en el resultado de la combinación (sal\_promedio) la cual se utiliza para filtrar los resultados a través de la condición de la cláusula WHERE.

#### Figura 14

Ejemplo de uso de subconsultas en cláusula FROM

```
SELECT e.apellidos, e.nombres, e.oficio, e.salario  
FROM empleados e, (SELECT AVG(salario) sal_promedio  
                      FROM empleados) x  
WHERE e.salario > x.sal_promedio;
```

Nota. Encalada, E. Morocho, J., 2022.

Cuando se utiliza subconsultas en las cláusulas WHERE o HAVING es necesario tener presente que el resultado de la subconsulta debe ser compatible con el campo y operador de la expresión condicional. Lo comprenderá mejor al revisar el recurso de aprendizaje propuesto.



Revise el siguiente recurso encontrará una amplia explicación sobre la implementación de subconsultas.

[Subconsultas \(Quintana, 2014, pp. 105-127\)](#)

Como se puede apreciar, existen subconsultas que son dependientes de la consulta principal, y, por lo tanto, se ejecutan múltiples veces en función del recorrido que realiza la consulta principal. Un ejemplo de ello se muestra en la figura 15, que corresponde a una consulta que obtiene los empleados cuyo salario es mayor al salario promedio de la oficina en la que trabaja el empleado. En este caso, la subconsulta se ejecutará por cada fila que recorre la consulta principal.

## Figura 15

Ejemplo de subconsultas con dependencia de consulta principal

```
SELECT e1.apellidos, e1.nombres, e1.oficio, e1.salario  
FROM empleados e1  
WHERE e1.salario > (SELECT AVG(salario)  
                      FROM empleados e2  
                     WHERE e2.oficina = e1.oficina);
```

Nota. Encalada, E. Morocho, J.. 2022.



### Actividades de aprendizaje recomendadas

- Ejercite el manejo de SQL avanzado revisando y resolviendo los ejercicios que plantean en los siguientes recursos de la biblioteca virtual UTPL:
  - [Ejercicios sobre agrupamientos \(Quintana, 2014, pp. 63-66\)](#)
  - [Ejercicios sobre subconsultas \(Quintana, 2014, pp. 127-134\)](#)



### Semana 5

A partir de la presente semana pasaremos a revisar otros modelos de bases de datos que se usan mucho actualmente y que tienen como propósito suplir ciertas limitantes de las bases de datos relacionales, en función de los nuevos requerimientos y retos que supone la gestión y aprovechamiento de los datos, que cada vez son más complejos, amplios, y en mayor volumen.

### Limitaciones de las bases de datos relacionales

Desde su surgimiento en la década de los 70s, las bases de datos relacionales han sido las más utilizadas, y lo siguen siendo actualmente. Su diseño sólido, los amplios controles que se pueden implementar para asegurar la integridad de los datos, y un lenguaje de consulta y gestión altamente estandarizado, han permitido que se convierta en el modelo de base de datos idóneo para informatizar la gestión operativa de un negocio, donde el control de transacciones y el aseguramiento de la consistencia los datos son fundamentales.

Pero, así como tiene sus fortalezas, el modelo relacional también tiene sus limitaciones, las cuales se han develado a partir de la necesidad cada vez más extendida de recabar, acumular, y aprovechar al máximo la información que se genera mediante múltiples medios tecnológicos, y que afecta o incide de alguna manera en el funcionamiento de una organización empresarial o social.



Para conocer en detalle las limitaciones actuales de las bases de datos relacionales, le invitamos a realizar la siguiente lectura:

[Las limitaciones de las bases de datos relacionales \(Sarasa, 2016, pp. 20-23\).](#)

La clave está sobre todo en el considerable aumento del volumen de información que en ciertos escenarios se requiere gestionar a través de una base de datos. El Big Data ha determinado la necesidad de flexibilizar las estructuras lógicas de almacenamiento de datos, y potenciar al máximo la escalabilidad horizontal mediante el aprovechamiento de la computación distribuida, que son aspectos para los cuales las bases de datos relacionales imposibilitan o restringen su aplicación.

### **Panorámica de las bases de datos NoSQL**

El término NoSQL (**Not only SQL**) denota a aquellos modelos de base de datos que suplen las principales deficiencias de las bases de datos relacionales, sobre todo en lo que corresponde a:

- Flexibilidad.
- Escalabilidad.
- Disponibilidad.

Son bases de datos que escalan horizontalmente, donde el esquema es prescindible, la redundancia es habitual, y en sus premisas no está el garantizar ACID en el marco de la gestión de transacciones.



Le invitamos a realizar la siguiente lectura a efectos de conocer con mayor detalle lo que son las bases de datos NoSQL, sus características y propiedades:

[Bases de datos NoSQL \(Sarasa, 2016, pp. 24-27\).](#)

En lo que refiere a la estructura lógica de almacenamiento, al igual que en las bases de datos relacionales, en las bases de datos NoSQL el almacenamiento es en sí de tipo estructurado, aunque con un esquema mucho menos rígido. La principal diferencia radica en la manera en que se organizan y almacenan los datos. Por ejemplo, si hablamos del almacenamiento de una factura:

- En una base de datos relacional separaríamos los datos de dicha unidad transaccional en varias tablas (cliente, factura, detalle\_factura, ítems, etc.) y para recuperar dicha unidad de información desde la vista de usuario que se muestra en el aplicativo, se requiere combinar (JOIN) dichas tablas.
- En NoSQL, simplemente se guardaría toda la factura como una unidad de información, sin necesidad de separar los datos en varias estructuras, lo cual facilita su exploración y recuperación.

Por otro lado, mención especial merece el tema de la consistencia de los datos. En las bases de datos NoSQL diríamos que se sacrifica el control de la consistencia en aras de garantizar la flexibilidad del modelo. Por ejemplo, desde el gestor de base de datos se restringe la posibilidad de gestionar transacciones o controlar atributos obligatorios.

Si usamos bases de datos NoSQL el control de la consistencia de los datos estaría únicamente en el lado de las aplicaciones o herramientas que acceden y manipulan la base de datos.

### **Tipos de bases de datos NoSQL**

Otra importante diferencia entre las bases de datos relacionales y las bases de NoSQL radica en que el esquema lógico de almacenamiento tiene algunas variantes, lo cual determina que existan distintos tipos o modelos de bases de datos NoSQL, principalmente 4:

- Clave-valor.
- Documentos.
- Columnas.
- Grafos.



En el siguiente recurso encontrará una explicación de cada tipo de base de datos NoSQL, le invitamos a revisarlo:

[Modelos de bases de datos NoSQL \(Medrano, s. f., “1.4. Modelos de datos” sección\)](#)

Como se puede evidenciar, aunque son modelos que manejan distintas estructuras para representar la información, tienen en común que son estructuras flexibles en los que es posible manejar esquemas dinámicos que facilitan la incorporación -en caliente- de nuevos elementos de información. Y además son todos modelos que facilitan implementar un procesamiento distribuido, lo cual garantiza la escalabilidad horizontal que es vital para el manejo de grandes volúmenes de datos.

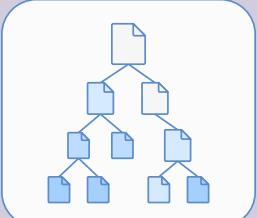
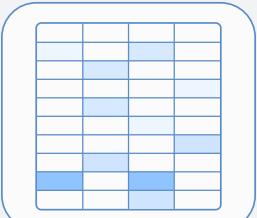
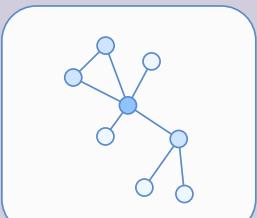
La siguiente infografía resume los principales modelos de bases de datos NoSQL.

La *tabla 4* resume los principales modelos de bases de datos NoSQL.

**Tabla 4.**

*Modelos de bases de datos NoSQL*

<p><b>Clave-valor</b></p> <div style="border: 1px solid #ccc; border-radius: 10px; padding: 10px; width: fit-content; margin-left: auto; margin-right: auto;">   <b>Key-Value Store</b> </div>	<p>Cada entidad de información se identifica por una clave única, y el conjunto de datos se almacena como una colección de pares clave-valor. Facilitan el escalamiento horizontal. Es el modelo más sencillo de bases de datos NoSQL.</p> <p>Motores: Redis, DynamoDB, Riak</p>
--	--

<b>Documentos</b>  <b>Document Store</b>	<p>Los datos se almacenan jerárquicamente en estructuras del tipo JSON o XML conocidas como documentos. Facilitan consultas avanzadas sin necesidad de JOINs.</p> <p>Motores: MongoDB, CouchDB</p>
<b>Columnas</b>  <b>Wide-Column Store</b>	<p>Guardan la información en familias de columnas. Los datos relacionados se almacenan como un conjunto de pares clave-valor anidados dentro de una sola columna. Resulta eficiente en cuanto a velocidad de lecturas.</p> <p>Motores: Cassandra, Bigtable, SimpleDB, HBase</p>
<b>Grafos</b>  <b>Graph Store</b>	<p>Se utiliza la estructura de grafo para representar y almacenar los datos, como propiedades de nodo y arista, donde las aristas son vitales, ya que permiten representar la interrelación entre los nodos.</p> <p>Motores: Neo4j, HyperGraphDB, FlockDB</p>

Nota. Adaptado de *Soluciones relacionales y datos NoSQL*, por Microsoft, 2022  
[Enlace web](#)

Actualmente, en NoSQL los modelos más conocidos y usados corresponden a las bases de datos de documentos y bases de datos clave-valor. Aunque las bases de datos relacionales son las que continúan predominando. Puede consultar uno de los rankings de motores de bases de datos más conocidos, que es [DB-Engines Ranking](#), el cual los mide según su popularidad.



Para ampliar la comprensión de los distintos tipos de bases de datos NoSQL revise el siguiente recurso:

Modelos de bases de datos NoSQL orientados hacia agregados (Sarasa, 2016, pp. 28-33)

En conclusión, cada tipo de base de datos NoSQL maneja estructuras lógicas de almacenamiento diferentes, pero todas conservan las características y beneficios de una base de datos NoSQL, sobre todo en términos de permitir un modelo flexible, una arquitectura escalable, y una alta disponibilidad.



### Actividades de aprendizaje recomendadas

- Investigue más acerca del teorema de CAP, el cual ayuda a dimensionar mejor las diferencias entre las bases de datos relacionales y las bases de datos NoSQL.
- Haga su propio análisis comparativo entre las bases de datos relacionales vs las bases de datos NoSQL. Determine cuando se deberían usar bases de datos relacionales, y cuando se deben usar bases de datos NoSQL.
- Revise el siguiente recurso de la biblioteca virtual UTPL: [Introduction to NoSQL Systems \(Elmasri y Navathe, 2016, pp. 884-889\)](#); y a partir de ello, y con base en lo estudiado en esta semana, resuelva las cuestiones planteadas al final de ese capítulo en la página 909. Específicamente resuelva los ítems 24.1 a 24.6.



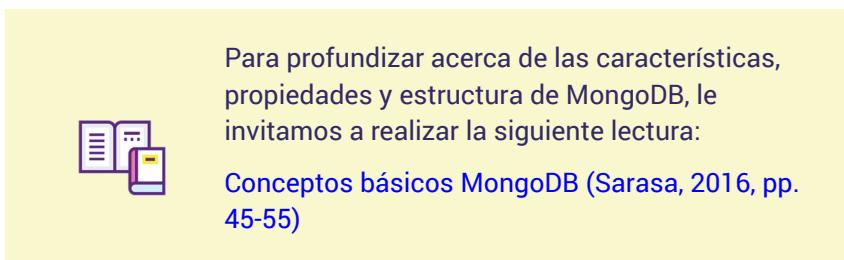
### Semana 6

Durante la presente semana aprenderemos a realizar una manipulación básica de una base de datos NoSQL de tipo documental. Para ello usaremos como gestor de bases de datos NoSQL el motor MongoDB, y como software de administración Studio 3T Free. Se recomienda instalar ambas herramientas:

- MongoDB Community Server
- Studio 3T Free

## Manipulación de datos en bases de datos documentales

MongoDB es un SGBD multiplataforma orientado a documentos de esquema libre. Para el almacenamiento y gestión de los datos utiliza un formato llamado BSON (Binary JSON), que es una versión modificada y enriquecida de JSON orientada a optimizar el rendimiento de consultas y agregaciones.



Como se ha visto, la estructura lógica de almacenamiento que maneja MongoDB es la que se muestra en la figura 16. Significa que en un servidor MongoDB podemos crear una o más bases de datos, luego, dentro de una base de datos podemos agregar una o más colecciones, y a su vez cada colección almacenará documentos en formato JSON. Y cada documento alberga toda la información asociada a cada instancia de un determinado tipo de entidad.

**Figura 16**  
*Estructura de almacenamiento MongoDB*



Nota. Encalada, E. Morocho, J., 2022.

Y si comparamos esta estructura con las bases de datos relacionales, la analogía sería la que se muestra en la tabla 5. Sin embargo, no se trata de

una equivalencia estricta, ya que, en una base de datos documental, por ejemplo, una colección puede contener información de varias tablas.

**Tabla 5.**

*Terminología modelo relacional vs MongoDB*

Base de datos relacional	MongoDB
Base de datos / Esquema	Base de datos
Tabla	Colección
Registro / Fila	Documento
Campo / Columna	Atributo / Campo
Rowid	Atributo <code>_id</code>

Nota. Encalada, E. Morocho, J.. 2022.

### Comandos básicos

Como todo sistema gestor de base de datos, el motor MongoDB permite acceder a una consola para ejecución de comandos. Por ejemplo: para abrir la consola en Windows, iremos a la ventana de comandos (símbolo del sistema), navegamos hasta la carpeta `bin` dentro del directorio donde instaló el servidor MongoDB (por ejemplo `C:\Program Files\MongoDB\Server\4.2\bin`), y ejecutamos el comando `mongo`. O, desde explorador de archivos vamos a dicha carpeta y ejecutamos `mongo.exe`. En ese momento estaremos listos para lanzar órdenes al sistema gestor de MongoDB.



Para conocer los principales comandos y operaciones que se pueden ejecutar desde la Shell de MongoDB, revise el siguiente recurso:  
[Comandos básicos Shell MongoDB \(Sarasa, 2016, pp. 56-65\)](#)

Destaquemos algunos de los comandos básicos más importantes:

- Para ver las bases de datos disponibles en nuestro servidor MongoDB, usamos `show dbs`.

- Podemos seleccionar una base de datos escribiendo: **use mydb**, de forma que, si no existe la base de datos **mydb**, será creada automáticamente. No es necesario una declaración explícita de creación de la base de datos.
- Mediante el comando **db** obtenemos el nombre de la base de datos a la que estamos conectados.
- Se elimina una base de datos mediante **db.dropDatabase()**.
- Si necesitamos acudir a la ayuda del sistema para consultar alguna funcionalidad podemos hacerlo mediante **help**, o a la ayuda de una determinada función mediante **nomrefuncion.help()**.

## Colecciones

Una colección en MongoDB sería más o menos equivalente a lo que conocemos como tabla en el modelo relacional, aunque como ya se mencionó antes, no es exactamente igual. En una sola colección se puede almacenar información que normalmente en una base de datos relacional se almacenaría en varias tablas.

Por ejemplo, consideremos el almacenamiento de información acerca de personas en una base de datos relacional y en MongoDB.

### Figura 17

*Almacenamiento de información sobre personas en una base de datos relacional*

Tabla: Personas			Tabla: Teléfonos	
Cédula	Nombre	Sueldo	Cédula	Teléfono
1111111111	Juan	1200.00	1111111111	0978786767
2222222222	María	1230.34	2222222222	027428483
3333333333	Pedro		2222222222	0983764552
			3333333333	0498263541

Tabla: Títulos		
Cédula	Denominación	Año
2222222222	Arquitecto	1990
3333333333	Ingeniero en informática	2000
3333333333	Máster en seguridad informática	2015

Nota. Encalada, E. Morocho, J., 2022.

Para el mismo ejemplo, la figura 18 muestra cómo se almacenaría esa misma información en una base de datos NoSQL documental MongoDB. En este caso no hace falta definir varias colecciones, dado que estamos hablando de una misma entidad de información (persona). La flexibilidad de los modelos NoSQL permiten manejar un esquema flexible y robusto, en este caso, como se ve, usando un esquema basado en documentos JSON, donde cada documento corresponde a la información de una persona. La flexibilidad del modelo facilita, por ejemplo, la implementación de atributos multivaluados; si un atributo es multivaluado entonces simplemente se define un arreglo de valores como el caso del teléfono, o incluso un arreglo de documentos, como el caso de atributo *título*. Igualmente, se puede agregar atributos nuevos (como el *email*) sin restricciones. En MongoDB no hay un esquema rígido.

