

INGENIERÍA MECATRÓNICA



DI_CERO

DIEGO CERVANTES RODRÍGUEZ

PROGRAMACIÓN: DESARROLLO BACKEND

SQL

Tipos de Bases de Datos, Nomenclatura y Diagramas

Contenido

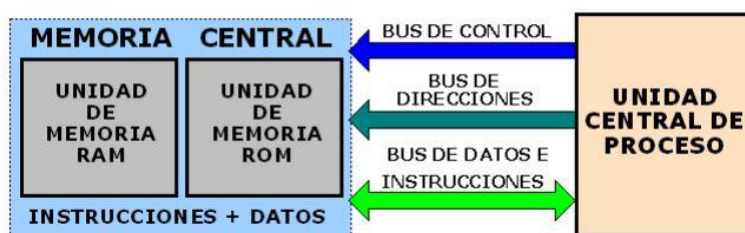
Introducción a las Bases de Datos	2
Tipos de Bases de Datos.....	2
Representación de las Bases de Datos: Nomenclatura de Chen	3
Diagrama ER (Entidad-Relación)	3
Diagrama Físico	8
Normalización: Tabla de Datos a Base de Datos Relacional (RDB)	9
Ejemplo del Diagrama de una Base de Datos: Blog Posts	12
Bases de Datos No Relacionales	15
Jerarquía de Datos de las Bases de Datos No Relacionales: Firestore	17
Conclusiones y Aplicaciones	18
Big Data	18
Data Warehouse	18
Data Mining.....	18
ETL.....	18
Business Intelligence.....	19
Machine Learning.....	19
Data Science.....	19
Referencias.....	19



Introducción a las Bases de Datos

Las bases de datos ayudan a complementar la **arquitectura de Von Neumann**, que es utilizada en ordenadores, la cual **a diferencia de la arquitectura Harvard usada en microcontroladores, utiliza una memoria centralizada para realizar sus funciones**. La necesidad de extender la capacidad de la memoria central es la de conservar los datos más allá de la memoria RAM o ROM, ya que en la arquitectura Von Neumann si se contempla el procesamiento de datos, pero no el almacenamiento de datos persistentes, por lo que es de suma importancia la utilización de las bases de datos.

ARQUITECTURA VON NEUMANN



Para resolver esta situación, donde se busca que de una forma fácil se puedan guardar y extraer datos de información, se obtuvieron dos soluciones:

- **Bases de datos basadas en archivos:** Este método de almacenamiento de datos persistentes consiste en guardar información en un archivo de texto plano, hojas de cálculo, etc. usualmente separados por comas o de alguna otra forma ordenada.
- **Bases de datos basadas en documentos:** En este tipo de base de datos, la unidad básica de almacenamiento es el documento, que puede contener datos en forma de texto, números, listas, objetos JSON (JavaScript Object Notation) y a veces incluso otros documentos anidados.

Tipos de Bases de Datos

Los diferentes tipos de bases de datos existentes son los siguientes:

- **Relacionales o RDB:** **Son bases de datos basadas en documentos** y están gobernadas por las 12 reglas de Edgar Codd, que dan como resultado el álgebra relacional, a través de las cuales se indican las reglas con las que los datos de las RDB se pueden mezclar o relacionar entre sí.
 - **Privadas:** Microsoft SQL Server, Oracle, etc.
 - **Open Source:** PostgreSQL, MySQL, MariaDB, etc.

Ejemplos de bases de datos relacionales



- **No relacionales o NRDB:** Hay varios tipos de bases de datos no relacionales, todas ellas pueden ser muy distintas unas de otras, pero se engloban dentro de la misma categoría de base de datos no relacionales porque utilizan lenguajes NoSQL (Not Only SQL) para sus consultas. Los diferentes tipos de bases de datos no relacionales a grandes rasgos son:
 - Basadas en Clave-Valor, en Documentos, en Grafos, en Memoria, Optimizadas para Búsquedas, etc. Algunos ejemplos de ellas son:
 - Memcached, Cassandra (Facebook), DynamoDB, ElasticSearch, BigQuery, Neo4j (GraphQL), MongoDB, Firestore (Firebase).

Bases de datos no relacionales



- **Auto Administradas:** En este tipo de bases de datos se instala, actualiza y mantiene el software en un ordenador de forma local y la consistencia de datos se realiza de forma manual.
- **Administradas:** Este tipo de base de datos funciona a través de una nube moderna, como las proporcionadas por Amazon, Google, Azure (Microsoft), etc. Para ello la instalación no se realiza de forma local y, por lo tanto, no se mantiene la consistencia de datos de forma manual, sino que se realiza de forma automática por el servicio de la nube.

Representación de las Bases de Datos: Nomenclatura de Chen

- **Entidad:** Se refiere a una **tabla** que almacena datos sobre un tipo de objeto o elemento del mundo real.
 - Cada **fila** en la **tabla** representa una **instancia individual** de esa **entidad**.
 - Cada **columna** en la **tabla** representa un **atributo o característica** de esa **entidad**.
- **Atributo:** Son las **columnas de una tabla** que representan las **características o propiedades** de la **entidad** que está siendo modelada, todas ellas tienen un **nombre y tipo de dato asociado**.
- **Registro:** Representa una **fila perteneciente a una tabla**. También es conocido como "**tupla**" y **contiene los valores** de los **atributos** correspondientes a una **instancia** específica de una **entidad**.

