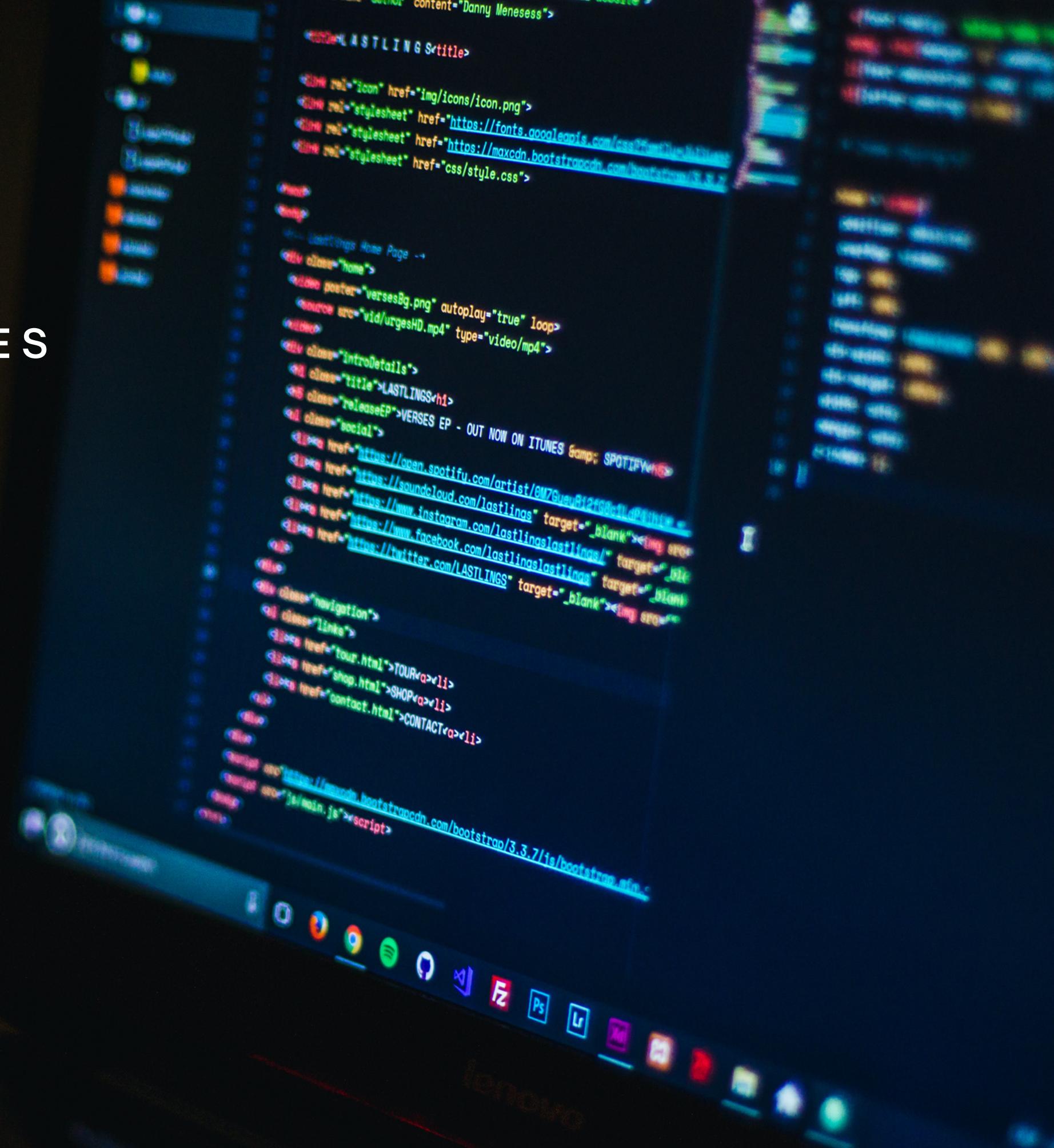


EXPLORANDO EL MUNDO DE LAS BASES DE DATOS: SQL PARA TODOS LOS GESTORES

Parte N°1

SERGIO GABRIEL OBERENKO





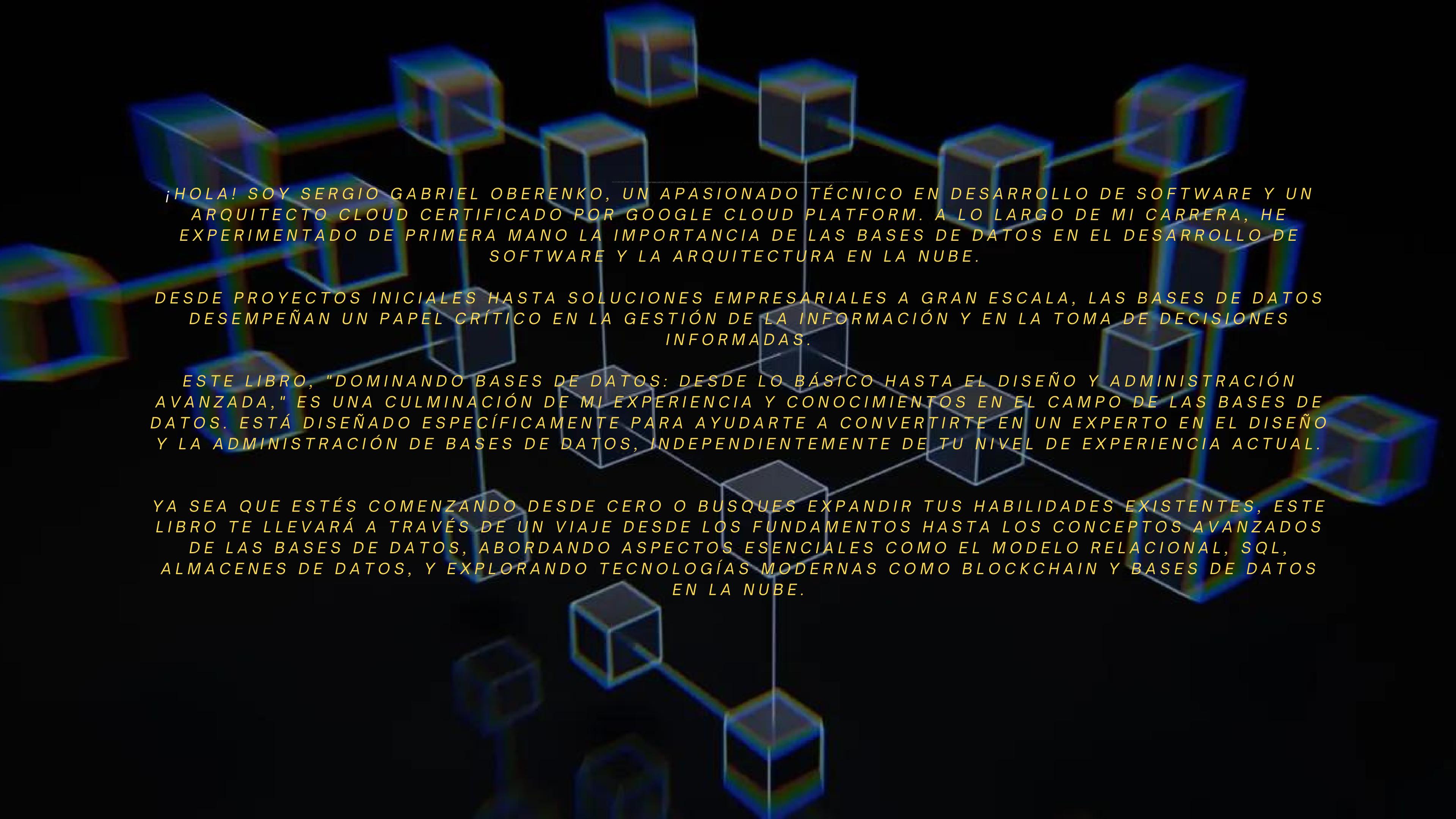
EN UN MUNDO CADA VEZ MÁS IMPULSADO POR LOS DATOS, LA GESTIÓN Y EL DISEÑO EFICIENTES DE BASES DE DATOS SE HAN CONVERTIDO EN UNA HABILIDAD FUNDAMENTAL Y ESENCIAL PARA UNA AMPLIA VARIEDAD DE INDUSTRIAS Y APLICACIONES. DESDE EL SEGUIMIENTO DE REGISTROS EMPRESARIALES HASTA EL ANÁLISIS DE INFORMACIÓN CRÍTICA, LAS BASES DE DATOS SON LA COLUMNA VERTEBRAL QUE SUSTENTA UNA GRAN CANTIDAD DE OPERACIONES Y TOMA DE DECISIONES EN LA ACTUALIDAD.