## Figura 18

Almacenamiento de información sobre personas en MongoDB

## Colección PERSONAS

```
[{  
    "cedula": "1111111111",  
    "nombre": "juan",  
    "telefono": [  
        "0978786767",  
        "0728938293"  
    ],  
    "sueldo": 1200  
}, {  
    "cedula": "2222222222",  
    "nombre": "maria",  
    "telefono": [  
        "027428483",  
        "0983764552"  
    ],  
    "sueldo": 1230.34,  
    "titulo": {  
        "denominacion": "Arquitecto",  
        "anio": 1990  
    }  
, {  
    "cedula": "3333333333",  
    "nombre": "pedro",  
    "telefono": "0498263541",  
    "titulo": [{  
        "denominacion": "Ingeniero en informática",  
        "anio": 2000  
    }, {  
        "denominacion": "Master en seguridad informática",  
        "anio": 2015  
    }],  
    "email": "pedro@mail.com"  
}]
```

Nota. Encalada, E. Morocho, J.. 2022.

A continuación, algunas consideraciones importantes acerca de las colecciones:

- Dentro de una base de datos, podemos agregar una o varias colecciones.
- Podemos ver las colecciones existentes mediante el comando **show collections**.
- Para crear una colección, no es necesario definir explícitamente el nombre de la colección y su estructura o esquema; se generará y actualizará automáticamente conforme vayamos insertando documentos.

## Inserción de documentos

Pasemos a revisar como se manipulan colecciones en MongoDB en términos de inserciones, actualizaciones y borrado de documentos. A partir de este punto usted puede pasar a usar una herramienta gráfica más especializada para acceder al servicio MongoDB. En nuestro caso se ha sugerido el uso de la herramienta [Studio 3T Free](#). La URI de la cadena de conexión que se usa por defecto para acceder al servicio MongoDB en una implementación standalone es la siguiente: **mongodb://localhost:27017**.



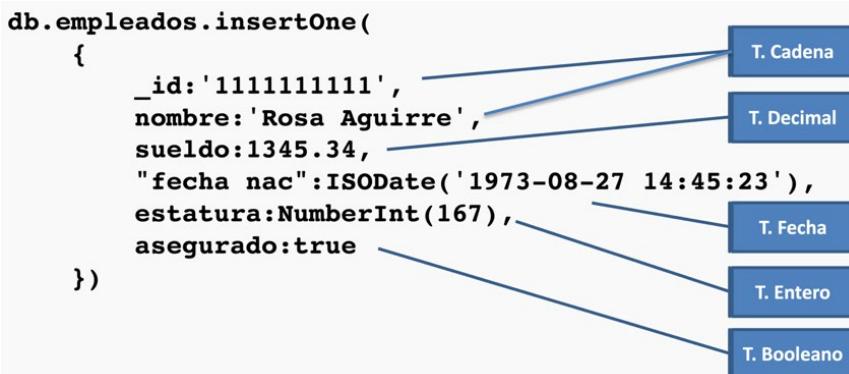
Revise el siguiente recurso para conocer los comandos que permiten manipular el contenido de una colección en MongoDB:

[Operaciones CRUD \(Sarasa, 2016, pp. 69-80\)](#)

En versiones actuales, MongoDB ha implementado dos comandos para insertar documentos, el comando **db.collection.insertOne()** y el comando **db.collection.insertMany()**, que permiten insertar uno o varios documentos respectivamente. En la *figura 19* y *figura 20* se muestran ejemplos de uso de estos comandos. Se muestra además la especificación de los tipos de datos más comunes que maneja MongoDB, que son: cadena, numérico decimal, numérico entero, fecha y booleano.

### Figura 19

Ejemplo comando MongoDB para insertar un documento



Nota. Encalada, E. Morocho, J., 2022.

## Figura 20

Ejemplo comando MongoDB para insertar varios documentos

```
db.empleados.insertMany(  
  [{  
    _id: '222222222222',  
    nombre: 'Pedro Salazar',  
    sueldo: 1200,  
    "fecha nac": ISODate('1980-05-11 10:34:32'),  
    estatura: NumberInt(180),  
    asegurado: false  
  }, {  
    _id: '333333333333',  
    nombre: 'Ana Sanchez',  
    sueldo: 986,  
    "fecha nac": ISODate('1983-11-04 18:04:00'),  
    estatura: NumberInt(171),  
    asegurado: true  
  }])
```

Nota. Encalada, E. Morocho, J., 2022.

Especial atención merece el atributo `_id`. Todo documento tendrá siempre un `_id` único, si no lo especificamos de manera explícita al realizar la inserción, el motor asignará uno automáticamente.

## Consulta y recuperación de datos

Al igual que en SQL, la consulta constituye la operación más compleja en MongoDB, y a diferencia de SQL, se manejan comandos distintos para generar listados, e informes de resumen. Cuando se trata de listados usaremos el método `db.collection.find()` y cuando se trata de agrupamientos el más usado suele ser el método `db.collection.aggregate()`. Nos centraremos aquí sobre todo en el uso del método `find()`.



Revise el siguiente recurso, el cual le permitirá comprender como se realiza la construcción y ejecución de consultas usando el método `find()`:

[Consultas en MongoDB \(Medrano, s. f., “2.4. Consultas” sección\)](#)

El método **find()** nos permite entonces recuperar todos o una parte de los documentos de una colección. Se le pueden pasar dos parámetros, el primero, un documento con las especificaciones de selección, y el segundo, un documento con las especificaciones de proyección. En general, en todo método MongoDB, los argumentos se establecen usando el formato de documento JSON.

## Figura 21

Ejemplo consulta MongoDB a través del método *find()*

```
db.encuestas.find(
  {
    $and: [
      {
        provincia: "Loja",
        estado_civil: "Soltero(a)",
        condicion_actividad: { $ne: "Empleo Adecuado/Pleno" }
      }
    ],
    {
      codigo: 1,
      area: 1,
      genero: 1,
      edad: 1
    }
  }
).sort({ genero: 1, edad: -1 })
```

El diagrama ilustra la estructura de un código MongoDB para el método `find()`. Se divide en tres secciones principales:

- Especificaciones de selección:** Representada por un cuadro azul que engloba el primer argumento, que incluye un operador `$and` y una lista de filtros.
- Especificaciones de proyección:** Representada por un cuadro azul que engloba el segundo argumento, que incluye los campos a devolver: `codigo: 1, area: 1, genero: 1, edad: 1`.
- Especificaciones de ordenamiento:** Representada por un cuadro azul que engloba el tercer argumento, que incluye la cláusula `.sort({ genero: 1, edad: -1 })`.

Nota. Encalada, E. Morocho, J., 2022.

La figura 21 nos muestra un ejemplo del uso del método **find()**. En este caso asumimos que tenemos una colección `encuestas` que contiene el detalle de encuestas socioeconómicas aplicadas a un grupo poblacional. El objetivo es (1) obtener los documentos que correspondan a encuestas aplicadas en la provincia de Loja, a personas solteras, que no tengan empleo adecuado; y (2) de ellos mostrar únicamente el código, área (urbano/rural), género y edad del encuestado. Lo primero, las condiciones de selección o filtrado de documentos lo especificamos en el primer argumento/documento, lo segundo, los atributos a mostrar, lo especificamos en el segundo documento que pasamos al método **find()**. Al final se organizan y muestran los documentos tal que aparezcan ordenados alfabéticamente de A-Z por género, y dentro de cada género, ordenados descendente por la edad del encuestado.



En el siguiente recurso encontrará varios ejemplos de consultas sobre una base de datos, en los que para cada ejemplo se contrasta la solución SQL vs la solución MongoDB. Le permitirá comprender mejor el lenguaje MongoDB a partir del ya estudiado lenguaje SQL:

#### Ejemplos de comandos de consulta en MongoDB

La gran diferencia entre los lenguajes SQL y MongoDB determina también el cambio de paradigma al que estamos asistiendo con las bases de datos NoSQL.

### Material complementario

Opcionalmente, y a efectos de reforzar la comprensión de los temas tratados, le recomendamos el siguiente recurso:

- [Document-Based NOSQL Systems and MongoDB \(Elmasri y Navathe, 2016, pp. 890-895\)](#): le permitirá reforzar la comprensión del modelo de bases de datos documentales con base en las características de MongoDB. Preste especial atención a los dos patrones de distribución que permite la versión comercial de este gestor de bases de datos NoSQL.



### Actividades de aprendizaje recomendadas

- Revise y resuelva ejercicios desarrollados y propuestos en el siguiente recurso educativo abierto: [Ejercicio MongoDB](#)
- Complemente lo estudiado revisando otras fuentes de información con ejemplos y ejercicios sobre el uso de comandos para manipulación y exploración de colecciones y documentos en bases de datos MongoDB. Se sugieren aquí algunos recursos complementarios:
  - Curso recomendado: [Bases de datos NoSQL](#)
  - Operadores para consulta y proyección: [Operadores de consulta](#)
  - Referencia general del lenguaje: [Métodos de shell](#)

- Analice como se podría realizar una migración de datos desde una base de datos relacional a una base de datos NoSQL MongoDB, es decir, ¿cómo haríamos para pasar de forma automática los datos desde un conjunto de tablas a una colección MongoDB? Tomando en cuenta la combinación de varias tablas, el tratamiento de valores nulos, y el manejo de atributos multivaluados. Considere por ejemplo el caso mostrado en la *figura 17* y *figura 18*.



## Semana 7

---

Corresponde ahora revisar algunos conceptos básicos en torno a otros de los modelos de bases de datos de NoSQL, conocidos como bases de datos de grafos o bases de datos orientadas a grafos.

### Bases de datos de grafos

Es un modelo de base de datos NoSQL basado en la teoría de los grafos que almacena datos ricos en relaciones como una colección de nodos y aristas. El modelado y el almacenamiento de datos sobre relaciones es el objetivo de las bases de datos de grafos.

La teoría de grafos es un campo de estudio muy amplio y fundamentado desde mucho tiempo atrás. Por ello, a lo largo de todos estos años se han creado varios algoritmos y aplicaciones que han ayudado a las bases de datos de grafos a que evolucionen muy rápidamente.

El interés en las bases de datos de grafos nace por las inmensas cantidades de datos que se producen y las relaciones que existen entre esos datos. Un ejemplo de las relaciones entre datos se da en aplicaciones como Facebook, Twitter o Instagram.

Un nodo representa una instancia específica de aquello que deseamos almacenar en la base de datos de grafos. La relación entre los nodos se denomina arista y puede ser bidireccional. Además, las aristas pueden tener propiedades.

Las propiedades son equivalentes a los atributos en una entidad, son los datos que se requieren almacenar de cada nodo. Estas propiedades podrían variar entre nodos a diferencia de un modelo relacional en el que los atributos tienen una estructura definida.

Cuando se requiere consultar la base de datos de grafos, se hace un recorrido por los nodos y de hecho se puede hacer consultas para encontrar el camino más corto y el grado de conexión.

En conclusión, las principales características que se pueden aprovechar de las bases de datos de grafos son:

- No obligan a que los datos se ajusten a estructuras predefinidas.
- Están optimizadas para proporcionar velocidad de procesamiento.
- No admiten el lenguaje de consulta SQL.



Para complementar lo explicado le invitamos a revisar el siguiente recurso, en el cual se realiza una interesante caracterización de las bases de datos orientadas a grafos, y además se establece una comparativa con las bases de datos relacionales.

[Graph databases: an overview \(Domenjoud, 2012\)](#)

Como se puede apreciar, las bases de datos relacionales tienen limitaciones en cuanto al volumen (size) y a la complejidad de la representación de los datos (complexity); y es en este último aspecto donde surge el principal potencial de las bases de datos orientadas a grafos, ya que facilitan y enfatizan la representación de las relaciones existentes entre distintos objetos de información.

Existen varios sistemas gestores de bases de datos de grafos, los más usados actualmente son:

- Neo4j
- OrientDB
- ArangoDB
- AllegroGraph
- HyperGraphDB
- InfiniteGraph

Las bases de datos de grafos están concebidas para aquellos escenarios donde es propicio el uso de una base de datos NoSQL en general y donde además las relaciones entre los elementos de datos son de vital importancia. Aplican muy bien para contextos como:

- Representación **estructuras jerárquicas**, en múltiples ámbitos.
- **Sistemas de recomendación**, donde por ejemplo la relación de proximidad entre productos relacionados facilita generar mejores sugerencias a los usuarios.
- **Rutas logísticas**, haciendo más eficiente la búsqueda del camino más corto.
- Representación de **redes y relaciones sociales**, que ayudan mucho en diferentes ámbitos, como el social, de gobierno, de seguridad, de investigación, y otros.
- **Gestión del conocimiento**, donde es muy importante establecer las relaciones entre distintos objetos, a efectos de mejorar y hacer más eficiente la búsqueda y transferencia de ese conocimiento.

## Web semántica y datos enlazados abiertos

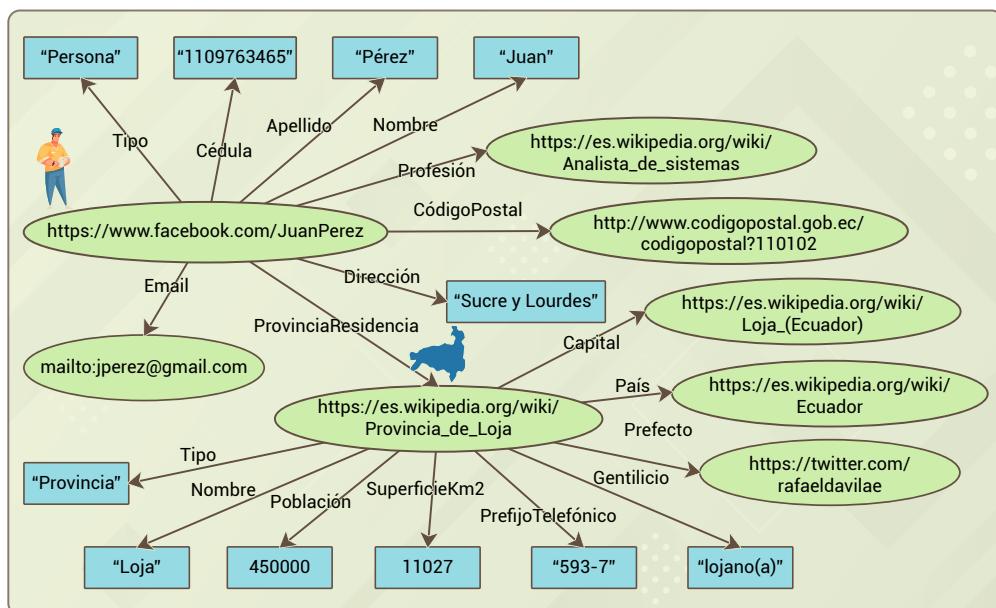
Otro escenario donde aplica muy bien el uso de bases de datos de grafos es cuando se trata caracterizar y relacionar los recursos publicados en la Web, en el marco de lo que se conoce como Web Semántica. En este caso, los recursos publicados (páginas, posts, videos, documentos, etc.) se identifican mediante una URI, y constituyen los nodos del grafo, y el predicado que define la relación entre esos recursos estaría representado en cada arista que une los nodos de los recursos asociados. Y en el mismo esquema también se pueden representar características que no corresponden a recursos en sí, sino a datos específicos.

La figura 22 ilustra un ejemplo de lo mencionado anteriormente. Primero, diferenciamos dos tipos de nodos, unos dibujados mediante elipses que representan objetos o recursos web, y otros graficados mediante rectángulos que representan valores literales. El objetivo como decíamos es caracterizar y asociar los diferentes recursos. Por ejemplo, la persona Juan Pérez, puede ser identificado mediante un recurso web a través de la URI de su cuenta de Facebook, y puede ser caracterizado mediante la

relación con otros nodos, en este caso las aristas identifican el predicado que describe a cada relación. Juan Pérez por ejemplo está conectado al nodo "Persona" mediante la relación "Tipo", significa que Juan Pérez es de tipo persona. Así mismo, Juan Pérez está conectado al nodo <[https://es.wikipedia.org/wiki/Provincia\\_de\\_Loja](https://es.wikipedia.org/wiki/Provincia_de_Loja)> mediante la relación "ProvinciaResidencia", lo que indica que él reside en la provincia de Loja. Y la provincia de Loja a su vez es otro objeto que puede ser caracterizado y conectado a otros recursos.

**Figura 22**

Ejemplo de grafo para representar información asociada a recursos web



Nota. Tomado de Bases de Datos Semánticas (p. 31), por A. E. Encalada, 2020 ([Enlace web](#))

Para la implementación de este tipo de esquemas de información, como el que se muestra en la figura 22, se usa un modelo conocido como **RDF** (Resource Description Framework), el cual es una implementación particular de bases de datos orientadas a grafos. En este modelo se definen tres elementos: sujeto, predicado, y objeto; donde el primero corresponde al nodo origen, el segundo corresponde a la arista, y el tercero corresponde al nodo destino. La figura 23 muestra cómo se representaría el grafo de la figura 22 en RDF.