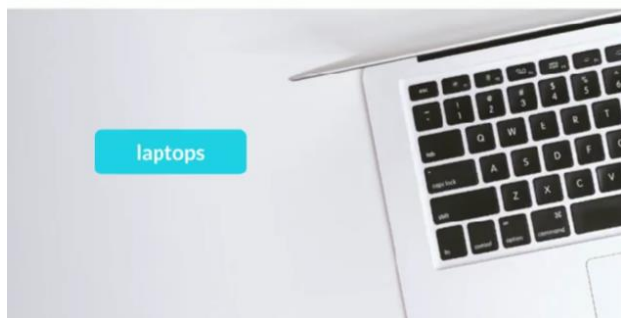
Diagrama ER (Entidad-Relación)

- **Entidad:** Una entidad es algo muy similar a un objeto, el cual se puede asociar con ciertos **atributos (características)**, de la misma forma como se maneja en la programación orientada a objetos (POO).
 - **Atributo:** En el **diagrama ER** se representa por medio de un **óvalo simple** cuando la **entidad** posea **solo una vez ese atributo**, si cuenta con más de uno, este se indica con

dos óvalos anidados que rodeen el nombre del atributo, a esto se le llama **atributo multivalor**.

- **Ejemplo 1:** Cualquier **automóvil** posee **un solo volante**, pero **varias llantas**, por lo cual el atributo “**volante**” será rodeado por un **óvalo simple** y el atributo “**llantas**” se rodeará de un **óvalo doble**.
- **Ejemplo 2:** Ahora se representará a través de un diagrama de Chen las **entidades (objetos) laptops**
 - Cabe mencionar que los atributos donde el nombre se encuentre subrayado se llamarán atributos clave y lo que hacen es diferenciar cada laptop individualmente (**instancia**).
 - Los atributos que tengan un **óvalo con línea punteada** representan los **atributos derivados**, que corresponden a datos que se pueden obtener a través de otros o que pueden tener otros **atributos relacionados**.

Entidades



Atributos



Atributos

no de serie	color	año	pantalla
LKJ789JKAS	gris	2017	AX4829i
KCO3100KJH	negro	2019	AX4930i
NSDJOIH128	negro	2018	AX4930i
09KSIHBD71	gris	2017	AX4829i

- **Tipos de atributos clave:** Los atributos clave pueden ser **naturales o artificiales**. Los **naturales** son pertenecientes al objeto y no se pueden remover; mientras que los **artificiales** son asignados de manera arbitraria.
- **Entidades fuertes:** No dependen de otra **entidad o tabla** para existir, estas se rodean de un cuadrado simple.
 - **Entidades débiles:** Sí dependen de otra **entidad o tabla** para existir, estas se rodean de un cuadrado doble, así como los **atributos multivalor**. Además, existen dos tipos de debilidad:

- **Debilidad por identidad:** Ocurre cuando para que se puedan diferenciar **dos entidades** entre sí, se debe hacer referencia al **atributo clave** de la entidad de la que dependen. *Ej: libro_id, Ejemplares.*
- **Debilidad por existencia:** Sucede cuando las **entidades** pueden tener un **identificador propio**, pero aun así dependen de otra **entidad** para existir. *Ej: id, Ejemplares.*

Entidades débiles



Entidades débiles: identidad

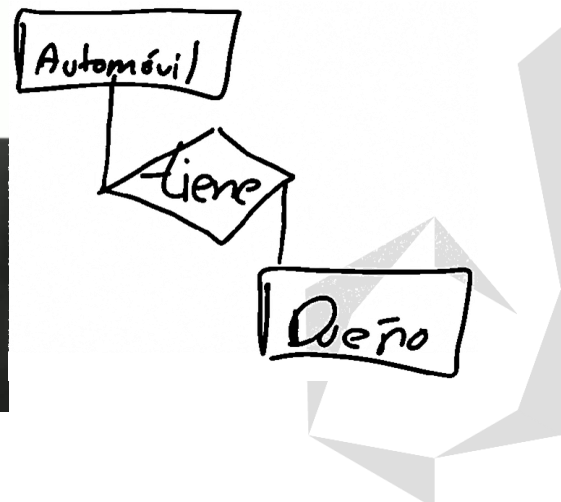
Libros			Ejemplares		
id	título	...	libro_id	localización	edición
LKJ789JKAS	Viaje al cent...	...	LKJ789JKAS	pasillo 1	1
KCO3100KJH	El señor de	KCO3100KJH	pasillo 1	1
NSDJOIH128	De la tierra...	...	NSDJOIH128	pasillo 1	3
09KSIHBD71	Amor en tie...	...	09KSIHBD71	pasillo 1	1

Entidades débiles: existencia

Libros			Ejemplares		
id	título	...	id	localización	edición
LKJ789JKAS	Viaje al cent...	...	JKE7823CLK	pasillo 1	1
KCO3100KJH	El señor de	JKFE1093JD	pasillo 1	1
NSDJOIH128	De la tierra...	...	82938ISHDIK	pasillo 1	3
09KSIHBD71	Amor en tie...	...	838439JHDUI	pasillo 1	1

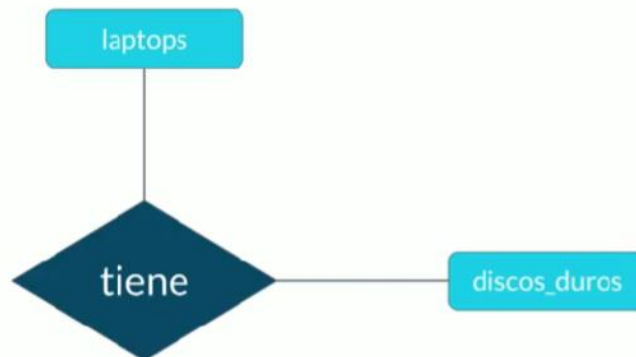
- **Relación:** Es la **conexión** con las que se ligan las distintas **tablas** que describen diferentes aspectos o partes de uno o varios objetos entre sí, **para ello en medio de las relaciones** se utilizan **verbos** que expliquen el **vínculo** de las **entidades** conectadas.