ESTE LIBRO, "DOMINANDO BASES DE DATOS: DESDE LO BÁSICO HASTA EL DISEÑO Y ADMINISTRACIÓN AVANZADA," ESTÁ DISEÑADO PARA LLEVARTE DESDE LOS FUNDAMENTOS DE LAS BASES DE DATOS HASTA CONVERTIRTE EN UN EXPERTO EN SU DISEÑO Y ADMINISTRACIÓN. SEA QUE ESTÉS COMENZANDO DESDE CERO O BUSCANDO AMPLIAR TUS CONOCIMIENTOS EXISTENTES, ESTA OBRA TIENE COMO OBJETIVO PROPORCIONARTE UNA COMPRENSIÓN SÓLIDA DE LOS CONCEPTOS Y LAS PRÁCTICAS CLAVE RELACIONADAS CON LAS BASES DE DATOS.



EN LAS PÁGINAS QUE SIGUEN, EXPLORAREMOS LOS SIGUIENTES ASPECTOS:

- FUNDAMENTOS DE LAS BASES DE DATOS: COMENZAREMOS POR LOS FUNDAMENTOS ESENCIALES, INCLUYENDO QUÉ SON LAS BASES DE DATOS, POR QUÉ SON IMPORTANTES Y LOS MODELOS DE DATOS MÁS COMUNES, CON UN ENFOQUE PARTICULAR EN EL MODELO RELACIONAL.
- SQL (STRUCTURED QUERY LANGUAGE): TE SUMERGIRÁS EN EL LENGUAJE DE CONSULTA ESTRUCTURADO (SQL) Y APRENDERÁS CÓMO REALIZAR CONSULTAS, INSERTAR, ACTUALIZAR Y ELIMINAR DATOS EN UNA BASE DE DATOS.
- DATA WAREHOUSING: ABORDAREMOS EL EMOCIONANTE MUNDO DE LOS ALMACENES DE DATOS, DONDE APRENDERÁS CÓMO DISEÑAR Y UTILIZAR ALMACENES DE DATOS PARA EL ANÁLISIS EMPRESARIAL.
- BASES DE DATOS AVANZADAS: EXPLORAREMOS TECNOLOGÍAS EMERGENTES Y AVANZADAS, INCLUYENDO BLOCKCHAIN Y BASES DE DATOS DISTRIBUIDAS, ASÍ COMO BASES DE DATOS EN LA NUBE, DONDE ENTENDERÁS CÓMO LAS BASES DE DATOS ESTÁN EVOLUCIONANDO PARA ENFRENTAR LOS DESAFÍOS MODERNOS.