## Figura 23

Ejemplo de almacén RDF para representar información asociada a recursos web

Sujeto	Predicado	Objeto
<a href="https://www.facebook.com/JuanPerez">https://www.facebook.com/JuanPerez</a>	Tipo	"Persona"
<a href="https://www.facebook.com/JuanPerez">https://www.facebook.com/JuanPerez</a>	Cedula	"1109763465"
<a href="https://www.facebook.com/JuanPerez">https://www.facebook.com/JuanPerez</a>	Apellido	"Pérez"
<a href="https://www.facebook.com/JuanPerez">https://www.facebook.com/JuanPerez</a>	Nombre	"Juan"
<a href="https://www.facebook.com/JuanPerez">https://www.facebook.com/JuanPerez</a>	Profesion	<a href="https://es.wikipedia.org/wiki/Analista_de_sistemas">https://es.wikipedia.org/wiki/Analista_de_sistemas</a>
<a href="https://www.facebook.com/JuanPerez">https://www.facebook.com/JuanPerez</a>	CodigoPostal	<a href="http://www.codigopostal.gob.ec/codigopostal7110102">http://www.codigopostal.gob.ec/codigopostal7110102</a>
<a href="https://www.facebook.com/JuanPerez">https://www.facebook.com/JuanPerez</a>	Direccion	"Sacre y Lourdes"
<a href="https://www.facebook.com/JuanPerez">https://www.facebook.com/JuanPerez</a>	ProvinciaResidencia	<a href="https://es.wikipedia.org/wiki/Provincia_de_Loja">https://es.wikipedia.org/wiki/Provincia_de_Loja</a>
<a href="https://www.facebook.com/JuanPerez">https://www.facebook.com/JuanPerez</a>	Email	<a href="mailto:jperez@gmail.com">mailto:jperez@gmail.com</a>
<a href="https://es.wikipedia.org/wiki/Provincia_de_Loja">https://es.wikipedia.org/wiki/Provincia_de_Loja</a>	Tipo	"Provincia"
<a href="https://es.wikipedia.org/wiki/Provincia_de_Loja">https://es.wikipedia.org/wiki/Provincia_de_Loja</a>	Nombre	"Loja"
<a href="https://es.wikipedia.org/wiki/Provincia_de_Loja">https://es.wikipedia.org/wiki/Provincia_de_Loja</a>	Poblacion	450000
<a href="https://es.wikipedia.org/wiki/Provincia_de_Loja">https://es.wikipedia.org/wiki/Provincia_de_Loja</a>	SuperficieKm2	11027
<a href="https://es.wikipedia.org/wiki/Provincia_de_Loja">https://es.wikipedia.org/wiki/Provincia_de_Loja</a>	PrefijoTelefonico	"593-7"
<a href="https://es.wikipedia.org/wiki/Provincia_de_Loja">https://es.wikipedia.org/wiki/Provincia_de_Loja</a>	Gentilicio	"lojano(a)"
<a href="https://es.wikipedia.org/wiki/Provincia_de_Loja">https://es.wikipedia.org/wiki/Provincia_de_Loja</a>	Prefecto	<a href="https://twitter.com/rafaeldavila">@rafaeldavila</a>
<a href="https://es.wikipedia.org/wiki/Provincia_de_Loja">https://es.wikipedia.org/wiki/Provincia_de_Loja</a>	Pais	<a href="https://es.wikipedia.org/wiki/Ecuador">https://es.wikipedia.org/wiki/Ecuador</a>
<a href="https://es.wikipedia.org/wiki/Provincia_de_Loja">https://es.wikipedia.org/wiki/Provincia_de_Loja</a>	Capital	<a href="https://es.wikipedia.org/wiki/Loja_(Ecuador)">https://es.wikipedia.org/wiki/Loja_(Ecuador)</a>

Nota. Tomado de *Bases de Datos Semánticas* (p. 32), por A. E. Encalada, 2020  
(Enlace web)

El modelo RDF es también conocido como **triplesstore** en referencia a los tres elementos que lo componen, es decir, es un almacén de tripletas.

Y para la exploración de la información contenida en un almacén RDF se usa un lenguaje conocido como SPARQL, el cual permite navegar entre nodos del grafo.



Revise le siguiente recurso para comprender a profundidad el ecosistema definido en torno a la web semántica o web de datos, que incluye la explicación de lo que se conoce como Linked Open Data, además del modelo RDF y del lenguaje SPARQL.

[Introducción a los Datos abiertos enlazados \(Blaney, 2017\)](#)

En conclusión, las bases de datos de grafos actualmente tienen muchas aplicaciones, ya que proveen una mayor versatilidad respecto a los otros modelos de bases de datos NoSQL, y al dar un mayor énfasis a las relaciones que existen entre distintas entidades, mejoran notablemente el rendimiento de las consultas.

## Material complementario

Opcionalmente, y a efectos de reforzar la comprensión de los temas tratados, le recomendamos los siguientes recursos:

- [NOSQL Graph Databases and Neo4j \(Elmasri y Navathe, 2016, pp. 903-908\)](#): le permitirá comprender mejor las características y propósito de las bases de datos orientadas a grafos a partir del análisis del motor Neo4j. Note la gran diferencia en el lenguaje de consulta que usa Neo4j, respecto a que implementa MongoDB. En este caso el lenguaje está orientado a poder recorrer el grafo mediante las conexiones entre los nodos.
- Video denominado "[¿Por qué Neo4j?](#)", disponible en el canal Neo4j de [YouTube](#): una breve, pero didáctica explicación de las ventajas de una base de datos orientada a grafos, concretamente Neo4j que es uno de los motores más usados actualmente a nivel mundial.



## Actividades de aprendizaje recomendadas

- Investigue más a profundidad acerca del modelo RDF y del lenguaje SPARQL, mediante búsqueda de recursos web, en los que se desarrollen ejemplos que pueda usted aplicar accediendo al endpoints abiertos como la DBpedia u otro.



## Semana 8

---

### Preparación para la primera evaluación bimestral

Llegamos al final del primer bimestre, esperamos que su experiencia de aprendizaje haya sido muy positiva y gratificante.

Esta semana corresponde dedicarla a revisar lo aprendido durante el primer bimestre como preparación para rendir el examen del primer bimestre. Le sugerimos hacerlo de manera sistemática con base en las siguientes recomendaciones:

1. Revise los contenidos estudiados durante el primer bimestre (unidad 1), en el orden que marca la ruta de aprendizaje del curso. En su repaso incluya el material complementario provisto por el tutor.
2. Complemente el repaso de los temas de cada semana con la revisión de las actividades de aprendizaje desarrolladas (prácticas, talleres, foros, y cuestionarios).
3. Refuerce el repaso de aquellos temas que usted considera le han resultado más difíciles de asimilar. Tome nota de aquellas cuestiones en las que necesite una mayor retroalimentación.
4. Aproveche los espacios de tutorías para compartir sus inquietudes con el tutor y recibir la retroalimentación necesaria.

Las preguntas del examen estarán orientadas a evaluar su capacidad de comprensión, aplicación, y análisis en torno a los temas estudiados.

¡Éxitos en su evaluación!



## Segundo bimestre

### Resultado de aprendizaje 2

- Diseña esquemas de bases de datos utilizando modelos relacionales y no relacionales e híbridos.

### Contenidos, recursos y actividades de aprendizaje



#### Semana 9

#### Unidad 2. Diseño de bases de datos



En esta semana iniciaremos el estudio del proceso de diseño de una base de datos, que requiere de un proceso metodológico para cubrir los requerimientos de datos de la organización para la cual se está construyendo el modelado.

#### Ciclo de vida de una base de datos

El desarrollo de una base de datos pasa por un proceso que se conoce como ciclo de vida. Empieza por analizar las necesidades de datos, continúa en la tarea de diseño (conceptual, lógico y físico) para su posterior implementación y finalmente la etapa de monitoreo y mantenimiento permanente, tal como lo muestra la figura 24.

**Figura 24**

*Etapas del ciclo de vida de una base de datos*



Nota. Tomado de Capítulo 3 Ciclo de Vida, por P. Araneda, 2020 ([Enlace web](#)). CC BY NC SA.

### Metodología de diseño de una base de datos

El proceso de diseño de una base de datos debe ser conducido por un proceso formal denominado metodología. Al igual que si se tratara del desarrollo de software, en donde tenemos varias opciones de metodologías a utilizar, también en el diseño de una base de datos debemos guarnos por un proceso metodológico. Una metodología nos proporciona procedimientos, técnicas, herramientas y guías para documentar todo el proceso de diseño de una base de datos con el objetivo de que el producto final cumpla con lo esperado por el cliente.

La metodología que se propone adoptar tiene tres fases bien definidas: diseño conceptual, diseño lógico y diseño físico. Se inicia por la definición de los requerimientos de datos que permiten construir el diseño conceptual a través del modelo entidad-relación (ER). Luego se pasa al diseño lógico a través de la aplicación de ciertas reglas que dan como resultado el modelo relacional, aplicando antes de finalizar esta fase el proceso de normalización. Finalmente, se ejecuta el diseño físico en donde se materializa el diseño relacional normalizado.



Para complementar lo explicado, revise el siguiente video, en el cual se detalla el alcance de las 3 etapas del diseño.

El recurso de video muestra una descripción detallada de las fases para el diseño de una base de datos relacional, que empieza en la revisión de los requerimientos que sirven de entrada para el diseño conceptual, luego pasamos al diseño lógico y finalmente la fase de diseño físico para materializar la base de datos.

### Modelo Entidad-Relación ER

El modelo entidad-relación (ER) es ampliamente utilizado en el diseño de bases de datos. Permite crear una representación gráfica de los datos requeridos en la base de datos, lo que facilita el entendimiento por usuarios no técnicos. El modelo ER se basa en las entidades y las relaciones entre ellas, de ahí su nombre.

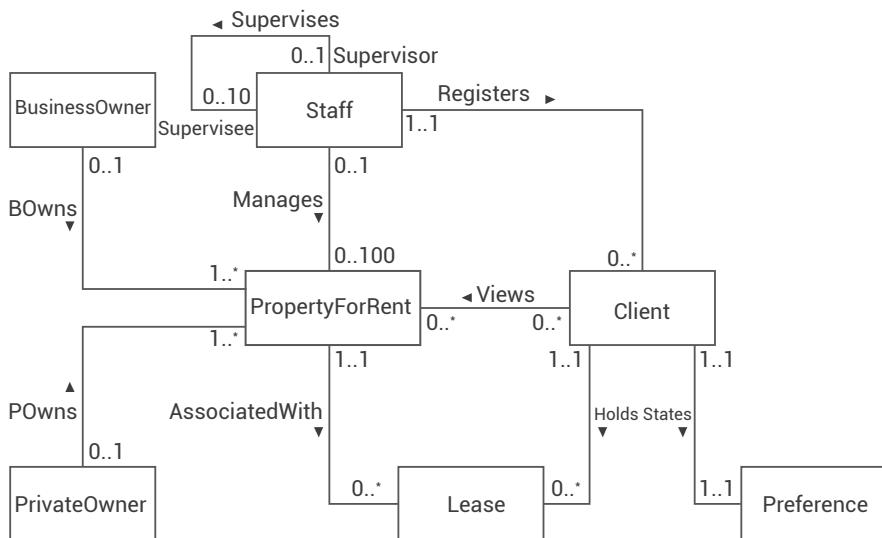
A menudo, las relaciones son binarias, pero debemos tener cuidado con relaciones complejas en donde intervienen más de dos tipos de entidad. Las relaciones deben tener un nombre que las identifique y que además permitan entender lo que están representando.

Las relaciones tienen cardinalidad, es decir, definen para cada tupla de una relación con cuántas tuplas de la otra relación tienen correspondencia. Entonces, las relaciones se clasifican en los siguientes tipos: uno-a-uno (1:1), uno-a-muchos (1:M) y muchos-a-muchos (M:N o M:M).

La figura 25 muestra un ejemplo de diagrama ER en el que se muestran los tipos de entidad y sus tipos de relación. Por ejemplo, entre *Client* y *Preference* tenemos un tipo de relación 1:1, debido a que un cliente tiene una y solo una preferencia en cuanto al estado/provincia donde habita y la preferencia es solamente de un cliente a la vez. Otro ejemplo tenemos entre *Client* y *Lease*, que tienen un tipo de relación 1:M o 1:\*, en donde un cliente no puede tener ningún contrato de arrendamiento, por eso el 0, o puede tener muchos contratos de arrendamiento, y viceversa un contrato de arrendamiento solamente está relacionado un uno y solo un cliente. Y un ejemplo de una relación M:M se da entre *Client* y *PropertyForRent*, en donde un cliente puede realizar ninguna o varias visitas a una propiedad en renta y así mismo una propiedad en renta puede recibir ninguna o varias visitas.

**Figura 25**

Ejemplo de diagrama ER

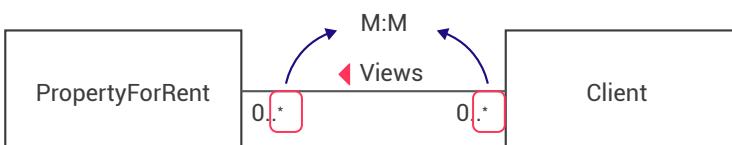


Nota. Tomado de *Sistemas de bases de datos. Un enfoque práctico para diseño, implementación y gestión* (p. 407), por T. M. Connolly y C. E. Begg, 2005, Pearson Educación.

Para pasar del diseño conceptual al diseño físico se requiere conocer la cardinalidad del tipo de relación, pero hasta ahora solamente hemos definido las cardinalidades en los extremos de las relaciones. Para encontrar la cardinalidad de la relación, tomamos los valores máximos de cada extremo, es decir los valores a la derecha de los dos puntos, tal como lo muestra la figura 26.

**Figura 26**

Cardinalidad del tipo de relación



Nota. Adaptado de *Sistemas de bases de datos. Un enfoque práctico para diseño, implementación y gestión* (p. 407), por T. M. Connolly y C. E. Begg, 2005, Pearson Educación.

El modelo ER utiliza símbolos que tienen un significado y con los cuales representa los modelos de datos. A continuación, la figura 27 muestra la notación de base de datos de Chen.

**Figura 27**

Notación de base de datos de Chen

Símbolo	Significado
	Entidad
	Entidad débil
	Relación
	Atributo
	Atributo clave
	Atributo multivaluado
	Atributo compuesto
	Atributo derivado
	Participación total de E2 en R
	Cardinalidad 1:N para E1:E2 en R
	Restricción estructural (min, max) sobre la participación de E en R

Nota. Tomado de *Guía didáctica de Fundamentos de Bases de Datos* (p. 120), por J. Morocho y A. Romero, 2019, Editorial de la Universidad Técnica Particular de Loja.



En el siguiente recurso podrá complementar lo referente al modelo ER de base de datos.

[¿Qué es un modelo ER?](#)

El recurso *online* muestra en detalle los componentes de un modelo entidad relación y cómo se construye este modelo. Esta guía en video pretende reforzar muchos aspectos que quizá todavía no estaban claros respecto del modelo ER.

## Diseño conceptual básico

La primera etapa de la metodología de diseño de base de datos es la fase de diseño conceptual. Como su nombre lo dice, esta fase produce un concepto abstracto de los requerimientos de datos de la organización. Se centra en los datos dejando de lado cualquier consideración relacionada con aspectos físicos de implementación. Los pasos a seguir para la construcción del modelo conceptual se resumen a continuación:

1. **Identificar tipos de entidad**, representan objetos de interés para el cliente y pueden ser personas, lugares, documentos, entre otros. Los objetos de interés pueden ser físicos, como por ejemplo personas, pero también pueden ser conceptos abstractos, como por ejemplo ventas. Se pueden identificar a través del documento de requisitos.
2. **Identificar tipos de relación**, nuevamente la fuente de información para definir los tipos de relación sería la especificación de requisitos y las podemos identificar porque generalmente se expresan mediante verbos o expresiones verbales. Identificar atributos.
3. **Determinar dominios de los atributos**, es decir, determinar los posibles conjuntos de valores para cada atributo. Por ejemplo, si tenemos un atributo que representa el estado civil de una persona, el dominio sería todos los estados civiles legalizados hasta ahora: soltero, casado, viudo, entre otros.
4. **Determinar atributos de clave candidata, principal y alternativa**, esto permitirá definir un conjunto de atributos como candidatos a ser clave primaria, luego de lo cual se elegirá a uno que será la clave principal y que permite identificar a cada tupla de la entidad de forma única. Los atributos candidatos que no fueron seleccionados como clave primaria se pueden utilizar como alternativas a la clave principal.
5. **Considerar conceptos de modelado avanzados**, es un paso opcional y se emplea cuando se requiere dotar de más semántica al modelado

de datos. Se puede aplicar principalmente especialización y generalización.

6. **Comprobar redundancia en el modelo para evitar duplicidad de datos**, es una de las características del modelo relacional.
7. **Validar el modelo comprobando las transacciones de los usuarios**, se refiere a que se debe chequear que el modelo de datos soporta todas las transacciones de los usuarios, satisfaciendo de esta manera los requisitos de datos de la organización.
8. **Revisar el modelo con los usuarios**, es recomendable que antes de seguir a la fase de diseño lógico se valide con el usuario el producto del diseño conceptual para evitar futuras inconsistencias en el modelado.

Una vez se ha completado todos los pasos descritos anteriormente, se puede pasar a la fase de diseño lógico que se describe en la semana 11.

### **Material complementario**

A efectos de reforzar la comprensión de los temas tratados, le recomendamos el siguiente recurso:

[Modelo Entidad/Relación](#): encontrará una explicación bastante didáctica sobre los componentes de un Modelo E-R.