Relaciones



- **Cardinalidad:** Es un concepto donde se indica cuántas **instancias (filas)** de una **entidad** están relacionadas con cuántas **filas** de la otra **tabla**. Para ello se utiliza el concepto de **verbo de conexión** para describir dicha **relación** de forma lógica.

Relaciones

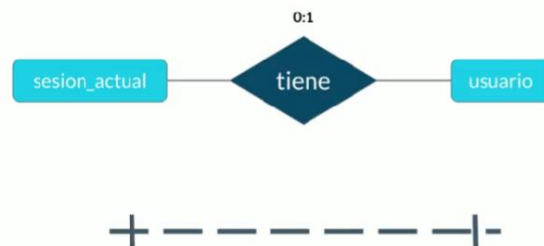


- Se debe indicar la **cardinalidad** en ambos lados de la **conexión**, ya que el número de sus **instancias** puede variar dependiendo de la **entidad** a la que nos referimos. Se maneja cierta nomenclatura en el **diagrama Entidad-Relación** para denotarlo, la cual se encuentra separada por dos puntos y está encima del **verbo de conexión**.

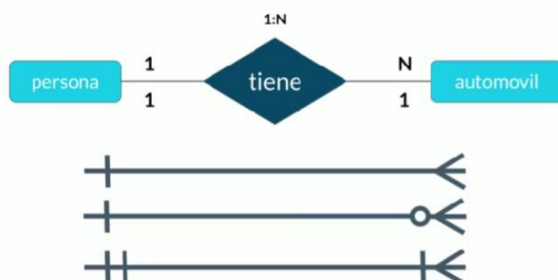
Cardinalidad: 1 a 1



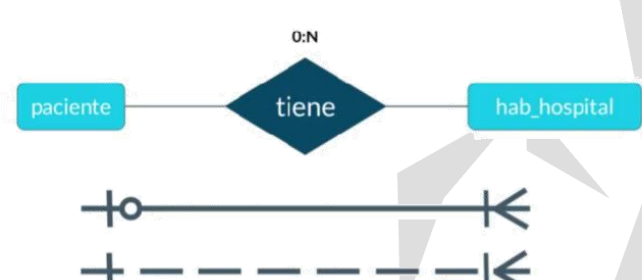
Cardinalidad: 0 a 1



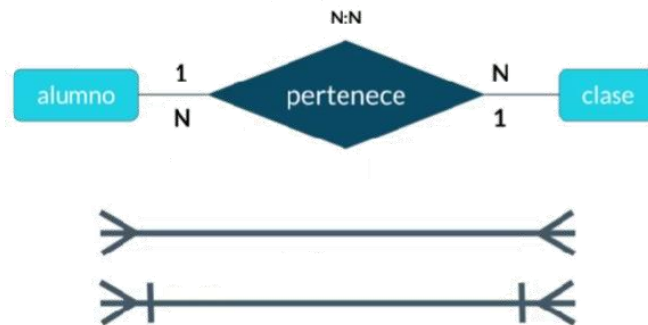
Cardinalidad: 1 a N



Cardinalidad: 0 a N



Cardinalidad: N a N



Todos los conceptos explicados previamente que describen los datos almacenados en una base de datos relacional se deben plasmar en un **diagrama ER (Entidad-Relación)**, para ello se utiliza la siguiente simbología:

