UNIVERSIDAD DE CUNDINAMARCA
Generación Siglo 21

DOCKER Y BASES DE DATOS

PRESENTADO POR:

NEISSY ALEJANDRA MEDINA CUBILLOS

PRESENTADO A:

WILLIAM ALEXANDER MATALLANA

UNIVERSIDAD DE CUNDINAMARCA

EXTENSION CHIA

INGENIERIA DE SISTEMAS

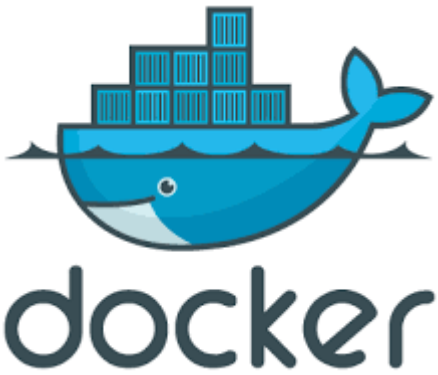
LINEA DE PROFUNDIZACION III

2025

UNIVERSIDAD DE CUNDINAMARCA
Generación Siglo 21

TABLA DE CONTENIDO

1. QUE ES DOCKER:



Docker es una plataforma de software que le permite crear, probar e implementar aplicaciones rápidamente. Docker empaqueta software en unidades estandarizadas llamadas contenedores que incluyen todo lo necesario para que el software se ejecute, incluidas bibliotecas, herramientas de sistema, código y versión ejecutable. Con Docker, puede implementar y ajustar la escala de aplicaciones rápidamente en cualquier entorno con la certeza de saber que su código se ejecutará.

Proporciona un motor de contenedores (Docker Engine), herramientas de construcción de imágenes y un ecosistema de distribución para flujos Dev/Test/Prod rápidos y portables.
El sistema de software de TI llamado "Docker" es la tecnología de organización en contenedores que posibilita la creación y el uso de los contenedores de Linux

¿Cómo funciona?

1. **Desarrollo:**

Los desarrolladores crean una aplicación y definen las dependencias necesarias en un Dockerfile.

2. **Construcción de la imagen:**

El Docker Daemon lee el Dockerfile para construir una imagen de contenedor, que es como una instantánea portable.

3. **Creación del contenedor:**

A partir de la imagen, se crean los contenedores, que son instancias de ejecución de la aplicación.

4. **Ejecución:**

Los contenedores se pueden ejecutar en cualquier sistema compatible con Docker, ya sea localmente o en la nube, con la garantía de que la aplicación funcionará igual.

❖ **Ventajas de usar Docker:**

- **Consistencia:**

Asegura que la aplicación funcione igual en todos los entornos (desarrollo, pruebas, producción).

- **Portabilidad:**

Los contenedores son autónomos y se pueden mover fácilmente entre máquinas o plataformas.

- **Eficiencia:**

Son más ligeros que las máquinas virtuales, ya que comparten el mismo kernel del sistema operativo del host, consumiendo menos recursos.

- **Agilidad:**

Permite la implementación y escalado rápido de aplicaciones, facilitando el desarrollo ágil y DevOps.

- **Microservicios:**

  Es ideal para arquitecturas de microservicios, donde se puede empaquetar y desplegar cada servicio de forma independiente.

2. EN QUE SE DIFERENCIA DE UNA MAQUINA VIRTUAL

En una VM cada instancia trae su propio sistema operativo invitado y kernel sobre un hipervisor, mientras que en Docker múltiples contenedores comparten el kernel del host mediante virtualización a nivel de SO (OS-level).

El resultado práctico es que contenedores arrancan en segundos y usan menos recursos, ideales para microservicios, mientras que VMs ofrecen aislamiento más fuerte al encapsular un SO completo cuando se requiere seguridad o kernels distintos

VMs ejecutan un SO invitado completo sobre un hipervisor; contenedores usan virtualización a nivel de SO compartiendo el kernel del host, por eso son más ligeros y rápidos en arranque.