### **Actividad de aprendizaje recomendada**

- A manera de práctica, intente diseñar un modelo ER para soportar su agenda de actividades diarias.



### **Semana 10**

---

En esta semana estudiaremos características avanzadas del modelo ER, que en ciertos escenarios puede ayudar a potenciar y mejorar la semántica de nuestro diseño conceptual.

## Diseño conceptual extendido

Esta actividad es opcional y se la utiliza para dar mayor semántica al modelo ER desarrollado en la primera fase del proceso de diseño de una base de datos.

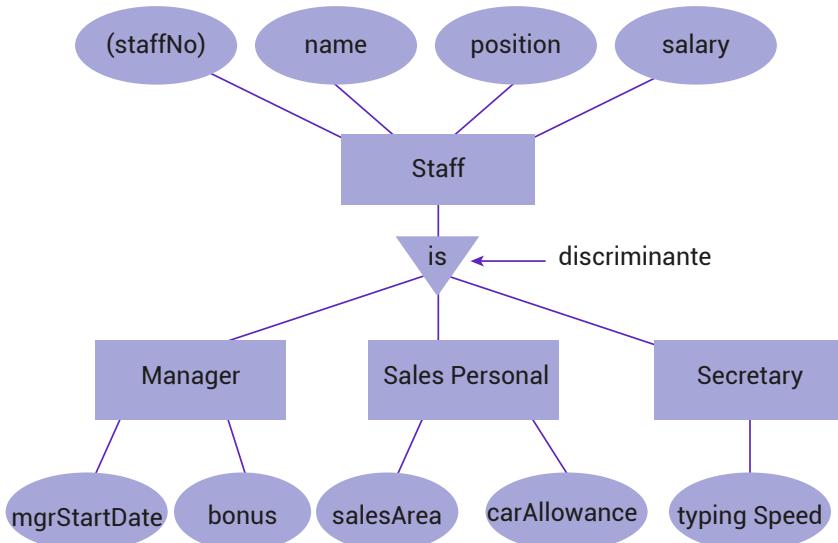
Se puede aplicar conceptos de modelado avanzados como la especialización y generalización en donde se incluyen tipos especiales de entidades que se conocen como superclases y subclases y con el proceso de herencia de atributos.

La **especialización** se considera una técnica arriba-abajo e intenta maximizar las diferencias entre los miembros de una entidad identificando sus características distintivas, es decir, describe desde el concepto más general al concepto más específico, y para representarlo utiliza la superclase y las subclases. Una superclase es un tipo de entidad que incluye atributos que son comunes a las subclases. Una subclase es un tipo de entidad que posee atributos que las distinguen unas de otras.

La figura 28 muestra un ejemplo en donde podemos identificar la entidad Staff que sería la superclase y de ella se derivan las subclases Manager, SalesPersonal y Secretary. La superclase (Staff) tiene los atributos que son comunes a todas las subclases y cada subclase posee, a más de los atributos comunes de la superclase que los heredarían de la superclase, sus propios atributos que las diferencias unas de otras. También vale destacar que la especialización se representa a través de un triángulo invertido que generalmente equivale a un atributo denominado discriminante y que en el caso del ejemplo podría ser tipo de empleado (type).

**Figura 28**

Ejemplo de especialización/generalización



Nota. Adaptado de *Guía didáctica de Fundamentos de Bases de Datos* (p. 124), por J. Morocho y A. Romero, 2019, Editorial de la Universidad Técnica Particular de Loja.

La **generalización** es el proceso inverso a la especialización, e intenta minimizar las diferencias entre entidades identificando sus características comunes, debido a que abstrae desde lo más específico a lo más general, por ello se denomina una técnica abajo-arriba.



En el siguiente recurso encontrará una explicación complementaria sobre el modelo ER extendido, ilustrados con distintos ejemplos.

[Modelo Entidad Relación extendido: jerarquías \(Hueso Ibáñez, 2015a, pp. 54-57\)](#)

El recurso presenta detalles de cómo afrontar un diseño de base de datos aplicando conceptos de modelado avanzados como lo son la especialización y generalización que sirven para dar más semántica al modelo.



## Actividad de aprendizaje recomendada

- Identifique casos prácticos en los que conviene aplicar las características avanzadas del modelo ER extendido.



## Semana 11

---

Durante la presente semana revisaremos el proceso de transformación del diseño conceptual a diseño lógico, mediante la aplicación de mecanismos que permiten traducir el modelo ER a un modelo relacional. Así mismo estudiaremos como se puede eliminar redundancia en un modelo lógico mediante técnicas de normalización.

### Diseño lógico

La segunda etapa de la metodología de diseño de la base de datos es el diseño lógico. Una vez tenemos el modelo ER, producto del diseño conceptual, este se traduce a un modelo lógico de los datos que representa los requisitos de datos de la organización.

Se empieza derivando un conjunto de relaciones (esquema relacional) tomando como base el diseño conceptual y luego se debe validar el esquema resultante a través de las técnicas de normalización. El esquema normalizado debe soportar las transacciones indicadas en la especificación de requisitos del usuario junto con todas las restricciones de integridad importantes. Así también, como paso final de la fase de diseño lógico de la base de datos se debe considerar hasta qué punto el modelo es capaz de soportar los posibles desarrollos futuros del sistema de base de datos.

### Traducción del diseño conceptual hacia el diseño lógico

Para traducir el modelo ER, producto del diseño conceptual, se debe seguir los siguientes lineamientos:

#### [Lineamientos para traducir el modelo ER](#)

Con estos sencillos pasos se puede obtener el esquema relacional, que es el resultado del diseño lógico, a partir del modelo ER que fue el resultado del diseño conceptual.



En el siguiente recurso podrá complementar lo referente al proceso de traducir el diseño conceptual al diseño lógico.

[Convertir el modelo ER al modelo relacional](#)

El recurso presenta una guía simplificada para traducir el diseño conceptual al diseño lógico. El modelo ER puede elaborarse siguiendo cualquiera de las notaciones de que se dispone como son la de Chen, pata de cuervo o UML, al final todas las notaciones lo que buscan es generar un diagrama que sea más entendible.

Continuando con el proceso de diseño lógico se tendrá que validar las relaciones mediante la técnica de la normalización, proceso que revisaremos más adelante.

Luego de validar las relaciones, comprobando que soporten las transacciones de los usuarios de acuerdo con los requisitos de datos, se debe comprobar las restricciones de integridad que sirven para proteger la base de datos frente a la posibilidad de que llegue a ser incompleta, imprecisa o incoherente. Dentro de los tipos de restricciones que podemos utilizar están las siguientes:

- Datos requeridos (NOT NULL).
- Restricciones relativas a los dominios de los atributos.
- Multiplicidad.
- Integridad de las entidades (PRIMARY KEY).
- Integridad referencial (FOREIGN KEY).
- Restricciones generales (DEFAULT, CHECK).

Repasar el modelo lógico de los datos con los usuarios, es otra actividad recomendada debido a que el diseño de la base de datos parte de los requisitos de los usuarios, por lo tanto, son ellos quienes deben dar el visto bueno al diseño propuesto.

Verificar las consideraciones derivadas del crecimiento futuro, se constituye en una actividad clave, ya que prepara el diseño de base de datos para un crecimiento sin mayores contratiempos, de otra manera el extender el modelo requerirá de mucho esfuerzo.

## Normalización

Ya se ha revisado el diseño lógico, pero una de las actividades clave en el proceso de diseño de una base de datos relacional lo es la Normalización y para ello emplearemos las formas normales, que permiten obtener un modelo de datos con las entidades mínimas requeridas y manteniendo un mínimo de redundancia.

Se ha optado por guiar el proceso de diseño basado en la metodología de diseño de base de datos propuesta por (Connolly y Begg, 2005) en la que tenemos las fases de diseño conceptual, diseño lógico y diseño físico. Una vez concluido el diseño lógico, se debe aplicar el proceso de normalización con el objeto de optimizar el modelo de datos, a través de la búsqueda de una redundancia mínima.

El proceso formal de normalización influye en dos escenarios:

- a. Su papel principal es apoyar en la identificación de relaciones, principalmente cuando tenemos esquemas grandes, a partir de los cuales desagregamos en varias relaciones de acuerdo con las dependencias funcionales, minimizando la redundancia de datos.
- b. Como medio de verificación para comprobar la estructura de las relaciones, cuando tenemos un modelo de relaciones que generalmente es producto de haber empezado por el diseño conceptual y luego haber llegado al diseño lógico.

Generalmente, cuando hay redundancia de datos, esta provoca anomalías de inserción, borrado y modificación.

## Dominios atómicos y primera forma normal

El modelo E-R permite que las entidades contengan atributos multivaluados y/o compuestos y al momento de crear la tabla eliminamos estas subestructuras. Para el caso de los atributos compuestos, creamos un atributo por cada componente y para atributos multivaluados creamos una tupla por cada ítem.

Cuando encontramos una entidad que tiene grupos repetitivos, decimos que está en una forma no normalizada (UNF). La *tabla 6* muestra un ejemplo de la tabla asignación donde se almacena datos de proyectos y empleados con grupos repetitivos.

**Tabla 6.**

*Tabla no normalizada*

projID	projName	empID	EmpName	jobClass	Chg_	Hours
15	Evergreen	103	June E. Arbough	Elect Engineer	84.5	23.6
		101	John G. News	Database Designer	105.0	19.4
		105	Alice K. Johnson*	Database Designer	105.0	35.7
		106	William Smithfield	Programmer	35.8	12.6
		102	David H. Senior	Systems Analyst	95.8	23.6
18	Amber Wave	114	Annelis Jones	Applications Designer	48.1	24.6
		118	Jammes J. Frommer	General Support	18.4	45.3
		104	Anne K. Ramoras*	Systems Analyst	96.8	32.4
		112	Darlene M. Smithson	DSS Analyst	45.9	44.0
22	Rolling Tide	105	Alice K. Johnson	Database Designer	105.0	64.7
		104	Anne K. Ramoras	Systems Analyst	96.8	46.4
		113	Delbert K.	Applications Designer	48.1	29.6
		111	Joenbrood*	Clencal Support	26.9	22.0
		106	Geoff B. Wabash	Programmer	35.8	12.6
25	Starflight	107	Maria D. Alonso	Programmer	35.8	24.6
		115	Travis B. Bawangi	Systems Analyst	96.8	45.8
		101	John G. News *	Database Designer	105.0	56.3
		114	Annelis Jones	Applications Designer	48.1	33.1
		108	Ralph B. Washington	Systems Analyst	96.8	23.6
		118	Jammes J. Frommer	General Support	18.4	30.5
		112	Darlene M. Smithson	DSS Analyst	45.9	41.4

Nota. Tomado de *Database Systems: Design, Implementation, & Management* (p. 202), por C. Coronel y S. Morris, 2019, Cengage Learning.

Para pasar a primera forma normal basta con hacer que cada intersección de fila y columna contenga uno y un solo valor. Esto también se conoce como dominio atómico. La *tabla 7* muestra la tabla Asignacion en 1FN.

**Tabla 7.**

*Tabla normalizada en 1FN*

projID	projName	empID	EmpName	jobClass	Chg_	Hours
15	Evergreen	103	June E. Arbough	Elect Engineer	84.5	23.6
15	Evergreen	101	John G. News	Database Designer	105.0	19.4
15	Evergreen	105	Alice K. Johnson*	Database Designer	105.0	35.7

projID	projName	empID	EmpName	jobClass	Chg_	Hours
15	Evergreen	106	William Smithfield	Programmer	35.8	12.6
15	Evergreen	102	David H. Senior	Systems Analyst	95.8	23.6
18	Amber Wave	114	Annelis Jones	Applications Designer	48.1	24.6
18	Amber Wave	118	Jammes J. Frommer	General Support	18.4	45.3
18	Amber Wave	104	Anne K. Ramoras*	Systems Analyst	96.8	32.4
18	Amber Wave	112	Darlene M. Smithson	DSS Analyst	45.9	44.0
22	Rolling Tide	105	Alice K. Johnson	Database Designer	105.0	64.7
22	Rolling Tide	104	Anne K. Ramoras	Systems Analyst	96.8	46.4
22	Rolling Tide	113	Delbert K. Joenbrood*	Applications Designer	48.1	29.6
22	Rolling Tide	111	Geoff B. Wabash	Clecal Support	26.9	22.0
22	Rolling Tide	106	William Smithfield	Programmer	35.8	12.6
25	Starflight	107	Maria D. Alonzo	Programmer	35.8	24.6
25	Starflight	115	Travis B. Bawangi	Systems Analyst	96.8	45.8
25	Starflight	101	John G. News *	Database Designer	105.0	56.3
25	Starflight	114	Annelis Jones	Applications Designer	48.1	33.1
25	Starflight	108	Ralph B. Washington	Systems Analyst	96.8	23.6
25	Starflight	118	Jammes J. Frommer	General Support	18.4	30.5
25	Starflight	112	Darlene M. Smithson	DSS Analyst	45.9	41.4

Nota. Tomado de *Database Systems: Design, Implementation, & Management* (p. 206), por C. Coronel y S. Morris, 2019, Cengage Learning.

## Dependencias funcionales

Un concepto asociado a Normalización es el de dependencias funcionales (DF), que describe la relación entre atributos.

Con las DF aparece el término determinante, que es el atributo o atributos situados a la izquierda de la flecha que describe la dependencia y el término determinado que es el atributo que está a la derecha.

Si tenemos una relación R, con los atributos A y B, se dice que B depende funcionalmente de A ( $A \rightarrow B$ ) siempre que para cada valor distinto en A hay un solo valor en B.

La normalización se basa en las dependencias funcionales para agrupar atributos e ir componiendo las entidades, obteniendo como resultado un modelo de tablas.

Una dependencia funcional se dice completa si es que el atributo determinado depende completamente de todos los atributos que forman la clave principal.

Por el contrario, una dependencia parcial se da cuando hay una dependencia funcional en la que el determinante es solo parte de la llave primaria (recuerde que estamos suponiendo que hay solo una clave candidata). Por ejemplo, si  $(A, B) \rightarrow (C, D)$ ,  $B \rightarrow C$  y  $(A, B)$  es la clave primaria, entonces la dependencia funcional  $B \rightarrow C$  es una dependencia parcial porque solo parte de la clave primaria ( $B$ ) se necesita para determinar el valor de  $C$ . Las dependencias parciales tienden a ser más bien sencillas y fáciles de identificar. (Coronel & Morris, 2011)

Otro tipo de dependencia funcional es la dependencia transitiva, que se basa en la ley de transitividad de conjuntos, si  $X \rightarrow Y$ ,  $Y \rightarrow Z$  y  $X$  es la llave primaria,  $X \rightarrow Z$  es una dependencia transitiva porque  $X$  determina el valor de  $Z$  a través de  $Y$ . (Coronel y Morris, 2011).

Es importante que en cada entidad se defina una clave primaria, para luego definir las dependencias funcionales con base a esta. Tomando el ejemplo de la tabla Asignacion, podemos definir la clave primaria como la combinación de PROJ\_NUM y EMP\_NUM, que determina al resto de atributos, como muestra:

PROJ\_NUM, EMP\_NUM  $\rightarrow$  PROJ\_NAME, EMP\_NAME, JOB\_CLASS, CHG\_HOUR, HOURS

### **Segunda forma normal**

La conversión a segunda forma normal se da siempre y cuando la clave primaria de la relación sea una clave compuesta. Si es una clave primaria simple, la tabla automáticamente pasa de 1FN a 2FN.

Como primer paso se crean nuevas tablas para eliminar las dependencias parciales. Para ello es necesario separar cada componente de la clave compuesta si este genera una dependencia parcial, como si fuera una nueva relación, pero conservando en la relación original el atributo determinante para que actúe como clave foránea. Continuando con el ejemplo de la tabla Asignación, quedaría de la siguiente forma:

PROJ\_NUM  $\rightarrow$  genera la tabla proyectos

EMP\_NUM  $\rightarrow$  genera la tabla empleados

PROJ\_NUM, EMP\_NUM  $\rightarrow$  la tabla asignaciones

Luego los atributos que son dependientes funcionalmente en una dependencia parcial se remueven de la tabla original y se colocan en la nueva tabla con su determinante. Cualquier atributo que no sea dependiente en una dependencia parcial permanecerá en la tabla original. Esto da origen a las siguientes tablas:

Proyectos (PROJ\_NUM, PROJ\_NAME)

Empleados (EMP\_NUM, EMP\_NAME, JOB\_CLASS, CHG\_HOUR)

Asignacion (PROJ\_NUM, EMP\_NUM, ASSIGN\_HOURS)

Se ha obtenido un modelo de tablas que minimiza los problemas de redundancia traducidos en anomalías de modificación, además se puede ya identificar las claves foráneas que sirven para relacionar las tablas.