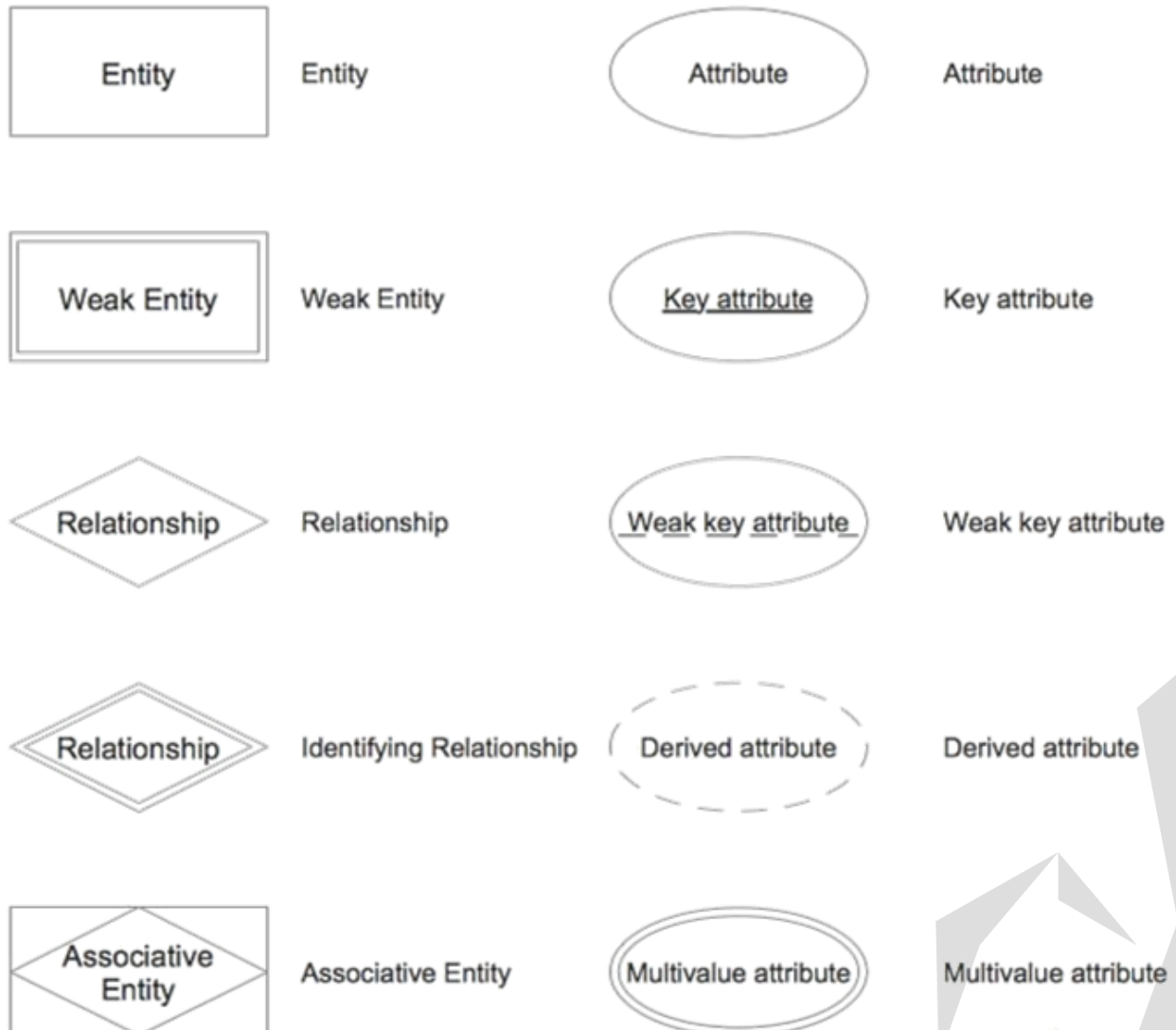


Diagrama Físico

Además del **diagrama ER (entidad-relación)** existe otro llamado **diagrama físico (que se deriva del ER)**, el cual es más específico ya que **menciona los tipos de datos**, que pueden ser los siguientes:

- **Texto:**
 - **Char(n):** Minimiza el espacio de memoria a solo los caracteres que ocupa el texto.
 - **VarChar(n):** Utiliza el espacio de memoria de forma dinámica, reservando como mínimo un espacio de memoria y extendiéndolo si es necesario hasta 255 caracteres.
 - **Text:** Reserva el espacio de memoria para cadenas de caracteres (palabras u oraciones) muy grandes.
- **Números:**
 - **Enteros:** *Integer*, *BigInt* y *SmallInt*.
 - **Decimales:** Se declara *Decimal(n, s)* o *Numeric(n, s)*, ambos son esencialmente lo mismo, donde n es el número y s indica cuantos decimales aparecen.
- **Fecha/Hora:**
 - **Date:** Contiene año, fecha y día.
 - **Time:** Contiene solo la hora.
 - **Datetime y Timestamp:** Contienen la fecha y la hora.
- **Lógicos:**
 - **Boolean:** Puede adoptar valores true (1) o false (0).

Tipos de dato

Texto	Números	Fecha/hora	Lógicos
CHAR(n)	INTEGER	DATE	BOOLEAN
VARCHAR(n)	BIGINT	TIME	
TEXT	SMALLINT	DATETIME	
	DECIMAL(n, s)	TIMESTAMP	
	NUMERIC (n, s)		

Además del tipo de dato, se indican las restricciones (reglas) de la base de datos que delimitan el tipo de dato que admite, cuántos datos admite, etc.

Constraints (Restricciones)

Constraint	Descripción
NOT NULL	Se asegura que la columna no tenga valores nulos
UNIQUE	Se asegura que cada valor en la columna no se repita
PRIMARY KEY	Es una combinación de NOT NULL y UNIQUE
FOREIGN KEY	Identifica de manera única una tupla en otra tabla
CHECK	Se asegura que el valor en la columna cumpla una condición dada
DEFAULT	Coloca un valor por defecto cuando no hay un valor especificado
INDEX	Se crea por columna para permitir búsquedas más rápidas

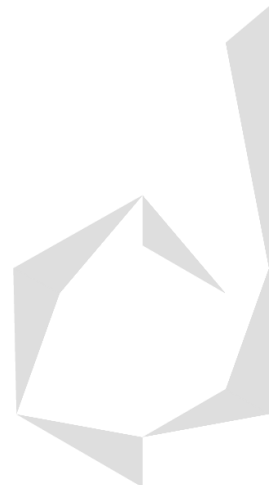
- **Índice:** Es un elemento que **permite realizar búsquedas de datos** en las **columnas** de la **tabla** de una **base de datos**. La desventaja que tiene el realizar búsquedas a través del **index** es que ocupan espacio adicional en el disco, pueden volver lento el procesamiento y requieren mantenimiento al realizar inserciones, actualizaciones y eliminaciones de datos. Por lo que más que nada se utiliza cuando en una base de datos se estarán realizando consultas constantes, pero no se introducirán datos nuevos de forma continua.