VMs ofrecen aislamiento más fuerte a nivel de SO; contenedores son ideales para microservicios, CI/CD y despliegues elásticos por densidad y velocidad

ASPECTO	DOCKER	MAQUINA VIRTUAL
Virtualización	A nivel de sistema operativo; los contenedores comparten el kernel del host	A nivel de hardware; cada VM ejecuta su propio sistema operativo invitado.
Kernel	Comparte el kernel del host.	Incluye su propio kernel dentro del SO invitado.
Aislamiento	Aislamiento por namespaces y cgroups; más ligero.	Aislamiento fuerte al encapsular un SO completo.
Arranque	Segundos.	Decenas de segundos a minutos.
Uso de recursos	Ligero; alta densidad de ejecución por host.	Más pesado; menor densidad por host.
Tamaño típico	MBs a cientos de MBs (según imagen base y dependencias).	GBs (imagen/SO invitado completo).
Rendimiento	Casi nativo por menor sobrecarga.	Overhead mayor por hipervisor y SO invitado.
Persistencia de datos	Con volúmenes y servicios externos; el contenedor es efímero.	Disco virtual propio por VM (persistente).
Portabilidad	Alta entre hosts con el mismo runtime de contenedores.	Alta entre hipervisores compatibles o nubes, pero con imágenes más pesadas.
Seguridad	Buena, pero comparte kernel; requiere	Aislamiento más fuerte al separar kernels; mayor

	endurecimiento y buenas prácticas.	superficie por SO completo.
Casos de uso típicos	Microservicios, CI/CD, escalado elástico, apps stateless, entornos homogéneos.	SO distinto al host, legacy, aislamiento fuerte, cargas pesadas o específicas.
Gestión	Declarativa (Dockerfile/Compose), rápida y reproducible.	Gestión del SO invitado (parches, drivers) y del hipervisor.
Redes	Redes lógicas (bridge/overlay), DNS interno por nombre de servicio.	Redes virtuales completas a nivel de VM.
Almacenamiento	Volúmenes gestionados por el runtime; recomendados para producción.	Discos virtuales (VMDK, VHD, etc.) y almacenamiento del hipervisor.
Orquestación	Compose, Swarm, Kubernetes.	Orquestadores de VM, nubes y herramientas de virtualización.
Actualizaciones	Rebuild/pull de imágenes inmutables; despliegues azules/verdes sencillos.	Actualizaciones de SO invitado y plantillas de VM; más costoso.
Observabilidad	Logs/metrics por contenedor; integración nativa con plataformas cloud-native.	Logs/metrics por VM; enfoque tradicional de monitorización.
Complejidad operativa	Baja a media; stacks declarativos y reproducibles.	Media a alta; gestión de múltiples SOs invitados.

3. QUE ES UNA IMAGEN EN DOCKER Y COMO SE RELACIONA EN UN CONTENDOR

Una imagen Docker es una plantilla de solo lectura, inmutable y autónoma que contiene todo lo necesario para ejecutar una aplicación (código, bibliotecas, dependencias y configuraciones). Esta imagen se vuelve un contenedor, que es una instancia en ejecución de esa imagen, cuando se lanza. Piensa en la imagen como los planos de construcción y el contenedor como la casa construida a partir de esos planos.

¿Cómo se relacionan una imagen y un contenedor?

La imagen es el "qué": Define el entorno de la aplicación y sus componentes. Contiene el sistema operativo base, las librerías, el código de la aplicación y cualquier otra dependencia que necesite.

El contenedor es el "cómo": Es la instancia viva de la imagen, un paquete ejecutable. Cuando la imagen se "instancia", se crea un contenedor que puede ejecutar la aplicación en un entorno aislado.