### Tercera forma normal

En este punto todavía pueden persistir anomalías en el modelo de datos, lo que puede causar problemas, así que se deben eliminar. La 3ra FN pide eliminar las dependencias transitivas, entonces se debe identificar las dependencias transitivas y escribir una copia de su determinante como clave primaria para una nueva tabla, pero en la tabla original debe permanecer el determinante para que actúe como clave foránea. Continuando con el ejemplo de la asignación de empleados a proyectos, quedaría de la siguiente forma:

JOB\_CLASS → determinante de la dependencia transitiva identificada

El siguiente paso sería reasignar los atributos dependientes de los determinantes identificados como dependencias transitivas. El esquema de tablas quedaría de la siguiente forma:

Proyectos (PROJ\_NUM, PROJ\_NAME)

Empleados (EMP\_NUM, EMP\_NAME, JOB\_CLASS)

Cargo (JOB\_CLASS ,CHG\_HOUR)

Asignacion (PROJ\_NUM, EMP\_NUM, ASSIGN\_HOURS)

Una vez eliminada la dependencia transitiva las tablas están en 3ra FN. Existen otras formas normales, pero generalmente un modelo normalizado

hasta la 3FN ha reducido al mínimo la redundancia y permitirá trabajar de forma adecuada. En la mayoría de los casos llegar a tercera forma normal es suficiente y no requiere procesos de normalización avanzados.

### **Material complementario**

Opcionalmente, y a efectos de reforzar la comprensión de los temas tratados, le recomendamos el siguiente recurso:

- [\*\*Fundamentos de la normalización de bases de datos\*\*](#): encontrará una explicación del proceso de normalización de base de datos que complementará lo revisado en este apartado. Incluye un ejemplo que muestra paso a paso la ejecución del proceso hasta llegar a tercera forma normal.



## Actividad de aprendizaje recomendada

- Con base en los siguientes datos no normalizados, aplique el proceso de normalización hasta llegar a la 3FN.

### Ventas

numFactura	fechaFactura	nomCliente	direcCliente	cantArticulo	nomArticulo	tipoArticulo	precioArticulo	monto
100346	15/06/2021	Henry Guarnizo	Av. Próceres 1246	2	Pantalón	Casimir	25	50
100347	16/06/2021	Claudia Pérez	Av. Ibarra 1345	1	Polo	Algodón	30	30
100348	16/06/2021	Paúl Mora	Av. Los Jardines 3456	2	Polo	Algodón	30	60
100348	16/06/2021	Paúl Mora	Av. Los Jardines 3456	1	Pantalón	Jean	45	45
100349	17/06/2021	Pedro Arias	Av. Las Flores 1738	3	Camisa	Algodón	24	72
100350	17/06/2021	Erika González	Av. Pedregal 1589	5	Medias	Deportivos	12	60



- Resultado de aprendizaje 3**
- Utiliza gestores de base de datos para implementar y administrar información.

## Contenidos, recursos y actividades de aprendizaje



Semana 12

### Unidad 3. Implementación y administración de bases de datos



En esta semana veremos cómo se definen las especificaciones físicas de implementación de la base de datos a partir del diseño conceptual y lógico de la base de datos. Incorporando al diseño características que aseguren la confidencialidad, integridad y disponibilidad de la información.

#### Diseño físico de una base de datos

Mientras que el diseño conceptual se ocupa del "para qué" y el diseño lógico se ocupa del "qué", el diseño físico se ocupa del "cómo". Aquí es donde se adapta el diseño lógico a un SGBD concreto. Es decir, en el diseño físico se establecen todas las especificaciones técnicas para implementar la base datos. Por lo que, en este punto es necesario haber elegido el SGBD que se va a usar, y conocer a fondo las capacidades y prestaciones del motor.

Aquí es donde aparece el rol conocido como DBA (Administrador de la Base de Datos), que es aquel especialista en el SGBD elegido, y es quien se encarga de todo el proceso que sigue a partir del diseño físico (implementación, carga, pruebas, puesta en operación, y mantenimiento), por lo tanto, el DBA es quien está en capacidad de aprovechar al máximo las prestaciones del motor, de cara a garantizar seguridad y eficiencia en la operación de la base de datos.



En el siguiente video encontrará una explicación del diseño físico de una base de datos con base en un caso de estudio.

[Diseño físico de bases de datos \(Encalada, 2016a\)](#)

Realmente el proceso de diseño físico puede tener distintos matices. Dependiendo del tipo de proyecto, de la magnitud y criticidad de los datos, de la carga esperada, y del nivel de disponibilidad requerido, pueden ajustarse los pasos, o limitarse el alcance de cada uno de ellos.

La figura 29 ilustra un proceso de diseño físico simplificado, que puede aplicar a la mayoría de los proyectos.

**Figura 29***Proceso simplificado del diseño físico*

Nota. Encalada, E. Morocho, J., 2022.

Como vemos, el artefacto principal resultante del diseño físico es el SCRIPT para la implementación de la base de datos, el cual está escrito en el lenguaje de la base de datos elegida (SQL en bases de datos relacionales) y contiene toda la secuencia de comandos necesarios para materializar la base datos, con base en las especificaciones lógicas y físicas que se hayan establecido. La gran mayoría de las especificaciones que se definen en cada paso del proceso, se van incorporando al script.

En la *tabla 8* se describe el alcance de cada paso. De ellos, los dos primeros los abordaremos en este apartado, y los siguientes en secciones subsiguientes.

**Tabla 8.***Descripción del proceso del diseño físico*

Paso	Descripción
1. Traducir el modelo lógico a lenguaje del SGBD.	Implica traducir a lenguaje de base de datos, todas aquellas especificaciones definidas a nivel del diseño lógico. En el caso de las bases de datos relacionales, implica traducir el diseño de tablas, con todas sus restricciones a lenguaje de definición de datos de SQL.

Paso	Descripción
2. Definir la organización física de la base de datos.	Conlleva definir como se organizarán las estructuras lógicas (tablas en bases de datos relacionales) en el almacenamiento secundario (disco) del servidor, de tal manera que se garantice espacio suficiente y acceso eficiente a los datos.
3. Definir mecanismos y restricciones de seguridad.	Además de las restricciones de integridad que ya vienen recogidas en el diseño conceptual y lógico de la base de datos, es necesario establecer otros mecanismos que ayuden a garantizar la seguridad de los datos. Por ejemplo: gestión de autenticación y autorización, implementación de vistas, entre otros.
4. Establecer otras especificaciones de implementación.	Otras especificaciones de implementación que se pueden considerar son, por ejemplo: la implementación de índices para mejorar el rendimiento de las consultas, configuraciones y restricciones necesarias a nivel de servidor para garantizar la disponibilidad del servicio.

Nota. Encalada, E. Morocho, J.. 2022.

### Traducir el modelo lógico a lenguaje de bases de datos

Lo primero que se hace en el diseño físico es generar el script inicial, que contiene la secuencia de comandos que permitirán implementar las estructuras y especificaciones definidas en el diseño lógico.

En el caso de las bases de datos relacionales, el lenguaje específico que se utiliza para la generación del script se denomina DDL (Data Definition Language) que es parte de SQL. El cual es bastante similar entre los distintos SGBD relacionales. En la semana 3, ya se introdujo acerca de este lenguaje, pero es necesario profundizar.



En el siguiente recurso encontrará una explicación más a detalle sobre el lenguaje DDL. La explicación está orientada a MySQL.

[Diseño físico de bases de datos \(Hueso Ibáñez, 2015a, pp. 92-105\)](#)

Más allá de que para generar el script nos ayudemos de alguna herramienta especializada, como MySQL Workbench, Oracle SQLDeveloper Datamodeler, PowerDesigner DataArchitect, u otra, es importante saber escribir y usar DDL, ya que en muchos casos es necesario introducir cambios en el script.



Revise el siguiente recurso donde se exponen las consideraciones a tener en cuenta al adaptar el modelo lógico al SGBD.

[Traducir modelo lógico a SGBD \(Encalada, 2016b\)](#)

Para identificar cuáles restricciones de integridad se pueden implementar mediante DDL, considere que usualmente la mayoría de los SGBD relacionales solo permiten implementar las siguientes restricciones:

- Integridad de entidades (constraint PRIMARY KEY).
- Integridad referencial (constraint FOREIGN KEY).
- Claves alternativas (constraint UNIQUE).
- Campos obligatorios (constraint NOT NULL).
- Validación de dominio (constraint CHECK).

El resto de las restricciones normalmente las deberá controlar con Triggers o en la aplicación de usuario final.

### **Definir la organización física de la base de datos**

A nivel físico, cada gestor de base de datos maneja su propio esquema y estructuras de almacenamiento, incluso si se tratan de motores del mismo tipo. Por ejemplo, la forma que Oracle, MySQL, SQL Server, DB2, PostgreSQL almacenan los datos en disco, difieren completamente, aunque todos son SGBD relacionales.

En este punto del diseño físico se debe establecer la manera en que se distribuirán los datos en el disco del servidor, en función de las prestaciones y facilidades del motor elegido. El objetivo es facilitar al motor el proceso de lecturas y escrituras (operaciones I/O), es decir, que las transferencias de bloques de datos entre almacenamiento primario (memoria RAM) y almacenamiento secundario (disco) sean lo más rápidas posibles. Por ejemplo: tablas que se acceden de manera concurrente, se las puede almacenar en ubicaciones distintas.



Revise el siguiente recurso donde se explica la estructura física de una base de datos, tomando como ejemplo Oracle.

[Diseño físico de bases de datos \(Cabré I Segarra et al., s. f.\)](#)

Entonces, a nivel de diseño físico, lo principal que se debe definir es:

- **El espacio requerido en almacenamiento secundario:** con base en una proyección de crecimiento del volumen de datos de tres a cinco años.
- **La distribución física de los datos:** con base en un análisis de las tablas críticas, según el nivel de transaccionalidad y concurrencia esperado.

## La seguridad de los datos

Una de las mayores preocupaciones al implementar una base de datos es la seguridad de los datos. Qué no se produzcan accesos no autorizados, qué no ocurran ataques que afecten a la integridad de los datos, qué no haya robo de información, qué no existan inconvenientes con la operación continua del servicio de base de datos, qué no se filtre información confidencial, etc., son las preocupaciones comunes. Y tanto el Administrador de datos como el DBA son los llamados a establecer todos los mecanismos preventivos y correctivos que permitan proteger el contenido, garantizar la integridad, y asegurar la operación de la base de datos.

La seguridad de los datos debe analizarse principalmente desde tres perspectivas: la base de datos, las aplicaciones, y la infraestructura. Dependiendo de las capacidades del SGBD, muchos controles para la seguridad de la información se pueden implementar en el motor (autenticación, autorización, restricciones de integridad, etc.); pero no es el único punto. Las aplicaciones de usuario final que acceden a la base de datos también son un punto crítico por el cual pueden producirse accesos indebidos que vulneren la base de datos, por lo que es muy importante que a nivel de los aplicativos existan los debidos controles de autenticación y de autorización. Y así mismo, a nivel físico (servidores y red transmisión de datos) se deben implementar controles y restricciones que coadyuben

disminuir la probabilidad de que ocurra una vulneración a los datos o una afectación a la operación del servicio.

### Objetivos de la seguridad de los datos.

La seguridad de los datos conlleva 3 objetivos principales: garantizar la **confidencialidad** de la información, preservar la **integridad** de los datos, y asegurar la **disponibilidad** del servicio de base de datos. Conocida como Triada CID.



Le invitamos a revisar los siguientes recursos donde encontrará una explicación sobre los 3 objetivos que abarca la seguridad de una base de datos: confidencialidad, integridad, y disponibilidad.

- [¿Qué es la tríada de la CIA?](#)
- [Confidencialidad, integridad y disponibilidad \(tríada CIA\)](#)

No lo olvide, la seguridad de la base de datos no solo es autenticación. Son todas aquellas medidas que apunten a conseguir esos tres objetivos:

- **Confidencialidad:** impedir el acceso a información no autorizada.
- **Integridad:** garantizar la validez y consistencia de los datos.
- **Disponibilidad:** asegurar que el servicio de la base de datos se mantenga operativo (tanto en tiempo como en rendimiento), según lo exija el negocio.

A continuación lo invitamos a revisar las amenazas y contramedidas.

### [Amenazas y contramedidas](#)

### **Gestión de acceso de usuarios**

Permite implementar el control lógico de acceso a la base de datos. Corresponde al control de autenticación y autorización. La autenticación

corresponde a la validación de credenciales (login y password) del usuario, mientras que la autorización valida los permisos del usuario.



Revise el siguiente recurso en el que se explica el sistema de acceso a los recursos de un servidor MySQL, mediante la implementación de mecanismos de autenticación y autorización.

[Gestión de cuentas de usuario y permisos  
\(Hueso Ibáñez, 2015b, pp. 61-77\)](#)

Como vemos, en el marco de la gestión de la seguridad de una base de datos es posible establecer restricciones de acceso a los distintos recursos del provee el servicio, dependiendo del perfil de usuario que requiera conectarse al motor. Y debe quedar claro que una cosa es la autenticación que valida las credenciales mediante usuario y contraseña, y otra cosa son las autorizaciones que tiene el usuario para acceder a los datos y recursos del servidor.

La gestión del acceso de usuarios implica:

- Gestión de usuarios.
- Gestión de privilegios.
- Gestión de roles.
- Gestión de perfiles.

La diferencia entre roles y perfiles depende de cada motor de base de datos. Normalmente, un rol corresponde a una agrupación de privilegios, mientras que un perfil corresponde a una configuración específica de cuotas de recursos y otras restricciones de acceso a la base de datos.

### **Material complementario**

Opcionalmente, y a efectos de reforzar la comprensión de los temas tratados, le recomendamos los siguientes recursos:

- [Almacenamiento y estructura de archivos \(Silberschatz et al., 2014, pp. 195-205\)](#): le permitirá comprender los diferentes medios físicos de almacenamiento que se usan en el marco del diseño físico de un base de datos.

- [Almacenamiento y diseño físico en Oracle \(Cuadra et al., 2014, pp. 411-416\)](#): encuentra una interesante explicación de las estructuras de almacenamiento físico de una base de datos Oracle. Le servirá para comprender mejor la gran diferencia que existe entre el nivel lógico y el nivel físico de una base de datos.



## Actividades de aprendizaje recomendadas

- A partir del diseño lógico propuesto para una base de datos orientada a la gestión del alquiler de bicicletas en una ciudad, realice la traducción a lenguaje DDL para MySQL. En el script DDL incluya especificaciones que permitan que las tablas físicamente se almacenen en un directorio distinto al predeterminado por MySQL. El diseño lógico propuesto lo tiene disponible en el siguiente enlace: <https://rebrand.ly/xkk6qhq>.
- Investigue acerca de la implementación de vistas de datos y sus beneficios en el contexto de la seguridad de una base de datos. Plantee ejemplos de casos en los que sería apropiado la implementación de vistas.
- Investigue más acerca de las amenazas a las que puede estar expuesta una base de datos, y las medidas más apropiadas para contrarrestarlas. Elabore una matriz de cruce para el efecto.



## Semana 13

---

Durante la presente semana veremos cómo se produce la materialización de la base de datos una vez definidas las especificaciones de implementación en el marco del diseño físico. Y también entenderemos a qué se refiere la carga de la base de datos, que formalmente siempre se realiza luego de la implementación.

## Implementación y carga de la base de datos

Concluido el diseño físico se procede a implementar la base de datos. Ello implica:

- Instalar y configurar el SGBD.
- Crear y configurar la base de datos (esquema, estructuras físicas, tablas, índices, etc.) con todas las especificaciones del diseño físico.
- Configurar cuentas de usuario y permisos.
- Realizar la carga inicial de los datos.

Las particularidades de esta fase son muy dependientes del SGBD que se vaya a utilizar. Cada motor -aunque sean del mismo tipo- maneja sus propios esquemas de instalación, configuración, y optimización. En nuestro caso usaremos como servidor de base de datos MySQL (<https://dev.mysql.com/downloads/mysql/>), y como herramienta de administración MySQL Workbench (<https://dev.mysql.com/downloads/workbench/>).

### Instalación y configuración del SGBD

El primer paso es preparar el entorno operativo sobre el cual se va a desplegar el servicio de la base de datos. Ello conlleva la preparación del servidor, la instalación y configuración del sistema operativo, y la instalación del SGBD seleccionado para materializar nuestra base de datos.



Revise el siguiente recurso en el que se explica el proceso de instalación y configuración de un servidor de base de datos MySQL.

[Instalación y configuración de un SGBD \(Hueso Ibáñez, 2015b, pp. 21-46\)](#)

Como ya sabemos, a nivel físico cada sistema gestor de base de datos tiene sus propias características, ello se extiende a nivel de la instalación y configuración. Es necesario estudiar cada motor específico para comprender las prestaciones que tiene, para con ello poder establecer la configuración más adecuada que se debe realizar durante en el proceso de instalación del servicio de base de datos.

## Creación y configuración de la base de datos

Una vez instalado el motor, estamos listos para implementar la base de datos. Ello implica materializar todas aquellas estructuras de datos, restricciones, y especificaciones definidas en el diseño. Básicamente conlleva dos pasos:

- a. Crear el esquema de base de datos.
- b. Ejecutar el script DDL.



Para comprender el proceso de creación de una base de datos, busque y revise en YouTube el video titulado "[Como crear y configurar Base Datos en MySQL utilizando el Workbench y la consola de Windows \(CMD\)](#)".