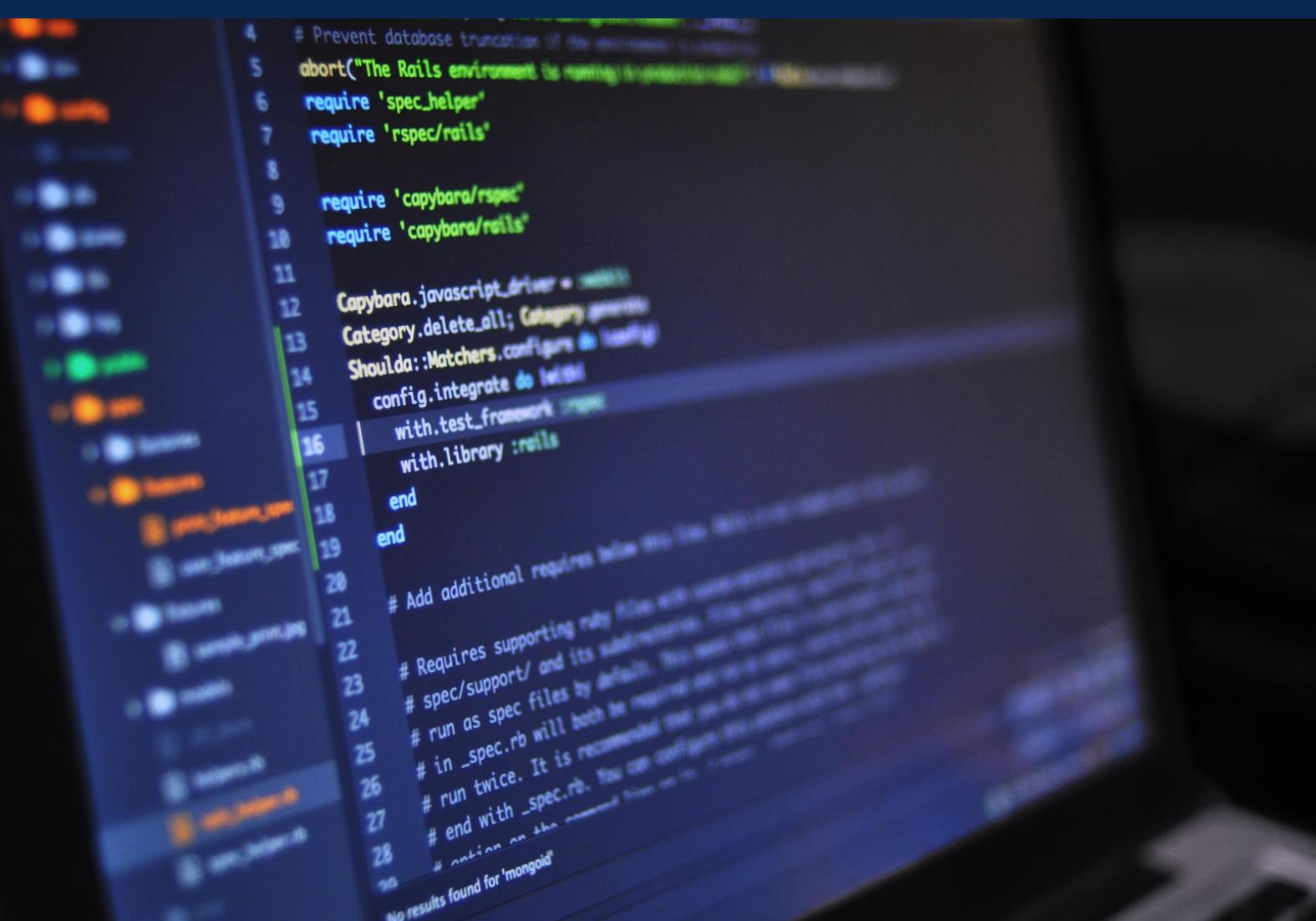
HOLA! SOY SERGIO GABRIEL OBERENKO, UN APASIONADO TÉCNICO EN DESARROLLO DE SOFTWARE Y UN ARQUITECTO CLOUD CERTIFICADO POR GOOGLE CLOUD PLATFORM. A LO LARGO DE MI CARRERA, HE EXPERIMENTADO DE PRIMERA MANO LA IMPORTANCIA DE LAS BASES DE DATOS EN EL DESARROLLO DE SOFTWARE Y LA ARQUITECTURA EN LA NUBE.