Características clave de las imágenes y su relación con los contenedores:

- Inmutables: Una vez creada, una imagen no puede ser modificada. Esto garantiza consistencia.
- Capas: Las imágenes están compuestas por capas, lo que optimiza el almacenamiento y la transferencia de datos.
- Plantillas: A partir de una imagen, se pueden crear múltiples contenedores, cada uno de los cuales se ejecutará en un entorno consistente.
- Aislamiento: Los contenedores aislados ejecutan aplicaciones de forma independiente, compartiendo el núcleo del sistema operativo del host, pero manteniendo sus propios sistemas de archivos.

- Portabilidad: Las imágenes se pueden compartir y transportar, asegurando que la aplicación se ejecute de la misma manera en diferentes infraestructuras

4. QUE ES DOCKER HUB Y COMO SE UTILIZA PARA DESCARGAR IMAGENES

Docker Hub es un repositorio en la nube que sirve como un gran mercado de imágenes de contenedores, permitiendo a los desarrolladores almacenar, distribuir y encontrar imágenes preconstruidas para aplicaciones y sistemas. Para descargar una imagen de Docker Hub, se utiliza el comando docker pull <nombre_imagen> en la terminal, el cual descarga la imagen especificada y la prepara para su uso local.

¿Qué es Docker Hub?

Un servicio de registro: Es un registro centralizado y basado en la nube que aloja imágenes de contenedores, similar a un almacén de software.

Una biblioteca de imágenes: Ofrece una vasta colección de imágenes de Docker, tanto oficiales como creadas por la comunidad y por terceros, que pueden ser sistemas operativos, marcos de trabajo, o aplicaciones.

Para compartir y distribuir: Permite a los equipos y desarrolladores compartir sus propias imágenes personalizadas de forma pública o privada.

Simplifica el desarrollo: Proporciona imágenes prediseñadas que aceleran el proceso de configuración y desarrollo al no requerir que se construyan desde cero.

¿Cómo se utiliza para descargar imágenes?

Para descargar una imagen de Docker Hub, necesitas tener Docker instalado en tu sistema y utilizar la línea de comandos.

Abre tu terminal: o la línea de comandos de tu sistema operativo.

Utiliza el comando docker pull: seguido del nombre de la imagen que deseas descargar.

Por ejemplo, para descargar la imagen oficial de Ubuntu, el comando sería:

docker pull ubuntu

Docker Hub descarga la imagen: El cliente Docker contactará a Docker Hub, encontrará la imagen solicitada y la descargará a tu máquina local, haciendo que esté disponible para crear contenedores a partir de ella.

Para buscar imágenes: Puedes usar el comando docker search <término> para encontrar imágenes relevantes en Docker Hub antes de descargarlas.

Acceso con autenticación: Si necesitas descargar imágenes desde un repositorio privado, primero debes autenticarte con tu cuenta de Docker Hub usando docker login

5. QUE SON LOS VOLUMENES EN DOCKER

Los volúmenes de Docker son un mecanismo para almacenar datos persistentes de manera externa a los contenedores, permitiendo que estos datos sobrevivan al ciclo de vida del contenedor. Son administrados por Docker y se almacenan en el sistema host, facilitando la portabilidad, el intercambio entre contenedores, las copias de seguridad y la migración de datos. Esto los diferencia de los montajes vinculados, que están directamente ligados a un directorio específico del host.

❖ Funciones principales de los volúmenes

- **Persistencia de datos:**

Los datos guardados en un volumen persisten incluso si el contenedor se detiene, se elimina o se actualiza, evitando la pérdida de información importante.

- **Compartir datos entre contenedores:**

Varios contenedores pueden montar el mismo volumen, lo que permite compartir datos o dependencias de manera eficiente.

- **Gestión y migración:**

Docker administra los volúmenes en un directorio dedicado del host, lo que facilita la realización de copias de seguridad y la migración de datos entre diferentes hosts.