Una vez creada la base de datos, lo que sigue es cargar y ejecutar el script DDL que se generó en el marco del diseño físico. Con ello tendremos creadas todas las tablas con las especificaciones lógicas y físicas definidas, y con las restricciones de integridad establecidas. Y si en el diseño físico se estableció una organización física diferente a la predeterminada, se tendrá que primero configurar dichos espacios a nivel de sistema operativo.

## Configuración de cuentas de usuario y permisos

En el marco de la seguridad de los datos y de cara a garantizar la confidencialidad y la integridad de la información, es muy importante implementar las especificaciones de autenticación y autorización que aseguren el control del acceso a la base de datos.



Para aprender a crear usuarios y asignar permisos en MySQL, en YouTube, revise el video titulado "[Gestión de cuentas de usuario y permisos en MySQL](#)".

No olvide que las cuentas de superusuario (*root* en MySQL) solo están reservadas para uso del DBA. Jamás se debe acceder con ese tipo de cuentas desde las aplicaciones de usuario final.

## Carga inicial de datos

En toda implementación de base de datos se requiere realizar una carga inicial, que corresponde a la inserción de los datos con los que la base de datos entrará en producción. Normalmente, en la mayoría de los casos, las bases de datos al entrar en operación no arrancan 100% vacías, siempre existirán tablas en las cuales se necesita cargar una información inicial, normalmente en tablas de catálogos y/o tablas de parámetros. Incluso existen escenarios más complejos en los cuales se necesita migrar la información de una base de datos anterior, cuando, por ejemplo, se está implementando un nuevo sistema que reemplaza a uno antiguo.

Muchos autores la consideran a la *carga inicial* como otra fase dentro del ciclo de desarrollo de una base de datos, precisamente, por la complejidad que puede llegar a tener.

El proceso para realizar la carga inicial es similar al siguiente:

1. **Identificar datos de partida:** implica establecer cuál es la información que se necesita tener precargada en la base de datos para cuando el sistema entre en producción.
2. **Seleccionar fuentes:** se debe identificar de donde se va a obtener los datos que se requieren cargar. Las fuentes pueden ser diversas: otros sistemas, internet, hojas de cálculo, documentos digitales, documentos impresos, etc.
3. **Preparar los datos a cargar:** corresponde a la extracción de los datos desde las fuentes, y su preparación para la carga. Se lo puede hacer con el uso de herramientas especializadas, hojas de cálculo, o con programación.
4. **Cargar los datos:** con los datos preprocesados, se procede al importar los datos en las tablas de la base de datos, mediante procesos de carga masiva.



En YouTube, revise el video titulado “[Base de Datos #MySQL | Carga Masiva de Datos MySQL](#)”, en el cual se explica cómo realizar cargas masivas de datos en MySQL, usando el comando load.

Es muy importante que todo este proceso de carga quede automatizado, de manera que se pueda volver a ejecutar. Tome en cuenta que luego de la fase de implementación y carga, corresponde realizar pruebas a la base de datos, y para ello se parte de la base de datos inicializada, incorporando nueva información no real, como parte dicho control de calidad. Por lo que, una vez superadas las pruebas, y antes de entrar en operación, se deberá volver a inicializar a la base de datos, lo que implica volver a ejecutar el proceso de carga inicial.



### Actividades de aprendizaje recomendadas

- Intente realizar su propia implementación de MySQL server, personalizando el juego de caracteres, y el número de sesiones concurrentes permitidas.
- Investigue como actualizar la contraseña de la cuenta del usuario root y restringir el acceso con esa cuenta solo a ciertas IPs.
- Revise como realizar una carga masiva de datos en MySQL.



### Semana 14

---

Durante la presente semana revisaremos los principales mecanismos de afinación de una base de datos, de manera que siempre estemos en condiciones de asegurar un rendimiento adecuado. Y también, discutiremos acerca de una de las principales responsabilidades que tiene el DBA durante la fase de monitoreo y mantenimiento de la base de datos, que es la obtención continua de respaldos, a efectos de garantizar la no pérdida de información.

## Gestión del rendimiento

Como se explicó en la semana 13, uno de los aspectos más importantes que se deben garantizar al implementar una base de datos es la disponibilidad, que significa que el servicio de la base de datos de operar dentro de parámetros de continuidad y rendimiento aceptables. Para ello es muy importante que una vez que se haya concluido con la implementación y carga de la base de datos, se someta a esta a una serie de pruebas para evaluar si cumple con los parámetros esperados.

A raíz de las pruebas se podría necesitar realizar ajustes a la implementación de la base de datos, que es lo que se conoce como afinación del desempeño. Si el diseño de la base de datos bien hecho, los ajustes que se deban realizar serán mínimos. En cambio, si se hizo un mal diseño, los ajustes podrían implicar incluso al diseño conceptual, algo nada deseable. Por ello en este punto es importante destacar la importancia que tienen el realizar un buen diseño, eso es fundamental para garantizar el éxito de la implementación.

Ante un problema de rendimiento, se deben identificar las causas en un orden determinado, tomando como base los niveles de abstracción que se estudiaron en la semana 2.



En el siguiente recurso encontrará una explicación sobre el proceso de afinación de un base de datos, definido en función del orden en que el se deben buscar las causas que originan el problema.

[Proceso de afinación de una base de datos  
\(Encalada, 2021\)](#)

En bases de datos relacionales, los problemas de rendimiento en su mayor parte se detectan al realizar consultas SQL (comando SELECT), dado que es la operación más compleja y que representa una mayor carga de trabajo para el sistema gestor de la base de datos. Por ello, muchas de las técnicas y mecanismos orientados a mejorar el rendimiento, tienen que ver justamente con la OPTIMIZACIÓN DE CONSULTAS, e incluyen, entre otros,

el manejo de índices, la introducción de redundancia (desnormalización), y las buenas prácticas SQL.

## Índices

Uno de los mecanismos que ayudan a mejorar el rendimiento de las operaciones de consulta en una base de datos, es la implementación de índices. Un índice es una estructura ordenada asociada a un determinado atributo, que facilita la búsqueda y ordenación en ese atributo. Evita que cuando el motor necesite filtrar u ordenar por un campo indexado tenga que hacer un barrido secuencial de todas las filas de una tabla, optimizando con ello los tiempos de respuesta.



Para comprender mejor los índices y su implementación, revise el siguiente recurso:  
[Índices \(Hueso Ibáñez, 2015a, pp. 183-189\)](#)

Aunque existan múltiples tipos de índices, el propósito final es facilitar al motor las tareas de búsqueda y ordenamiento de los datos. Y son necesarios, ya que, por defecto, una tabla no es una estructura ordenada, los registros no se guardan en un orden específico, se van agregando siempre al final en el orden de llegada.

Los índices son útiles sobre todo en tablas que tienen una gran cantidad de registros. Y se los identifica con base en:

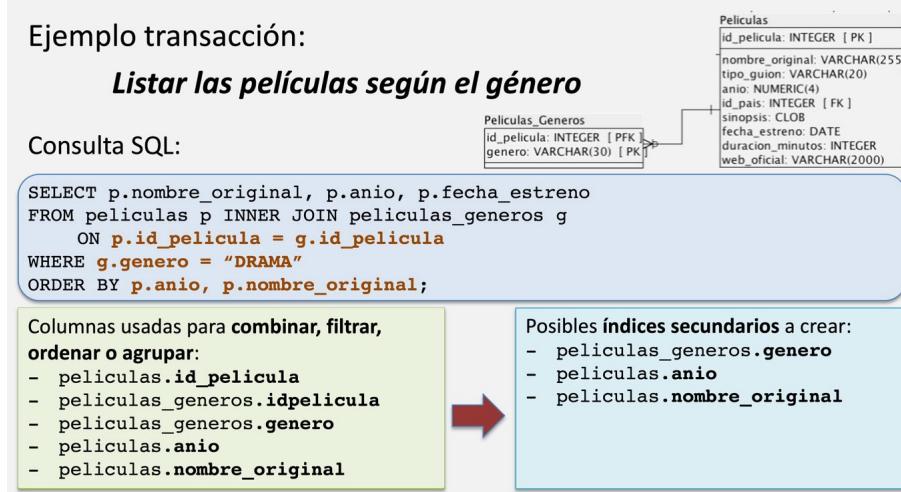
- Atributos que son más usados en operaciones de consulta para **filtrar, combinar, ordenar o agrupar** información.
- Llaves primarias (PK) siempre tienen un índice asociado (el SGBD lo crea automáticamente).
- Para llaves foráneas (FK) se recomienda siempre crear un índice (algunos SGBD también lo hacen de forma automática).

En la *figura 30* se muestra un ejemplo de cómo se identifican las columnas a ser indexadas. Se toman como base consultas que se realizarían con frecuencia y que acceden a las tablas de mayor volumen, y a partir de ello

se identifican las columnas que son usadas para combinar, filtrar, ordenar, y agrupar. De esas columnas se asume que las llaves primarias y llaves foráneas siempre estarán indexadas, y las restantes son las que se tendrían que indexar manualmente (ver figura 31).

### Figura 30

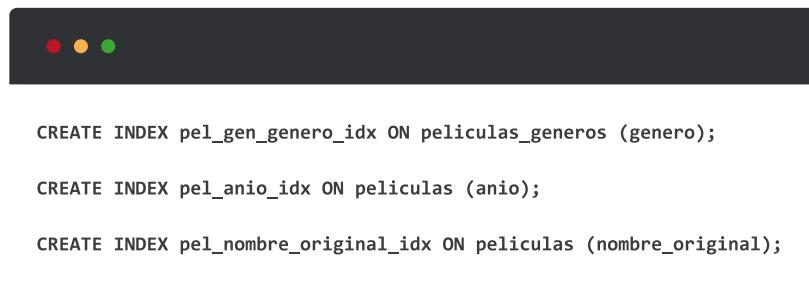
Ejemplo de identificación de columnas candidatas a indexar



Nota. Tomado de Índices (p. 5), por A. E. Encalada, 2022 ([Enlace web](#))

### Figura 31

Ejemplo implementación de índices



Nota. Encalada, E. Morocho, J., 2022.

En bases de datos transaccionales se debe reducir el uso de índices al mínimo necesario, sobre todo en columnas de tablas que se actualizan con mucha frecuencia; esto es debido a que los índices aceleran las consultas, pero ralentizan las actualizaciones.

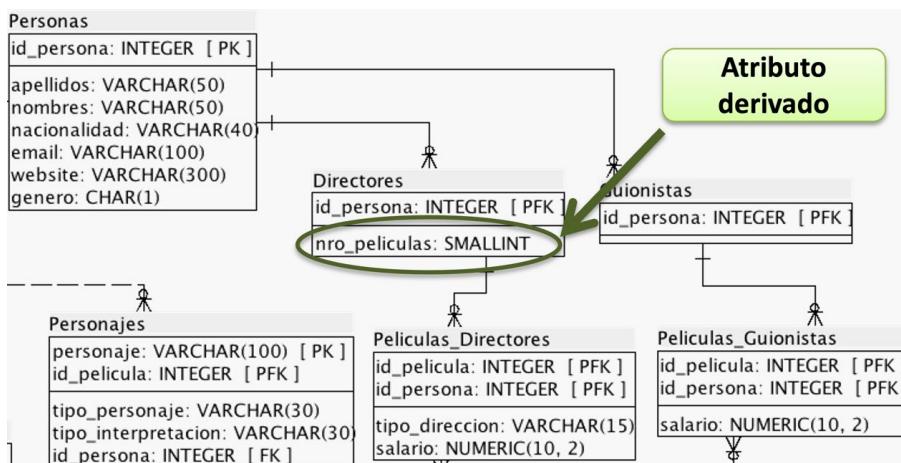
## Desnormalización

La desnormalización es la acción de introducir redundancia a una base de datos, es el proceso contrario a la normalización. El objetivo es reducir los cálculos y/o la cantidad de tablas que requieren acceder en una consulta SQL. La inclusión de atributos derivados, la unificación de tablas, la duplicación de campos en distintas tablas, son ejemplos de técnicas de desnormalización.

Un ejemplo de la técnica *inclusión de atributos derivados* es la que se muestra en la figura 32. En ese caso *nro\_peliculas* es un campo que se considera como derivado o calculado, ya que si no existiera se podría calcular a partir del resto de la información de la base de datos, por lo tanto, es un campo redundante; pero que puede ser muy útil para acelerar la ejecución de ciertas consultas.

**Figura 32**

Ejemplo de inclusión de atributos derivados



Nota. Adaptado de Técnicas de desnormalización (p. 9), por A. E. Encalada, 2017 ([Enlace web](#)).

En la figura 33 se muestra cómo sería la consulta sin atributos derivados y con la inclusión de atributos derivados. La diferencia es clara, en este caso se reduce el número de tablas a consultar, y se evita la necesidad del agrupamiento y agregación. Naturalmente, la segunda consulta representará menos carga para el SGBD y se ejecutará mucho más rápido.

### Figura 33

Contraste de consultas con la inclusión de atributos derivados

- Sin atributos derivados:

```
SELECT p.apellidos, p.nombres,  
       COUNT(distinct f.id_pelicula) nro_peliculas  
  FROM personas p INNER JOIN directores d  
    ON p.id_persona = d.id_persona  
    INNER JOIN peliculas_directores f  
      ON d.id_persona = f.id_persona  
 GROUP BY p.apellidos, p.nombres;
```

- CON atributos derivados:

```
SELECT p.apellidos, p.nombres, d.nro_peliculas  
  FROM personas p INNER JOIN directores d  
    ON p.id_persona = d.id_persona;
```

Nota. Adaptado de Técnicas de desnormalización (p. 10), por A. E. Encalada, 2017 ([Enlace web](#)).

Sin embargo, la desnormalización debe usarse con mucho cuidado, sobre todo en bases de datos transaccionales, ya que, en contraposición, puede conllevar riesgos de inconsistencia en los datos. En el ejemplo del atributo derivado, será necesario garantizar que ese campo siempre se encuentre sincronizado con la información de la cual se deriva, lo que implica además que las actualizaciones serán más demoradas.

### Buenas prácticas SQL

Cuando el SGBD recibe una consulta SQL, esta pasa por un proceso de análisis, simplificación y optimización. Es decir, el motor antes de ejecutar la consulta trata de optimizarla de manera que se utilice el camino más corto para llegar al resultado.

Sin embargo, eso no significa que no debamos tener cuidado al momento de escribir una consulta. Uno de los mecanismos quizás más efectivos para

lograr el mejor rendimiento del sistema de base de datos es aplicar buenas prácticas al momento de escribir sentencias SQL.



En el siguiente documento se exponen las recomendaciones más importantes a tener en cuenta en la formulación de consultas SQL

[Buenas prácticas SQL \(Encalada, 2017\)](#)

Algunas de las recomendaciones más relevantes se resumen a continuación:

- Evitar el SELECT \*. Traer únicamente los datos necesarios.
- Evitar productos cartesianos.
- Combinar en la cláusula FROM (JOIN)
- Ordenar solo si es necesario
- Cualificar los nombres de columna en consultas multitable
- En subconsultas, si es posible, usar IN en lugar de NOT IN
- Usar EXISTS en lugar de IN
- Evitar DISTINCT si no es necesario.
- Paginar.
- Usar LIKE al final de las condiciones.

### Respaldo y recuperación

Una de las medidas de seguridad que se debe implementar en toda base de datos, es la obtención periódica de respaldos, también llamados copias de seguridad, es decir, obtener en almacenamiento terciario (cintas, discos externos, etc.) una copia de los datos almacenados en la base de datos, a efectos de garantizar la no pérdida de datos en caso de que se produzca un fallo en el servidor, o en caso de que se pierdan los datos por algún otro motivo.



Revise el siguiente recurso que explica el tema del respaldo y recuperación aplicado a bases de datos MySQL. Preste especial atención a los tipos de respaldos que existen y a las

recomendaciones que se deben seguir para asegurar que los respaldos sean efectivos.

Copias de seguridad de bases de datos (Hueso Ibáñez, 2015a, pp. 243-258)

En sistemas con requerimientos de alta disponibilidad se necesita trabajar con motores que permitan realizar respaldos en caliente, es decir, que las copias de seguridad se puedan obtener mientras el servicio está en operación, sin que ello conlleve riesgo de inconsistencia en los respaldos.

Algunas recomendaciones a tener en cuenta son las siguientes:

- Los respaldos deben estar debidamente etiquetados.
- Debe haber múltiples réplicas del mismo respaldo almacenados en lugares diferentes.
- El acceso físico a los respaldos debe estar protegido.
- En muchos casos se contratan servicios de empresas aseguradoras.

La tarea de “Respaldos y recuperación” es una las funciones que debe cumplir el DBA, y aquí también es importante aclarar que, aunque el respaldo se realice siguiendo todas las recomendaciones técnicas, siempre es necesario que periódicamente se realice un ensayo de recuperación de la base de datos, a efectos que validar tanto los respaldos como el procedimiento de recuperación. Es la mejor manera de estar preparados y saber qué hacer ante una caída de la base de datos.

### **Material complementario**

Opcionalmente, y a efectos de reforzar la comprensión de los temas tratados, le recomendamos el siguiente recurso:

- [Copias de seguridad y recuperación de bases de datos \(Hueso Ibáñez, 2015b, pp. 218-224\)](#): afianza la explicación de cómo obtener respaldos de nuestros datos en MySQL. Se centra en llevarlo a la práctica, mediante la presentación de múltiples ejemplos en los que se explica el uso de los comandos necesarios tanto para respaldar la base de datos, como para recuperar la información a partir de un respaldo previo.