Normalización: Tabla de Datos a Base de Datos Relacional (RDB)

La normalización es un proceso en el diseño de **bases de datos relacionales** que busca optimizarlas, **dividiendo sus tablas en partes más pequeñas y relacionándolas entre sí**, reduciendo así la redundancia (que no se repitan datos) y mejorando su integridad. Por ejemplo, al realizar la separación de los **atributos multivaluados**.

Este proceso permite optimizar la estructura de una base de datos a partir de una **tabla** que la represente, separándola en **entidades, atributos y relaciones** más pequeñas. Para ello se aplican las 12 reglas del álgebra relacional de Codd, también llamadas formas normales o FN.

Normalización



A continuación, se denotará este concepto con un ejemplo, donde partiendo de una tabla de datos, esta se reorganizará para ser normalizada, convirtiéndola así en una **base de datos relacional**:

Sin normalizar

alumno	nivel_curso	nombre_curso	materia_1	materia_2
Juanito	Maestría	Data engineering	MySQL	Python
Pepito	Licenciatura	Programación	MySQL	Python

Las formas normales que se siguen para normalizar la tabla son las siguientes:}

- **1FN (Primera Forma Normal) - Atributos atómicos:** Esta norma indica que **no se pueden tener columnas repetidas**, osea que sus **atributos deben ser atómicos**. Un atributo es atómico si sus elementos son simples e indivisibles.

alumno	nivel_curso	nombre_curso	materia_1	materia_2
Juanito	Maestría	Data engineering	MySQL	Python
Pepito	Licenciatura	Programación	MySQL	Python

alumnos				
alumno_id	alumno	nivel_curso	nombre_curso	materia
1	Juanito	Maestría	Data engineering	MySQL
1	Juanito	Maestría	Data engineering	Python
2	Pepito	Licenciatura	Programación	MySQL
2	Pepito	Licenciatura	Programación	Python

- **2FN (Segunda Forma Normal) - Clave Única:** Esta norma indica que cada **fila** de la tabla debe depender de una clave única, si no es posible, se debe separar en **entidades** distintas que estén **relacionadas** por el valor de una **columna**.

alumnos				
alumno_id	alumno	nivel_curso	nombre_curso	materia
1	Juanito	Maestría	Data engineering	MySQL
1	Juanito	Maestría	Data engineering	Python
2	Pepito	Licenciatura	Programación	MySQL
2	Pepito	Licenciatura	Programación	Python

alumnos			
alumno_id	alumno	nivel_curso	nombre_curso
1	Juanito	Maestría	Data engineering
2	Pepito	Licenciatura	Programación

materias		
materia_id	alumno_id	materia
1	1	MySQL
2	1	Python
3	2	MySQL
4	2	Python

- **3FN (Tercera Forma Normal) - Campos Clave Sin Dependencias:** Esta norma indica que solo una de las **columnas** de una **tabla** debe tener información real, mientras que **los demás atributos** deben hacer referencia a otras **entidades** para **relacionarlas** y separar los datos lo más posible, evitando dependencias innecesarias entre ellos.

alumnos			
alumno_id	alumno	nivel_curso	nombre_curso
1	Juanito	Maestría	Data engineering
2	Pepito	Licenciatura	Programación

alumnos		
alumno_id	alumno	curso_id
1	Juanito	1
2	Pepito	2

cursos		
curso_id	nivel_curso	nombre_curso
1	Maestría	Data engineering
2	Licenciatura	Programación

materias		
materia_id	alumno_id	materia
1	1	MySQL
2	1	Python
3	2	MySQL
4	2	Python

materias		
materia_id	alumno_id	materia
1	1	MySQL
2	1	Python
3	2	MySQL
4	2	Python

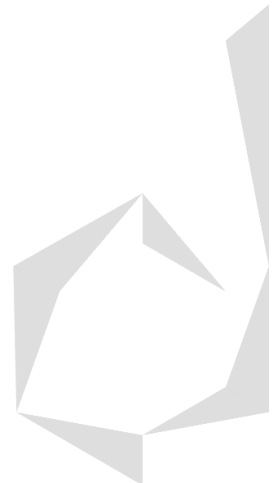
- **4FN (Cuarta Forma Normal) - Campos Multivaluados:** Esta norma indica que **los atributos multivaluados** deben ser identificados y separados por una **clave única**, evitando así que se repitan en cada **entidad**.

alumnos		
alumno_id	alumno	curso_id
1	Juanito	1
2	Pepito	2

cursos		
curso_id	nivel_curso	nombre_curso
1	Maestría	Data engineering
2	Licenciatura	Programación

materias	
materia_id	materia
1	MySQL
2	Python

materias_por_alumno		
mpa_id	materia_id	alumno_id
1	1	1
2	2	1
3	1	2
4	2	2



Ejemplo del Diagrama de una Base de Datos: Blog Posts

Para crear una base de datos, primero que nada, debemos pensar en las **entidades** que se utilizarán en ella y posteriormente deberemos pensar en los **atributos** que le pertenecen, **todo esto se coloca en un diagrama ER (entidad-relación)** para modelar cómo los datos se **relacionan** entre sí. En el caso de un sitio de blog con posts, sus características son las siguientes:

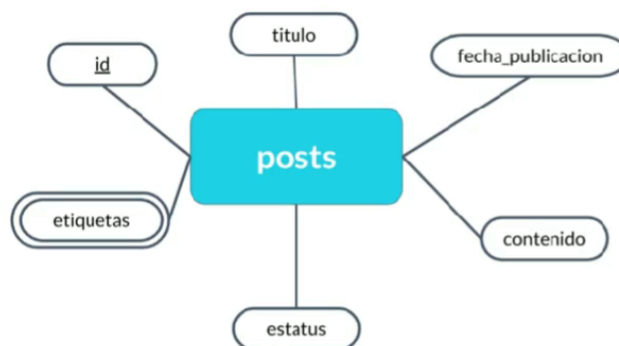
- **Entidades:**
 - Posts (Publicaciones).
 - Usuarios.
 - Comentarios.
 - Categorías.
 - Etiquetas.