- **Portabilidad:**

Los volúmenes son más portátiles que los montajes vinculados, ya que no están atados a una ruta específica en el sistema host.

❖ Tipos de volúmenes

- **Volúmenes con nombre (Named Volumes):**

Son volúmenes gestionados por Docker, creados mediante un nombre y almacenados en un directorio en el host. Son la opción preferida para la mayoría de las aplicaciones.

- **Montajes de volumen en el host (Bind Mounts):**

Permiten montar un archivo o directorio del sistema host directamente dentro de un contenedor.

- **Volúmenes anónimos (Anonymous Volumes):**

Son volúmenes sin nombre que Docker crea y adjunta a un contenedor específico, pero no son reutilizables.

❖ ¿Por qué usar volúmenes?

- **Desacoplar los datos del contenedor:** Esto permite una gestión de datos más flexible y robusta.
- **Mejorar el rendimiento:** Al permitir el acceso directo a los datos desde el almacenamiento del host, se reduce la sobrecarga de trabajar con archivos dentro de los contenedores.
- **Facilitar copias de seguridad y restauración:** La capacidad de manejar datos por separado simplifica los procesos de respaldo y recuperación.

6. QUE ES UNA RED EN DOCKER

Una red en Docker es una infraestructura de software que permite a los contenedores comunicarse entre sí, con el host y con redes externas, proporcionando aislamiento y seguridad. Utilizando un puente virtual o un conjunto de reglas, las redes Docker permiten que los contenedores compartan recursos y datos de manera eficiente, ya sea a través de una red interna de contenedores, con el sistema anfitrión o con el Internet.

¿Cómo funciona?

1. **Puente virtual:**

Las redes Docker utilizan un puente virtual en el host para conectar los contenedores.

2. **Aislamiento:**

Los contenedores en una red específica están aislados de aquellos en otras redes, pero pueden comunicarse fácilmente entre sí dentro de su propia red.

3. **Descubrimiento y comunicación:**

Los contenedores pueden descubrirse y comunicarse usando sus nombres en lugar de direcciones IP, simplificando la conectividad.

❖ **Tipos de Redes Docker**

Docker ofrece varios tipos de controladores de red para diferentes casos de uso:

- **Bridge:**

El controlador predeterminado, que crea un puente virtual en el host, permitiendo que los contenedores se comuniquen entre sí pero los aisla de otros.

- **Host:**

Este tipo de red elimina el aislamiento entre un contenedor y el host, ya que el contenedor comparte la red directamente con el servidor anfitrión.

- **MacVLAN:**

Permite asignar una dirección MAC a un contenedor, tratándolo como un dispositivo físico en la red, lo que le permite interactuar con la LAN.

- **Redes de Swarm:**

En un clúster de Docker (swarm), estas redes facilitan la comunicación entre servicios que se ejecutan en diferentes nodos, asegurando una comunicación segura y fluida.

7. QUE ES DOCKER COMPOSE Y QUE VENTAJAS TIENE FRENTE A EJECUTAR UN CONTENEDOR CON DOCKER RUN

Docker Compose es una herramienta para definir, ejecutar y gestionar aplicaciones Docker multicontenedor desde un único archivo YAML, mientras que docker run solo inicia un contenedor a la vez. Las ventajas principales de Compose son la facilidad para definir y gestionar múltiples servicios interconectados, la ejecución con un solo comando, la creación automática de redes y volúmenes, el entorno consistente para desarrollo y pruebas, y la simplificación de la colaboración del equipo al compartir la configuración.

❖ **¿Qué es Docker Compose?**

- Docker Compose es una herramienta que te permite definir los servicios, redes y volúmenes de tu aplicación en un archivo de configuración docker-compose.yml con formato YAML.
- Este archivo actúa como "código" para describir la infraestructura de tu aplicación, lo que simplifica su implementación y gestión.