DESDE PROYECTOS INICIALES HASTA SOLUCIONES EMPRESARIALES A GRAN ESCALA, LAS BASES DE DATOS DESEMPEÑAN UN PAPEL CRÍTICO EN LA GESTIÓN DE LA INFORMACIÓN Y EN LA TOMA DE DECISIONES INFORMADAS.

ESTE LIBRO, "DOMINANDO BASES DE DATOS: DESDE LO BÁSICO HASTA EL DISEÑO Y ADMINISTRACIÓN AVANZADA," ES UNA CULMINACIÓN DE MI EXPERIENCIA Y CONOCIMIENTOS EN EL CAMPO DE LAS BASES DE DATOS. ESTÁ DISEÑADO ESPECÍFICAMENTE PARA AYUDARTE A CONVERTIRTE EN UN EXPERTO EN EL DISEÑO Y LA ADMINISTRACIÓN DE BASES DE DATOS, INDEPENDIENTEMENTE DE TU NIVEL DE EXPERIENCIA ACTUAL.

YA SEA QUE ESTÉS COMENZANDO DESDE CERO O BUSQUES EXPANDIR TUS HABILIDADES EXISTENTES, ESTE LIBRO TE LLEVARÁ A TRAVÉS DE UN VIAJE DESDE LOS FUNDAMENTOS HASTA LOS CONCEPTOS AVANZADOS DE LAS BASES DE DATOS, ABORDANDO ASPECTOS ESENCIALES COMO EL MODELO RELACIONAL, SQL, ALMACENES DE DATOS, Y EXPLORANDO TECNOLOGÍAS MODERNAS COMO BLOCKCHAIN Y BASES DE DATOS EN LA NUBE.

INDICE DE CONTENIDOS



```
4 # Prevent database truncation if the environment is test or higher.
5 abort("The Rails environment is running in production mode")
6 require 'spec_helper'
7 require 'rspec/rails'
8
9 require 'capybara/rspec'
10 require 'capybara/rails'
11
12 Capybara.javascript_driver = :webkit
13 Category.delete_all; Category.create!(name: "Electronics")
14 Shoulda::Matchers.configure do |config|
15   config.integrate do |with|
16     with.test_framework :rspec
17     with.library :rails
18   end
19 end
20
21 # Add additional requires below this line to access routes in feature tests
22 # Requires supporting ruby files with custom matchers and helpers
23 # in spec/support/ and its subdirectories. This directory also
24 # contains supporting files for this generator.
25 # run as spec files by default. You can change this using
26 # --withFEATURE in your command line options or in
27 # your Rakefile. By default, it will run
28 # twice. It is recommended you do not change this value.
29 # end with _spec.rb. You can configure the
30 # generator to ignore files ending in _spec.rb
31 # via the ignore_generator option in your Rakefile
32
33 # mongoid
34
35 # No results found for 'mongoid'
```

1. FUNDAMENTOS DE BASES DE DATOS
2. DATA WAREHOUSING
3. BASES DE DATOS AVANZADAS
4. APLICACIONES PRÁCTICAS
5. CONCLUSIÓN

PROYECTO FIN DE GRADO

FUNDAMENTOS DE BASES DE DATOS

DEFINICIÓN DE BASES DE DATOS:

Una base de datos es una colección organizada de información o datos, diseñada para facilitar el almacenamiento, la recuperación, la gestión y la manipulación eficiente de esos datos. Las bases de datos son esenciales en numerosas aplicaciones y campos, desde la gestión empresarial hasta la investigación científica y la toma de decisiones.