Diagrama ER: Platziblog



- **Atributos de las Entidades:**
 - **Atributos Entidad Posts:**
 - Título.
 - Fecha_publicacion.
 - Contenido.
 - Estatus (**Check** Activo o Inactivo).
 - Etiquetas (Categoría interna).
 - Id (Clave única) (**Primary Key o PK**).

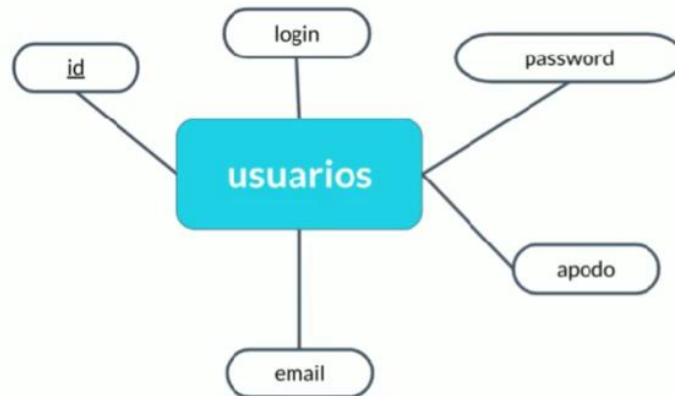
Entidades Platziblog



○ **Atributos Entidad Usuarios:**

- Login (Nombre de Usuario) (**Not Null o NN**).
- Password (**Not Null o NN**).
- Nickname (**Not Null o NN**).
- Email (**Not Null o NN y Unique**).
- Id (**Primary Key o PK**).

Entidades Platziblog

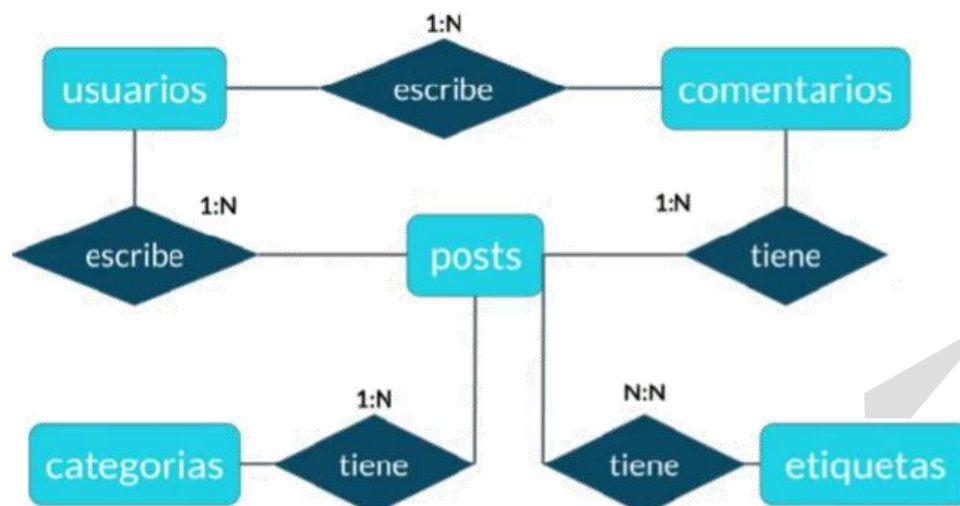


• **Diagrama ER (Entidad-Relación):**

○ **Relaciones y Cardinalidad:**

- Un usuario tiene (puede escribir) varios posts.
- Un usuario tiene (puede escribir) varios comentarios.
- Un post tiene varios comentarios.
- Una categoría tiene (engloba) varios posts.
- Un post tiene varias etiquetas y una etiqueta tiene (engloba) varios posts.

Diagrama ER: Platziblog



- **Diagrama Físico (Tipo de Dato y Constraints):**

- **Tipos de Datos:**

Texto	Números	Fecha/hora	Lógicos
CHAR(n)	INTEGER	DATE	BOOLEAN
VARCHAR(n)	BIGINT	TIME	
TEXT	SMALLINT	DATETIME	
	DECIMAL(n, s)	TIMESTAMP	
	NUMERIC (n, s)		

- **Constraints (Restricciones):**

- **PK: Primary Key.**

- Esta clave debe estar **ligada** con una **foreign key** de alguna **entidad** que sea dependiente de ella.

- **FK: Foreign Key.**

- Esta clave debe estar **ligada** con una **primary key** para ver de qué **entidad** depende, para ello nos debemos fijar en la **cardinalidad** del diagrama.