❖ **Ventajas de Docker Compose sobre docker run**

1. **Gestión de múltiples contenedores:**

- docker run solo inicia un contenedor individualmente.
- Docker Compose gestiona un conjunto de servicios (contenedores) interconectados, como una base de datos, un frontend y un backend, todos definidos en un solo archivo.

2. **Comandos simplificados:**

- Con docker run, debes escribir comandos extensos y específicos para cada contenedor que quieras iniciar, vincular y configurar.
- Docker Compose permite iniciar, detener y reconstruir toda tu aplicación con un solo comando, por ejemplo, docker-compose up.

3. **Redes y volúmenes automáticos:**

- Docker Compose crea automáticamente una red para que los contenedores de tu proyecto puedan comunicarse entre sí sin configuración adicional.
- También administra los volúmenes de almacenamiento, facilitando la persistencia de datos entre contenedores y reinicios.

4. **Consistencia y repetibilidad del entorno:**

- La configuración en el archivo YAML asegura que todos los servicios se ejecuten siempre con la misma configuración.
- Esto elimina los problemas de "¿por qué funcionaba en mi máquina?" y facilita la creación de entornos de desarrollo y prueba consistentes.

5. **Mejora de la colaboración en equipo:**

   Al compartir el archivo docker-compose.yml, los desarrolladores pueden desplegar una réplica exacta del entorno de la aplicación en sus máquinas, lo que mejora la colaboración y reduce los ciclos de prueba.

En resumen, mientras que docker run es ideal para tareas sencillas con un solo contenedor, Docker Compose es esencial para construir y gestionar aplicaciones multicontenedor complejas de manera eficiente y consistente.

BASES DE DATOS

1. ENTIDADES

En el contexto de las bases de datos, una entidad es un objeto, concepto o cosa distinta y única en el mundo real o abstracto, sobre la cual se desea almacenar información. Las entidades suelen convertirse en las tablas de la base de datos y se describen mediante sus atributos o propiedades. Ejemplos comunes incluyen "Cliente", "Producto", "Libro", "Vehículo" o "Cuenta Bancaria", tanto para cosas tangibles como intangibles.

Características clave de una entidad:

- Singularidad: Cada entidad es única y se puede distinguir de otras.
- Representación: En un diseño de base de datos, una entidad se representa como una tabla.
- Atributos: Cada entidad tiene propiedades o características que la describen, como "Nombre" o "Fecha de Nacimiento" para un cliente.
- Relaciones: Las entidades se vinculan entre sí a través de relaciones para formar un modelo de datos completo

Tipos de entidades

- Entidades fuertes: tienen una clave primaria propia que identifica de forma única cada fila y no dependen de otra entidad para existir.
- Entidades débiles: dependen de otra entidad para su identificación; su clave parcial se completa con la clave de la entidad fuerte relacionada.
- Tangibles e intangibles: pueden ser objetos físicos (Persona, Vehículo) o conceptos/lógicos (Cuenta, Inscripción).

2. ATRIBUTOS

Los atributos son características o propiedades que describen a una entidad en una base de datos relacional, como "nombre" o "color". Se clasifican en tipos según su composición (compuesto o simple/atómico), cantidad de valores (multivaluado o monovaluado), origen de sus datos (almacenado o derivado), y complejidad (simple, compuesto, multivaluado, o complejo).

Tipos de atributos según la composición

- Atributo simple o atómico: No puede dividirse en componentes más pequeños, como "DNI".
- Atributo compuesto: Se puede dividir en otros atributos más simples, por ejemplo, "dirección" puede dividirse en "calle", "ciudad" y "país".

Tipos de atributos según la cantidad de valores

- Atributo monovaluado: Tiene un único valor por entidad, como el "número de identificación".
- Atributo multivaluado: Puede tener varios valores para una misma entidad, como el número de email de un empleado, que puede tener varias cuentas de correo.