Beneficios de utilizar bases de datos:

- Organización: Las bases de datos permiten organizar los datos de manera estructurada, lo que facilita la búsqueda y recuperación de información específica.
- Integridad de datos: Ayudan a mantener la integridad de los datos, evitando la duplicación y asegurando que la información sea precisa y consistente.
- Seguridad: Las bases de datos pueden implementar medidas de seguridad para proteger la información sensible.
- Acceso concurrente: Permiten el acceso simultáneo de múltiples usuarios, lo que es esencial en entornos empresariales.
- Escalabilidad: Pueden adaptarse a medida que la cantidad de datos crece.
- Eficiencia: Facilitan la gestión de grandes volúmenes de información y la realización de consultas complejas.

MODELOS DE DATOS Y SU EVOLUCIÓN:

- LOS MODELOS DE DATOS DEFINEN CÓMO SE ESTRUCTURAN Y REPRESENTAN LOS DATOS EN UNA BASE DE DATOS.
- LOS MODELOS INICIALES INCLUYEN EL MODELO JERÁRQUICO Y EL MODELO DE RED, QUE EVOLUCIONARON HACIA EL MODELO RELACIONAL.
- EL MODELO RELACIONAL, BASADO EN TABLAS, SE CONVIRTIÓ EN EL ESTÁNDAR DE LA INDUSTRIA DEBIDO A SU SIMPLICIDAD Y EFICIENCIA.
- MODELOS MÁS RECIENTES, COMO LOS MODELOS DE OBJETOS, NOSQL Y NEWSQL, SE HAN DESARROLLADO PARA ABORDAR NECESIDADES ESPECÍFICAS Y DESAFÍOS DE DATOS MODERNOS.

CONCEPTOS BÁSICOS DEL MODELO RELACIONAL:

CONCEPTOS BÁSICOS DEL MODELO RELACIONAL:

Conceptos básicos del modelo relacional:

- El modelo relacional organiza los datos en tablas (relaciones) que consisten en filas (registros) y columnas (atributos).
- Cada tabla representa una entidad o concepto, y cada fila es una instancia de esa entidad.
- Los atributos (columnas) contienen información sobre las características de la entidad.
- Las tablas se relacionan a través de claves primarias y foráneas.