## Actividades de aprendizaje recomendadas

- Investigue acerca de otras buenas prácticas SQL.
- Realice una prueba de respaldo y recuperación en una base de datos MySQL.



## Semana 15

---

Durante la presente semana revisaremos conceptos y tecnologías en torno a la gestión de grandes volúmenes de datos. El objetivo es sentar bases que más adelante le permitan profundizar en el estudio y especialización de esta rama asociada a la Ciencia de Datos, que actualmente es muy demandada dentro de las organizaciones.

### Gestión de los grandes volúmenes de información

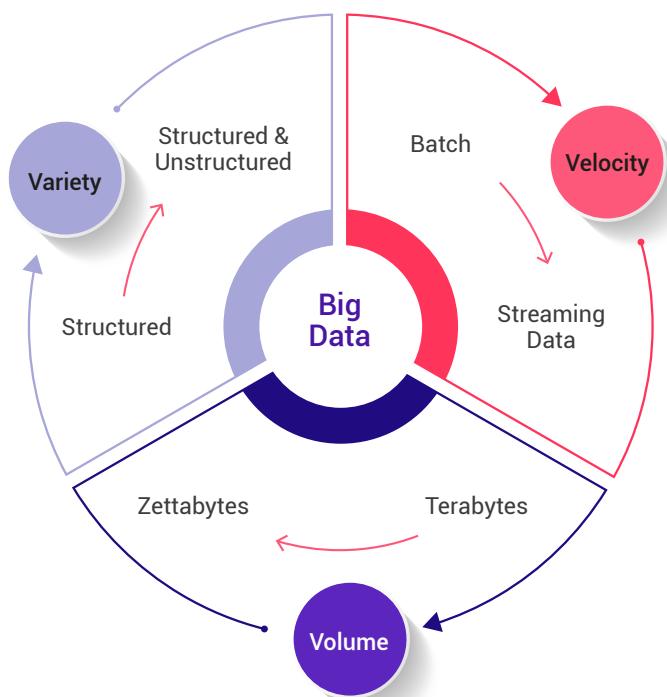
En las semanas 5 y 6, hablamos de las bases de datos NoSQL, y en ese contexto se había explicado que su aparición surgió -entre otras razones- ante las limitaciones de las bases de datos relacionales para manejar grandes volúmenes de datos, o Big Data. De hecho, en un ecosistema Big Data, las bases de datos NoSQL son una parte fundamental.

El concepto de Big Data resulta a veces ambiguo en su definición, ya que es difícil establecer la línea que marca la separación entre lo que es Big Data y lo que no lo es. Es decir, ¿cuáles son las condiciones que debe cumplir un proyecto de analítica de datos, para ser considerado como Big Data? ¿Solo tiene que ver con el volumen de datos? ¿Cuánto es el volumen de datos mínimo para ser Big Data?, etc.

Manyika et al. (2011) define Big Data como un conjunto de datos de tamaño considerable en relación con la capacidad de captura, almacenado, gestión y análisis de las herramientas de base de datos. En cambio, según Salvador (2014) Big Data es la "disponibilidad de grandes cantidades de información en formatos estructurados y desestructurados en tiempo real" (p.5).

La definición quizá más ajustada (al menos intenta serlo) viene de Gartner (s.f.), quien afirma que "son activos de información caracterizados por su alto volumen, velocidad y variedad, que demandan soluciones innovadoras y eficientes de procesado para la mejora del conocimiento y toma de decisiones en las organizaciones.". La figura 34 ilustra justamente este concepto.

**Figura 34**  
Las 3V de Big Data



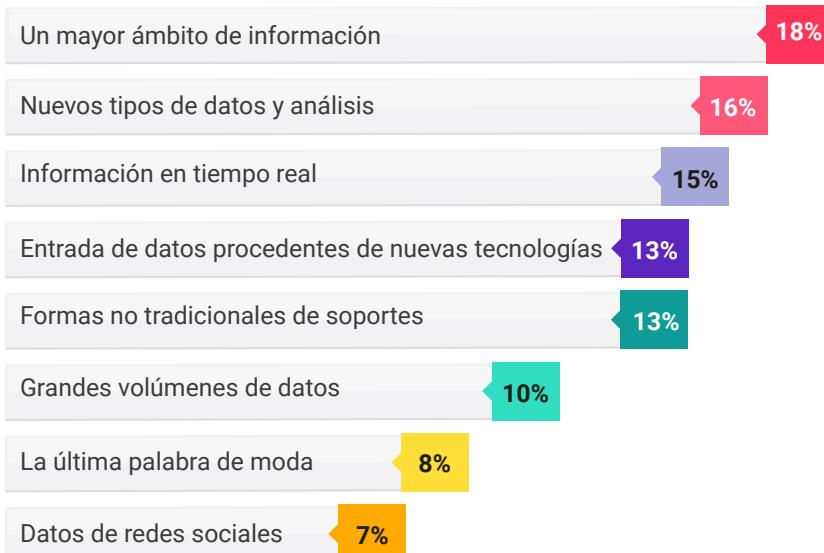
Nota. Tomado de *Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data* (p. 5), por P. C. Zikopoulos, C. Eaton, D. DeRoos, T. Deutsch, y G. Lapis, 2012, McGraw-Hill.

Como se ve, existen distintas definiciones que no son exactamente coincidentes. Lo que queda aún más en evidencia si mencionamos un estudio realizado por IBM Institute for Business Value en colaboración con el Saïd Business School, en el cual se encuestaron a más de 1100 negocios y profesionales de TI de 95 países y decenas de expertos, a quienes se les propuso una serie de características sobre Big Data, y se les pidió que escogieran las dos que mejor describen el concepto. El resultado se muestra en la figura 35.

**Figura 35**

Resultados de estudio sobre “Definición de Big Data”

## Definición de big data



Nota. Adaptado de How To Identify Big Data, por C. Ulger, 2015 ([Enlace web](#))

Es claro, entonces, que definir "Big Data" no resulta sencillo, incluso para los expertos. Digamos que se trata de una nueva perspectiva de la extracción del conocimiento, que se realiza sobre grandes cantidades de datos (volumen), estructurados y no estructurados (variedad), en tiempo real (velocidad) usando herramientas y técnicas avanzadas de almacenamiento, procesamiento y analítica de datos. Carrasco (2017) contribuye con algunas características adicionales que se deben considerar, y que se resumen así:

- Se procesa la información en crudo y se la depura conforme se requiera.
- Se explora y analiza todos los datos disponibles, buscando patrones y relaciones sin necesidad de hipótesis previas.
- Los datos se analizan conforme se van generando, en tiempo real.

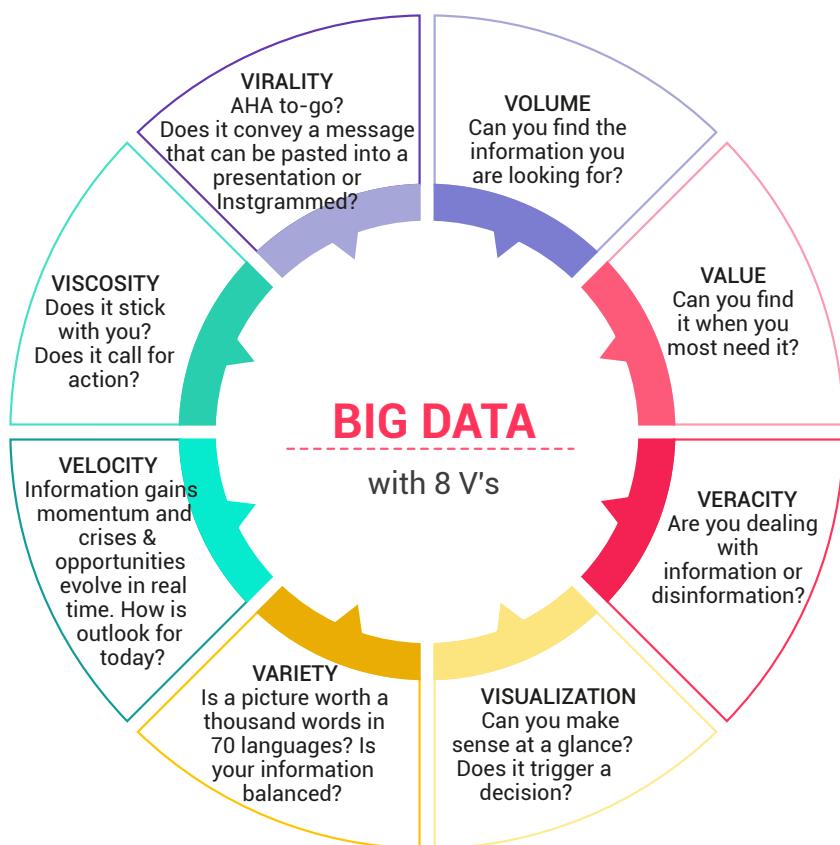


Le invitamos a realizar la siguiente lectura, en la que se busca aclarar mejor el alcance concepto Big Data.

Conceptos principales en relación con el campo de big data (Ríos Insua & Gómez-Ullate Oteiza, 2019, pp. 24-32)

En referencia a las conocidas 'uves' que definen las propiedades Big Data, estas han ido evolucionando con el concepto, y actualmente ya se habla hasta de 8 V's, tal como se expone en la figura 36.

**Figura 36**  
*Las 8 V's de Big Data*



Nota. Tomado de *Big Data with 8v's and today's challenges*, por S. Mathur, 2021 ([Enlace web](#)).

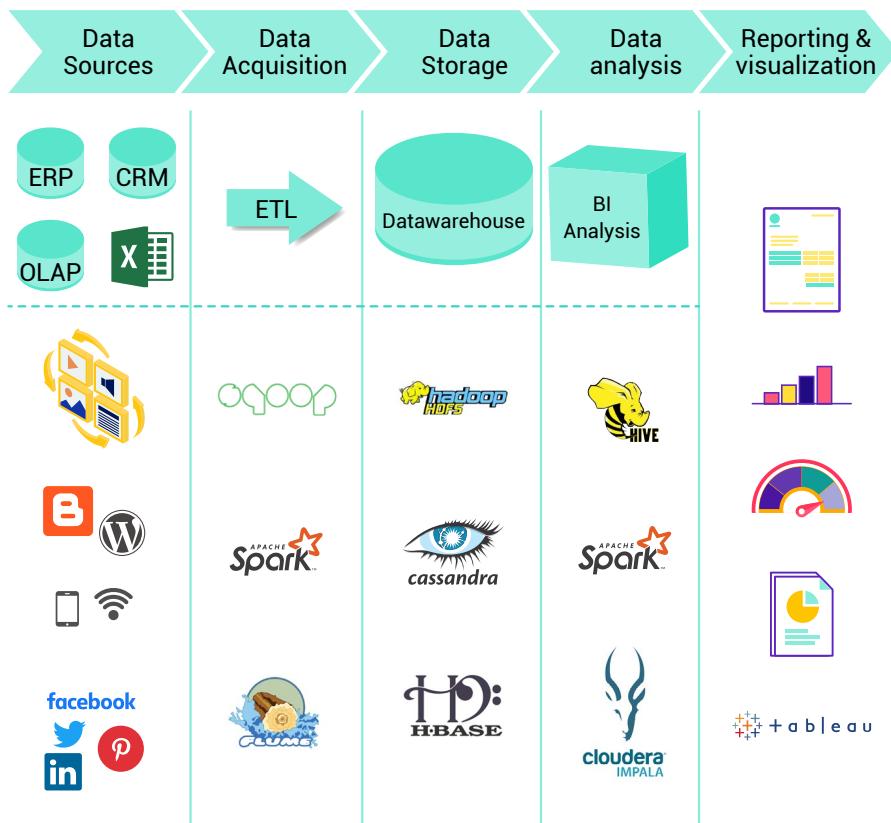
## Ecosistema y tecnologías Big Data

Con la evolución del concepto de Big Data han ido evolucionando también los componentes, los procesos, y las tecnologías asociadas. Es claro que actualmente Big Data no es simplemente un repositorio de grandes volúmenes de datos, sino que es todo un ecosistema de generación de conocimiento a partir de los datos. La *Figura 37* muestra las capas que integran un ecosistema Big Data, que son:

1. **Las fuentes:** el origen de los datos puede ser muy diverso, entre otros están las bases de datos transaccionales, hojas electrónicas, archivos planos, internet, redes sociales, dispositivos inteligentes, sensores, internet de las cosas y muchos otros.
2. **La adquisición:** corresponde a la capa donde se realiza la obtención y preparación de los datos.
3. **El almacenamiento:** los datos preprocesados son cargados a un repositorio desde donde se podrán acceder independientemente de su formato, volumen, frecuencia y origen.
4. **El proceso de análisis:** en esta capa se realiza la exploración de los datos almacenados, mediante modelos, algoritmos y herramientas especializadas, a efectos de obtener y generar información y conocimiento útil.
5. **La presentación y visualización de resultados:** el conocimiento generado es consumido por los interesados a través de informes, o usando herramientas y técnicas de visualización en tiempo real.

**Figura 37**

Capas de un ecosistema Big Data



Nota. Tomado de *Big Data*, por J. Mendoza, J. Abarca-Tipo, y Y. Muñoz-Medrano, 2019 ([Enlace web](#)).

En cuanto a las tecnologías, Big Data ha marcado una importante evolución de estas, sobre todo en lo que corresponde a la adquisición, almacenamiento, y procesamiento de los grandes volúmenes de datos.

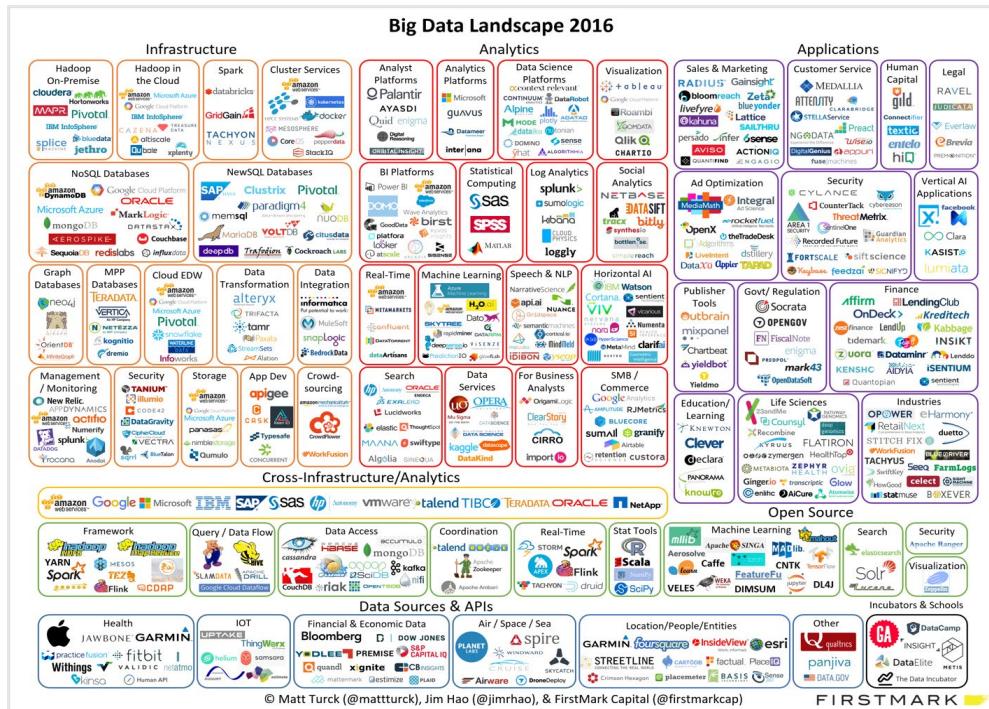


Le invitamos a revisar el siguiente recurso en el que se exponen las principales tecnologías asociadas a Big Data.

Tecnologías del big data (Ríos Insua & Gómez-Ullate Oteiza, 2019, pp. 32-38)

Realmente las tecnologías y herramientas para Big Data son muchas, tal como se ilustra en la figura 38, en la que vemos una panorámica de la inmensa gama de productos y herramientas asociados a las distintas capas de este ecosistema.

**Figura 38**  
Big Data Landscape 2016



Nota. Tomado de *Big Data Landscape 2016 (Version 3.0)*, por M. Turck, J. Hao, y FirstMark Capital, 2016 ([Enlace web](#)).

En lo que corresponde a la capa de presentación, las tecnologías han ido muy a la mano de las soluciones de inteligencia de negocios, donde se han desarrollado herramientas especializadas para la generación de informes mediante técnicas de visualización interactiva en tiempo real.

## Ciencia de datos

Con Big Data, la importancia de los datos ha crecido tanto a lo largo de los años, que en la actualidad ya es considerado un recurso estratégico para las organizaciones y los estados. Su adecuado manejo y aprovechamiento contribuye significativamente al desarrollo y evolución de las sociedades, y representa una gran ventaja competitiva para las empresas. Su influencia

en el desarrollo humano, más la gran cantidad de información que se genera y se almacena a diario, ha impulsado el desarrollo de nuevas tecnologías y modelos avanzados que permiten un adecuado tratamiento, y la obtención de conocimiento que es clave en la toma de decisiones.