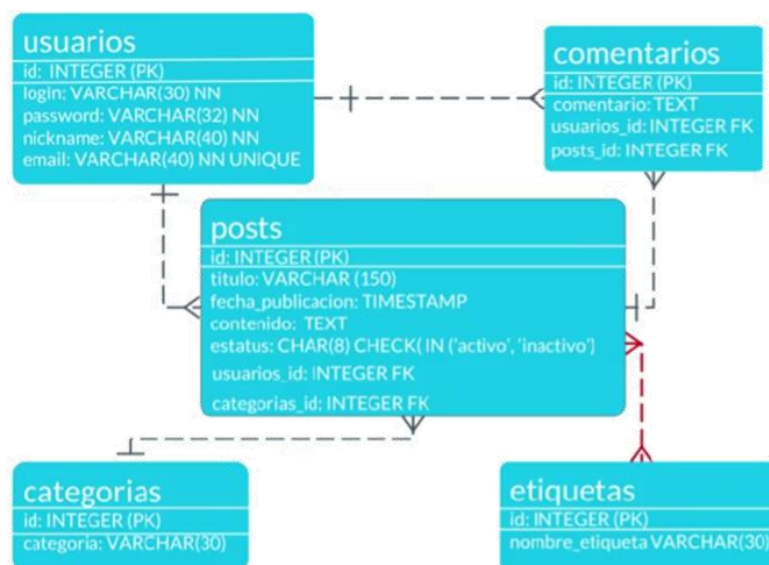
- **NN: Not Null.**

- **UNIQUE.**

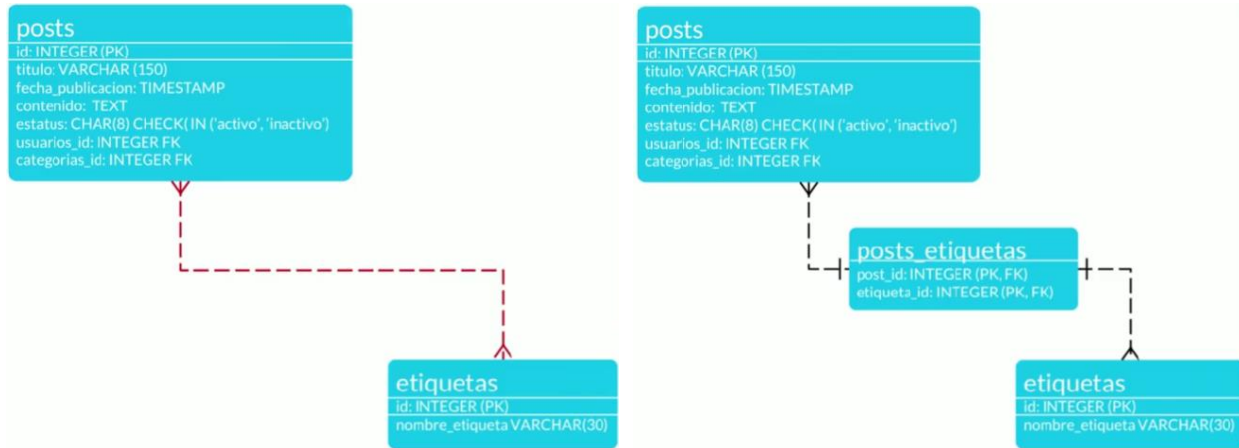
- **CHECK.**

Constraint	Descripción
NOT NULL	Se asegura que la columna no tenga valores nulos
UNIQUE	Se asegura que cada valor en la columna no se repita
PRIMARY KEY	Es una combinación de NOT NULL y UNIQUE
FOREIGN KEY	Identifica de manera única una tupla en otra tabla
CHECK	Se asegura que el valor en la columna cumpla una condición dada
DEFAULT	Coloca un valor por defecto cuando no hay un valor especificado
INDEX	Se crea por columna para permitir búsquedas más rápidas

- **Diagrama relacional global sin cardinalidad N:N:**



- **Diagrama N:N Intermedio o de Pivote:** Este se utiliza cuando se tiene una relación con cardinalidad de N:N entre dos entidades, para ello se debe agregar un diagrama intermedio que relacione ambos id. Es importante mencionar que para crear las llaves únicas de estas relaciones se deben combinar ambos id.



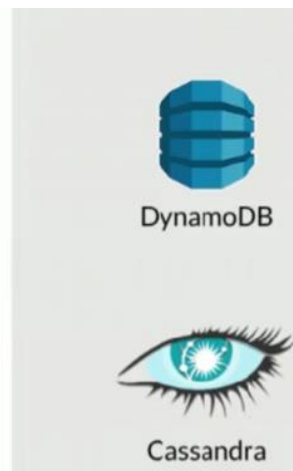
Bases de Datos No Relacionales

Como ya se había mencionado previamente, las bases de datos no relacionales o NRDB (Non Relational Data Bases) no se conforman de un solo tipo de bases de datos, sino de varios, y aunque puedan ser muy distintas unas de otras, todas se engloban dentro de la misma categoría de base de datos no relacional que utilizan lenguajes NoSQL (Not Only SQL). Los diferentes tipos de bases de datos no relacionales son:

- **NRDB Basadas en Clave-Valor:** Estas bases de datos no relacionales están hechas con el objetivo de guardar y extraer datos de forma rápida, todo a través de una clave. La particularidad que tienen es que varios datos están ligados a un mismo id y para hacer consultas dentro de ese grupo de datos se debe utilizar una clave adicional llamada hash o hasmap.
 - Algunos ejemplos de estas bases de datos no relacionales son DynamoDB de AWS y Cassandra de Facebook.

Clave - valor

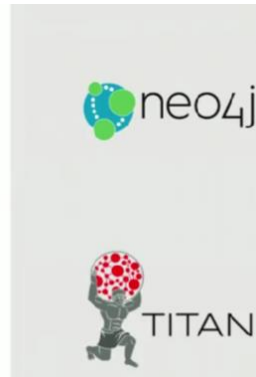
Son ideales para almacenar y extraer datos con una clave única. Manejan los diccionarios de manera excepcional.