Tipos de atributos según el origen del valor

- Atributo almacenado: Es un atributo cuyo valor se guarda directamente en la base de datos.
- Atributo derivado: Su valor se calcula a partir de otros atributos, como la "edad" que se calcula a partir de la "fecha de nacimiento".

Tipos de atributos según la complejidad

- Atributo complejo: Es una combinación de atributos compuestos y multivaluados, que pueden tener varios niveles de sub-atributos

Atributos clave

- Clave primaria: atributo o conjunto de atributos que identifican unívocamente cada instancia de la entidad.
- Clave foránea: atributo que referencia la clave primaria de otra entidad para materializar relaciones

3. RELACIONES

Las relaciones en bases de datos son asociaciones lógicas entre tablas que organizan los datos de manera eficiente y evitan la duplicidad, permitiendo conectar información relacionada a través de claves principales y externas. Sirven para establecer vínculos entre diferentes conjuntos de datos, como un cliente y sus pedidos, garantizando la integridad y consistencia de la información.

Componentes Clave

Tablas (o "Relaciones"): Son colecciones de datos organizadas en filas (registros) y columnas (atributos).

Claves:

Clave Primaria (Primary Key): Un identificador único para cada registro dentro de una tabla.

Clave Externa (Foreign Key): Un campo en una tabla que se refiere a la clave primaria de otra tabla, creando así la relación entre ellas.

Tipos de Relaciones

Las relaciones se clasifican por la cantidad de registros que pueden conectarse:

- Uno a Uno: Un registro en una tabla se relaciona con un único registro en otra tabla.
- Uno a Muchos (o Uno a Varios): Un registro en la primera tabla puede relacionarse con múltiples registros en la segunda tabla, pero cada registro en la segunda tabla se relaciona con un único registro en la primera.
- Muchos a Muchos: Varios registros en la primera tabla pueden relacionarse con varios registros en la segunda tabla.

Propósito y Beneficios

- Integridad Referencial: Aseguran que los datos no se modifiquen o pierdan durante el proceso, manteniendo la consistencia entre las tablas.
- Eliminación de Duplicidad: La estructura relacional ayuda a evitar la repetición de información, haciendo que las bases de datos sean más eficientes.
- Consulta de Datos: Permiten que los usuarios realicen consultas para combinar y analizar datos de diferentes tablas, extrayendo información valiosa para la toma de decisiones.

Grado de la relación

- Binaria: involucra dos entidades (Cliente–Pedido), es la más común en diseños relacionales.
- N-aria: involucra tres o más entidades (por ejemplo, Cliente–Cuenta–Sucursal), útil cuando la asociación depende de múltiples participantes

Participación y opcionalidad

- Total vs. parcial: participación total exige que toda instancia de la entidad participe en la relación, mientras que parcial significa que solo algunas instancias lo hacen.
- La opcionalidad influye en restricciones y claves foráneas al transformar el modelo a tablas.

4. CARDINALIDAD

En bases de datos, la cardinalidad describe el número de instancias de una entidad que pueden estar relacionadas con instancias de otra entidad, y existen dos tipos principales de cardinalidad: la que se refiere a las relaciones entre tablas (uno a uno, uno a muchos, muchos a muchos) y la que describe la unicidad de los valores en una columna (alta, media o baja cardinalidad).

Cardinalidad en relaciones de tablas

Este tipo de cardinalidad define cuántos elementos de una tabla pueden asociarse con cuántos elementos de otra tabla. Los tipos más comunes son:

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

Cada registro en una tabla se relaciona con un solo registro en la otra tabla. Por ejemplo, un empleado puede tener una única cuenta de usuario.

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

Un registro en una tabla puede estar relacionado con muchos registros en la otra tabla, pero cada registro en la segunda tabla se relaciona con solo un registro en la primera. Por ejemplo, un autor puede escribir muchos libros, pero un libro es escrito por un único autor.