La ciencia de los datos es un paso evolutivo en campos interdisciplinarios como el análisis de negocios que incorpora la informática, el modelado, las estadísticas, la analítica y las matemáticas en uno solo proceso (NYU center for Data Science).



Le invitamos a revisar el siguiente recurso en el que se sintetiza el origen, alcance, y las tecnologías en torno al concepto Ciencia de Datos.

[El triangulo de la ciencia de datos](#)

Como vemos, la ciencia de datos es una rama interdisciplinaria que combina la informática, con la matemática-estadística, y con el conocimiento específico del dominio sobre el cual se desarrolla el proceso de análisis. Y al profesional que cumple con esos tres perfiles, se lo conoce como Científico de Datos.

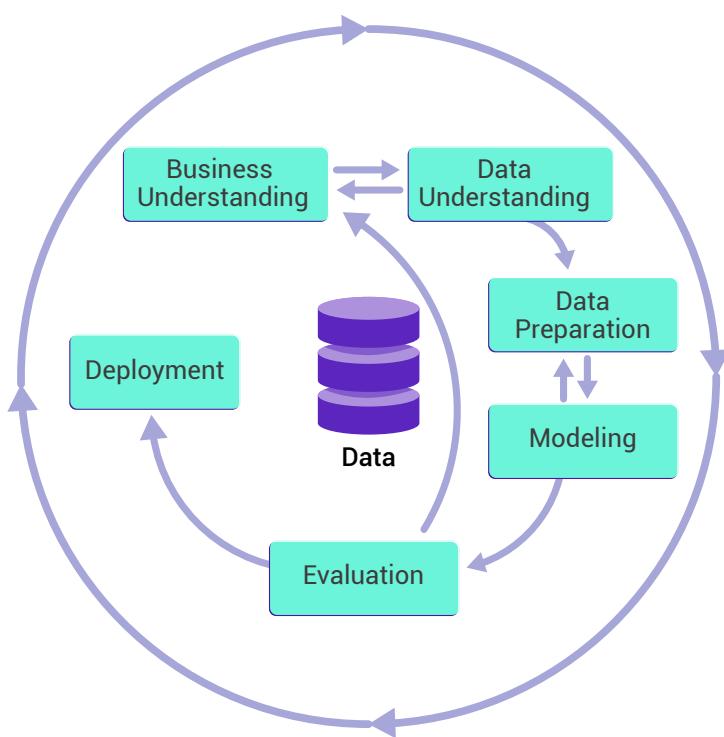
### Científico de datos (Data Scientist)

Sobre este importante rol tan demandado actualmente, Patil & Mason (2015) explican que un científico de datos no hace nada fundamentalmente nuevo. Ya existen estadísticos, analistas y programadores. Lo nuevo es la forma en que combina varias habilidades y disciplinas en una sola profesión. Lo primero y más importante es la matemática (estadística y álgebra lineal, sobre todo). También, necesita habilidades informáticas, incluida la programación y el diseño de la infraestructura, debe ser capaz de obtener datos desde una base de datos en un paquete de analítica y viceversa. Por último, un científico de datos sabe hacer las preguntas correctas y sabe comunicarse; es valorado por su capacidad para crear historias en torno a su trabajo. No vive en un mundo matemático abstracto; sabe asociar los resultados en una historia más amplia, y es consciente que, si sus resultados no conducen a la acción, no tienen sentido.

## Proceso de desarrollo de un proyecto de ciencia de datos

Para el desarrollo de proyectos de ciencia de datos, existen diversos modelos de proceso; uno de los más conocidos es el propuesto por la industria de la Minería de Datos, conocido como Proceso Estándar Intersectorial para Minería de Datos (CRISP-DM por sus siglas en inglés), que establece una metodología de 6 fases que normaliza el proceso de extracción de conocimiento a partir de los datos. La figura 39 ilustra estas fases.

**Figura 39**  
Modelo de proceso CRISP-DM



Nota. Tomado de *Data Analytics Made Accessible*, por A. Maheshwari, 2018.

Basados en la explicación que realiza Maheshwari (2018) sobre CRISP-DM, resumimos aquí algunas ideas clave de cada fase. Empezamos con la (1) comprensión del negocio: significa hacer las preguntas correctas, establecer claramente el problema y el alcance, y demostrar los beneficios del proyecto. Luego pasamos a la (2) comprensión de los datos: aquí se debe ser imaginativo para buscar datos en múltiples fuentes, deben ser datos relevantes y de calidad. Lo siguiente es la (3) Preparación los

datos (preprocesamiento): considerando que esta tarea puede consumir hasta un 70% del tiempo, implica depurar, integrar y validar los datos, es imprescindible asegurar la confiabilidad de los datos. Luego viene el (4) modelado, que conlleva ejecutar muchos algoritmos sobre los datos y usar distintas herramientas buscando encontrar conocimiento importante, requiere paciencia para probar muchos escenarios (diferente combinación de variables y parametrización de algoritmos) hasta encontrar algo relevante. Seguidamente, hay que realizar la (5) evaluación del modelo: no se debe aceptar el modelo inmediatamente, se lo debe someter a múltiples casos de prueba, y si es necesario se lo debe reajustar hasta conseguir niveles de precisión y alcance esperados. Finalmente, viene el (6) despliegue: el modelo debe implementarse e integrarse a los procesos de la organización.

## **Material complementario**

Opcionalmente, y a efectos de reforzar la comprensión de los temas tratados, le recomendamos los siguientes recursos:

- [¿Qué es la Ciencia de Datos, Minería de Datos y Big Data? \(Prieto-Espinosa, 2018\)](#): video que presenta ideas básicas sobre conceptos y disciplinas que han surgido en los últimos años y que abordan el problema del uso eficiente de los datos, con el propósito de sacarles el máximo provecho posible.
- [Arquitecturas y tecnologías para el big data \(Zorrilla & García-Saiz, 2017\)](#): presentación en la que se amplía la explicación de arquitecturas y tecnologías asociadas a Big Data.



### **Actividades de aprendizaje recomendadas**

- Investigue acerca de los usos y aplicaciones actuales de Big Data.
- Analice la importancia de Big Data en torno los conceptos de SmartLand y SmartCities.



## Semana 16

---

### Preparación para la segunda evaluación bimestral

Estamos cerrando el estudio de nuestra asignatura, confiamos en que la presente experiencia de aprendizaje le haya motivado a interesarse aún más por el mundo de las bases de datos. Debe saber que hay un extenso campo de estudio e investigación detrás de este tema, con diversas opciones de especialización, y con múltiples oportunidades laborales. Le animamos a continuar explorando este interesante campo dentro las tecnologías de la información.

Como corresponde, durante esta última semana su dedicación y esfuerzo debe abocarse a preparar el examen del segundo bimestre. Para prepararse de la mejor manera, le sugerimos lo siguiente:

1. Revise los contenidos estudiados durante el segundo bimestre (unidades 2 y 3).
2. Repase las actividades desarrolladas durante todo el bimestre (prácticas, talleres, foros, y cuestionarios).
3. Refuerce el repaso de aquellos temas en los cuales usted vea que no ha logrado alcanzar los resultados esperados. Tome nota de inquietudes en las que necesita una mayor retroalimentación.
4. Aproveche los espacios de tutorías para compartir sus inquietudes con el tutor y recibir la retroalimentación necesaria.

Al igual que en el primer bimestre, el examen estará orientado sobre todo a evaluar su nivel de comprensión, aplicación, y análisis en torno a los temas estudiados.

¡Le deseamos el mejor de los éxitos!



---

## 4. Referencias bibliográficas

---

- Araneda, P. (2020). Base de Datos, el camino de los datos a la información. Recuperado agosto de 2022, disponible en <https://bookdown.org/paranedagarcia/database/>
- Blaney, J. (2017, 7 mayo). Introducción a los Datos abiertos enlazados. Programming Historian. <https://programminghistorian.org/es/lecciones/introduccion-datos-abiertos-enlazados>
- Cabré I Segarra, B., Casas Roma, J., Costal Costa, D., Juanola Juanola, P., Rius Gavidia, A., & Segret I Sala, R. (s. f.). Diseño físico de bases de datos. UOC. <https://rebrand.ly/70025r4>
- Carrasco, R. (2017, 28 febrero). Big Data, Internet of Things and Smart Cities. studylib.es. Recuperado agosto de 2022, de <https://studylib.es/doc/5776559/big-data–internet-of-things-and-smart-cities>
- Connolly, T. M., & Begg, C. E. (2005). Sistemas de bases de datos. Un enfoque práctico para diseño, implementación y gestión (4.a ed.). Pearson Educación.
- Coronel, C., Morris, S., y Rob, P. (2011). Bases de Datos. Diseño, implementación y administración. (9a. ed.). Cengage Learning. <https://visorweb.utpl.edu.ec/reader/bases-de-datos-disenos>
- Coronel, C., & Morris, S. (2019). Database Systems: Design, Implementation, & Management (13.a ed.). Cengage Learning.
- Cuadra, D., Castro, E., & Iglesias, A. M. (2014). Desarrollo de bases de datos: casos prácticos desde el análisis a la implementación (2.a ed.). Editorial Ra-Ma. <https://elibro.net/es/lc/bibliotecautpl/titulos/106429>
- Darmawikarta, D. (2014). Sql for mysql : A beginner's tutorial. Brainy Software. <https://ebookcentral.proquest.com/lib/utpl/detail.action?docID=3003883>

Domenjoud, M. (2012, 12 octubre). Graph databases: an overview. OCTO Talks ! <https://blog.octo.com/en/graph-databases-an-overview/>

Elmasri, R., y Navathe, S. B. (2016). Fundamentals of Database Systems (7a. ed.). Hoboken, Estados Unidos: Pearson. <https://visorweb.utpl.edu.ec/reader/fundamentals-of-database-systems>

Encalada, A. E. [Eduardo Encalada - UTPL]. (2016a, abril 18). Bases de datos: Diseño Físico [Video]. YouTube. <https://rebrand.ly/taw9852>

Encalada, A. E. [Eduardo Encalada - UTPL]. (2016b, abril 22). Bases de datos: Traducir modelo lógico a SGBD [Video]. YouTube. <https://rebrand.ly/youtu3506>

Encalada, A. E. (2017, septiembre). Buenas prácticas SQL. Opus Cum Data. Recuperado agosto de 2022, de <https://rebrand.ly/bpsql3728>

Encalada, A. E. (2017b, septiembre). Técnicas de desnormalización. Opus Cum Data. Recuperado 10 de mayo de 2022, de <https://rebrand.ly/tecdes8291>

Encalada, A. E. (2020, julio). Bases de Datos Semánticas. Opus Cum Data. Recuperado 15 de agosto de 2022, de [https://opuscumdata.com/wp-content/uploads/2022/08/BasesDeDatosSemanticas\\_v1.pdf](https://opuscumdata.com/wp-content/uploads/2022/08/BasesDeDatosSemanticas_v1.pdf)

Encalada, A. E. (2022a, agosto). Ejemplos de comandos de consulta en MongoDB [Diapositivas]. Opus Cum Data. <https://opuscumdata.com/wp-content/uploads/2022/08/EjemplosComandosConsultaMongoDB.pdf>

Encalada, A. E. [Eduardo Encalada - UTPL]. (2021, 16 junio). Proceso de afinación de una base de datos [Vídeo]. YouTube. <https://rebrand.ly/afibd7823>

Encalada, A. E. [Eduardo Encalada - UTPL]. (2022b, agosto 20). Uso de SQL para la obtención de reportes estadísticos [Video]. YouTube. <https://www.youtube.com/watch?v=PuDgQ7n-DF8>

Encalada, A. E. (2022, junio). Índices. Opus Cum Data. Recuperado 15 de agosto de 2022, de [https://opuscumdata.com/wp-content/uploads/2022/08/Indices\\_v1.pdf](https://opuscumdata.com/wp-content/uploads/2022/08/Indices_v1.pdf)

Gartner. (s. f.). Definition of Big Data - Gartner Information Technology Glossary. Recuperado agosto de 2022, de <https://www.gartner.com/en/information-technology/glossary/big-data>

Hueso Ibáñez, L. (2015a). Gestión de bases de datos (2a. ed.).. RA-MA Editorial. <https://elibro.net/es/ereader/bibliotecaupl/62491>

Hueso Ibáñez, L. (2015b). Administración de sistemas gestores de bases de datos.. RA-MA Editorial. <https://elibro.net/es/ereader/bibliotecaupl/62482>

Lauro, C. (2019, 21 febrero). A definition of Data Science. Linkedin. [https://www.linkedin.com/pulse/definition-data-science-carlo-lauro?trk=portfolio\\_article-card\\_title](https://www.linkedin.com/pulse/definition-data-science-carlo-lauro?trk=portfolio_article-card_title)

Maheshwari, A. (2018). Data Analytics Made Accessible (2018.a ed.). Fairfield.

Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., & Hung Byers, A. (2011). Big data: The next frontier for innovation, competition, and productivity. McKinsey Global Institute.

Mathur, S. (2021, febrero). Big Data with 8v's and today's challenges. Medium. <https://shreyanshmathur1998.medium.com/big-data-with-8vs-and-today-s-challenges-9938f00363c5>

Medrano, A. (s. f.). NoSQL. Experto Java Universidad de Alicante. <http://expertojava.ua.es/si/nosql/nosql02.html>

Mendoza, J., Abarca-Tipo, J. & Muñoz-Medrano, Y. (2019, septiembre). Big Data. Medium. <https://medium.com/@jackyabit1/big-data-79455539e1d>

Microsoft. (2022, 24 agosto). Soluciones relacionales y datos NoSQL. Microsoft Docs. Recuperado 30 de agosto de 2022, de <https://docs.microsoft.com/es-es/dotnet/architecture/cloud-native/relational-vs-nosql-data>

Monsó, D. [OpenWebinars]. (2018, 10 abril). Fases de diseño de una base de datos [Vídeo]. YouTube. <https://www.youtube.com/watch?v=HouTrIFc6Qw>

Morocho, J., Romero, A. (2019). Guía didáctica de Fundamentos de Bases de Datos. Loja, Ecuador: Editorial de la Universidad Técnica Particular de Loja.

Patil, D., & Mason, H. (2015). Data Driven. (T. McGovern, Ed.) (Primero ed). Sebastopol: O'Reilly Media.

Prieto-Espinosa, A. [Alberto Prieto Espinosa]. (2018, 7 junio). ¿Qué es la Ciencia de Datos, Minería de Datos y Big Data? [Vídeo]. YouTube. <https://www.youtube.com/watch?v=UiO0WYCgkTI>

Pulido Romero, E. Escobar Domínguez, Ó. & Núñez Pérez, J. Á. (2019). Base de datos. Grupo Editorial Patria. <https://elibro.net/es/ereader/bibliotecaupl/121283>

Quintana, G. (2014). Aprende SQL. Universitat Jaume I. Servei de Comunicació i Publicacions. <https://elibro.net/es/lc/bibliotecaupl/titulos/53252>.

Ríos Insua, D. & Gómez-Ullate Oteiza, D. (2019). Big data: conceptos, tecnologías y aplicaciones. Editorial CSIC Consejo Superior de Investigaciones Científicas. <https://elibro.net/es/lc/bibliotecaupl/titulos/122031>

Salvador, F. (2014). Big Data: ¿la ruta o el destino? Advanced Series Foundation - Tecnología y crecimiento, 3, 26. Recuperado de <https://bit.ly/2QM8Z7L>

Sánchez, J. (s. f.-a). Manual de Gestión de Bases de Datos. Sistemas Gestores de Bases de Datos. JorgeSanchez.net. <https://jorgesanchez.net/manuales/gbd/sgbd.html>

Sánchez, J. (s. f.-b). Manual de Gestión de Bases de Datos. Modelo Relacional. JorgeSanchez.net. <https://jorgesanchez.net/manuales/gbd/modelo-relacional.html>

Sarasa, A. (2016). Introducción a las bases de datos NoSQL usando MongoDB. Editorial UOC. <https://elibro.net/es/lc/bibliotecaupl/titulos/58524>.

Silberschatz, A., Korth, H. F., y Sudarshan, S. (2014). Fundamentos de Bases de Datos (6a. ed.). Madrid, España: McGraw-Hill. <https://visorweb.utpl.edu.ec/reader/fundamentos-de-bases-de-datos-1617718451>.

Turck, M., Hao, J. & FirstMark Capital. (2016, 12 febrero). Big Data Landscape 2016 (Version 3.0). Matt Turck. <https://mattturck.com/big-data-landscape/>

Ulger, C. (2015, febrero). How To Identify Big Data. From Cem. <https://fromcem.com/how-to-identify-big-data/>

Zikopoulos, P. C., Eaton, C., DeRoos, D., Deutsch, T., & Lapis, G. (2012). Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data. McGraw-Hill.

Zorrilla, M., & García-Saiz, D. (2017, enero). Arquitecturas y tecnologías para el big data [Diapositivas]. ocw.unican.es. <https://rebrand.ly/mtvamel>