- **NRDB Basadas en Grafos:** Los grafos se componen de nodos o entidades (tablas) que tienen relaciones muy complejas con otras y generalmente se relacionan todos contra todos, su mayor uso es en el de la creación de inteligencia artificial o redes neuronales.

Basadas en grafos

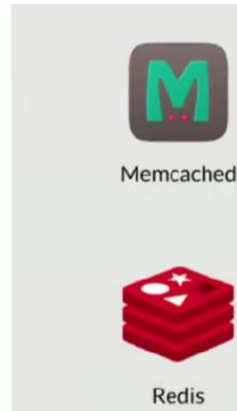
Basadas en teoría de grafos sirven para entidades que se encuentran interconectadas por múltiples relaciones. Ideales para almacenar relaciones complejas.



- **NRDB Basadas en Memoria:** Este tipo de base de datos son sumamente rápidas, pero tienen la gran desventaja de que son volátiles.

En memoria

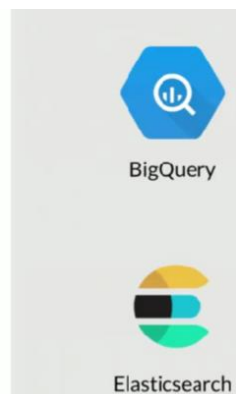
Pueden ser de estructura variada, pero su ventaja radica en la velocidad, ya que al vivir en memoria la extracción de datos es casi inmediata.



- **NRDB Optimizadas para Búsquedas:** Este tipo de base de datos pueden ejecutar Queries muy complejos de forma muy rápida a grandes repositorios de datos históricos que almacenan un gran volumen de datos, son muy utilizadas en aplicaciones de business intelligence y machine learning.
 - Algunos ejemplos de estas bases de datos no relacionales son BigQuery de Google y Elasticsearch.

Optimizadas para búsqueda

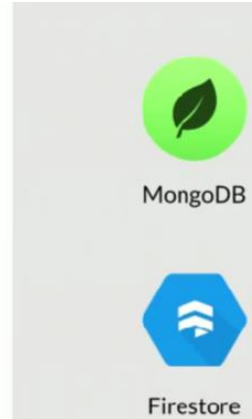
Pueden ser de diversas estructuras, su ventaja radica en que se pueden hacer queries y búsquedas complejas de manera sencilla.



- **NRDB Basadas en Documentos:** El concepto de Documento es mayormente utilizado para referirnos a archivos de tipo JSON (JavaScript Object Notation) o XML.
 - Algunos ejemplos de estas bases de datos no relacionales son MongoDB y Firestore de Google.

Basados en documentos

Son una implementación de clave valor que varía en la forma semiestructurada en que se trata la información. Ideal para almacenar datos JSON y XML.



Jerarquía de Datos de las Bases de Datos No Relacionales: Firestore

En las bases de datos no relacionales, en vez de que se cuente con **tablas (entidades)**, **atributos (columnas)**, **relaciones (conexiones)**, etc. su estructura se basa en **colecciones de datos**, que son el equivalente a las **entidades**, las cuales clasifican los distintos documentos que contienen estructuras JSON que asocian cada **valor** con una **clave**.

Jerarquía de datos en firestore



Cuando trabajamos con bases de datos basadas en documentos como Firestore, cambiaremos el concepto de las **tablas** por las **colecciones** y las **tuplas (filas)** por los **documentos**.

- **Tabla** → **Colección**.
- **Tupla** → **Documento**.

Además, en el contexto de las **colecciones**, identificamos dos categorías principales:

- Las "Top level collections" o colecciones de nivel superior.
- Y las "subcollections" o subcolecciones, que se incorporan dentro de otra colección.

No existe una regla estricta para determinar si la colección debe ser de nivel superior o una subcolección al crear una base de datos basada en documentos; más bien, esta decisión depende del caso de uso específico.

Una consideración clave al diseñar la base de datos es anticipar cómo se extraerán los datos. En el contexto de una aplicación, es útil pensar en términos de las vistas que se mostrarán en un momento específico. En otras palabras, al estructurar la base de datos, debemos asegurarnos de que refleje o contenga, al menos, todos los datos necesarios para satisfacer los requisitos visuales de nuestra aplicación en un momento dado.

Esta regla se aplica salvo algunas excepciones, como cuando se tiene una entidad que necesita existir y modificarse de manera constante e independiente de otras colecciones.

Conclusiones y Aplicaciones

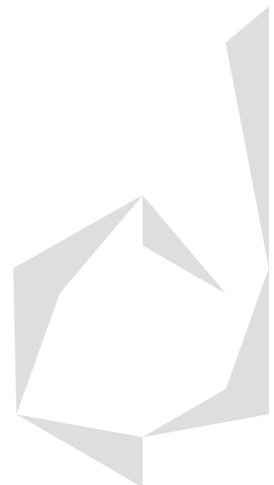
Al final, no se debe elegir una sola base de datos para que funcione con un proyecto en específico, sino que, conociendo el funcionamiento de cada tipo, se pueden utilizar distintas bases de datos que tengan propósitos específicos, como realizar reportes o análisis de datos con SQL o permitir un ingreso y extracción de datos rápida con NoSQL.

Big Data

Data Warehouse

Data Mining

ETL



Business Intelligence

Machine Learning

Data Science

Referencias

Platzi, Israel Vázquez, “Curso de Fundamentos de Bases de Datos”, 2018 [Online], Available: <https://platzi.com/new-home/clases/1566-bd/19781-bienvenida-conceptos-basicos-y-contexto-historico/>