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

Varios registros en una tabla pueden relacionarse con varios registros en la otra tabla. Por ejemplo, un estudiante puede inscribirse en varios cursos y un curso puede tener varios estudiantes inscritos.

❖ **Cardinalidad de columnas (unicidad de datos)**

Esta definición se refiere a la cantidad de valores únicos que existen en una columna de una tabla en relación con el número total de filas. Se utiliza para que el optimizador de la base de datos elija las consultas más eficientes.

- **Alta Cardinalidad:**

Una columna tiene muchos valores distintos, con mínima repetición. Ejemplos: identificadores de clave principal, direcciones de correo electrónico.

- **Baja Cardinalidad:**

Una columna tiene una gran cantidad de valores que se repiten. Ejemplos: una columna de género que puede tener solo "masculino", "femenino" o "otro".

- **Cardinalidad Media:**

Un equilibrio entre valores únicos y repetidos. Por ejemplo, una columna de país en una base de datos global.

Participación y opcionalidad

- Total vs. parcial: participación total exige que todas las instancias participen en la relación; la parcial permite que solo algunas lo hagan.
- La opcionalidad guía la necesidad de claves foráneas no nulas o restricciones únicas al transformar ER a relacional.

5. FORMAS NORMALES DE UNA BD

Las formas normales (FN) son reglas que ayudan a organizar las columnas y tablas de una base de datos relacional para reducir la redundancia y mejorar la integridad de los datos. Las formas más comunes son la Primera Forma Normal (1NF), que requiere valores atómicos; la Segunda Forma Normal (2NF), que elimina dependencias parciales; la Tercera Forma Normal (3NF), que elimina dependencias transitivas; y la Forma Normal de Boyce-Codd (BCNF). Las formas superiores (4FN y 5FN) abordan problemas más complejos relacionados con las dependencias multivaluadas y de unión.

¿Por qué usar formas normales?

- Reduce la redundancia: Evita que los mismos datos se repitan en múltiples lugares.
- Mejora la integridad de los datos: Al eliminar la redundancia, se disminuye la posibilidad de inconsistencias.
- Facilita la gestión de la base de datos: Una base de datos bien normalizada es más fácil de mantener y modificar.

Las Formas Normales Principales

Primera Forma Normal (1NF):

- Cada atributo (columna) debe contener un solo valor atómico (indivisible).
- No debe haber grupos repetidos de columnas en la misma fila.

Segunda Forma Normal (2NF):

- La tabla debe estar en 1NF.
- Todos los atributos no clave deben depender completamente de toda la clave primaria (no solo de una parte de ella).

Tercera Forma Normal (3NF):

- La tabla debe estar en 2NF.
- No debe haber dependencias transitivas, lo que significa que un atributo no clave no debe depender de otro atributo no clave.
- Forma Normal de Boyce-Codd (BCNF):
- Es una versión más estricta de 3NF.
- Para cada dependencia funcional A → B, A debe ser una superclave.

Formas Normales Superiores

Cuarta Forma Normal (4NF):

- Elimina las dependencias multivaluadas o dependencias de múltiples valores.

Quinta Forma Normal (5NF) (Forma Normal de Proyección-Unión o PJ/NF):

- Aborda las dependencias de unión, asegurando que los datos no puedan descomponerse y volver a unirse de manera redundante sin perder información

Cuando desnormalizar

- Desnormalizar es una decisión de rendimiento con costo en integridad; solo debería hacerse con métricas claras y usualmente acompañada de vistas materializadas o índices adecuados.

GLOSARIO:

- Contenedor: proceso aislado que comparte el kernel del host y ejecuta una app con sus dependencias, iniciando en segundos y con bajo consumo de recursos.
- Imagen: plantilla inmutable por capas, versionable, normalmente construida desde un Dockerfile; un contenedor es una imagen "+" capa escribible en runtime.
- Registro: servicio para almacenar y distribuir imágenes; Docker Hub es el registro público por defecto usado por docker pull/push