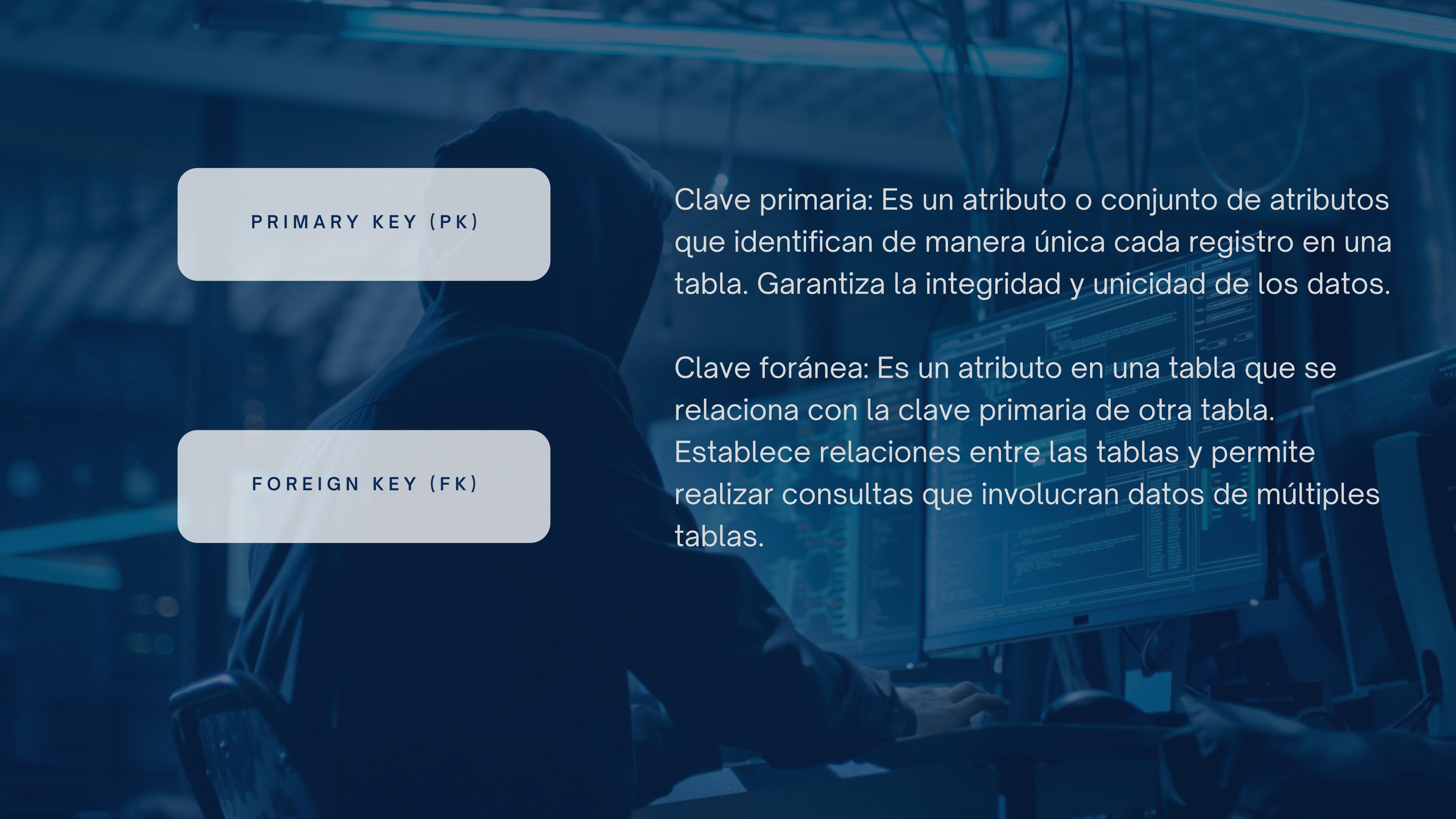
Tablas, registros y atributos:

- Tablas: Son la estructura fundamental del modelo relacional, y representan una entidad o concepto específico (por ejemplo, una tabla "Clientes").
- Registros: Son las filas individuales dentro de una tabla y representan instancias de la entidad (por ejemplo, un registro para un cliente específico).
- Atributos: Son las columnas de una tabla y contienen información sobre las características de la entidad (por ejemplo, Nombre, Dirección para una tabla de Clientes).

TABLAS

REGISTROS

ATRIBUTOS

The background of the slide shows a person from behind, wearing a hooded sweatshirt, sitting at a desk and working on a computer. There are multiple monitors in front of them, each displaying lines of code or data. The scene is dimly lit, with the screens being the primary light source.

PRIMARY KEY (PK)

FOREIGN KEY (FK)

Clave primaria: Es un atributo o conjunto de atributos que identifican de manera única cada registro en una tabla. Garantiza la integridad y unicidad de los datos.

Clave foránea: Es un atributo en una tabla que se relaciona con la clave primaria de otra tabla. Establece relaciones entre las tablas y permite realizar consultas que involucran datos de múltiples tablas.

SQL (STRUCTURED QUERY LANGUAGE)

INTRODUCCIÓN A SQL:

SQL (STRUCTURED QUERY LANGUAGE) ES UN LENGUAJE DE PROGRAMACIÓN DISEÑADO PARA GESTIONAR Y CONSULTAR BASES DE DATOS RELACIONALES.

CONSULTAS BÁSICAS Y AVANZADAS CON SQL:

SQL PERMITE REALIZAR CONSULTAS BÁSICAS COMO SELECT PARA RECUPERAR DATOS DE UNA TABLA.

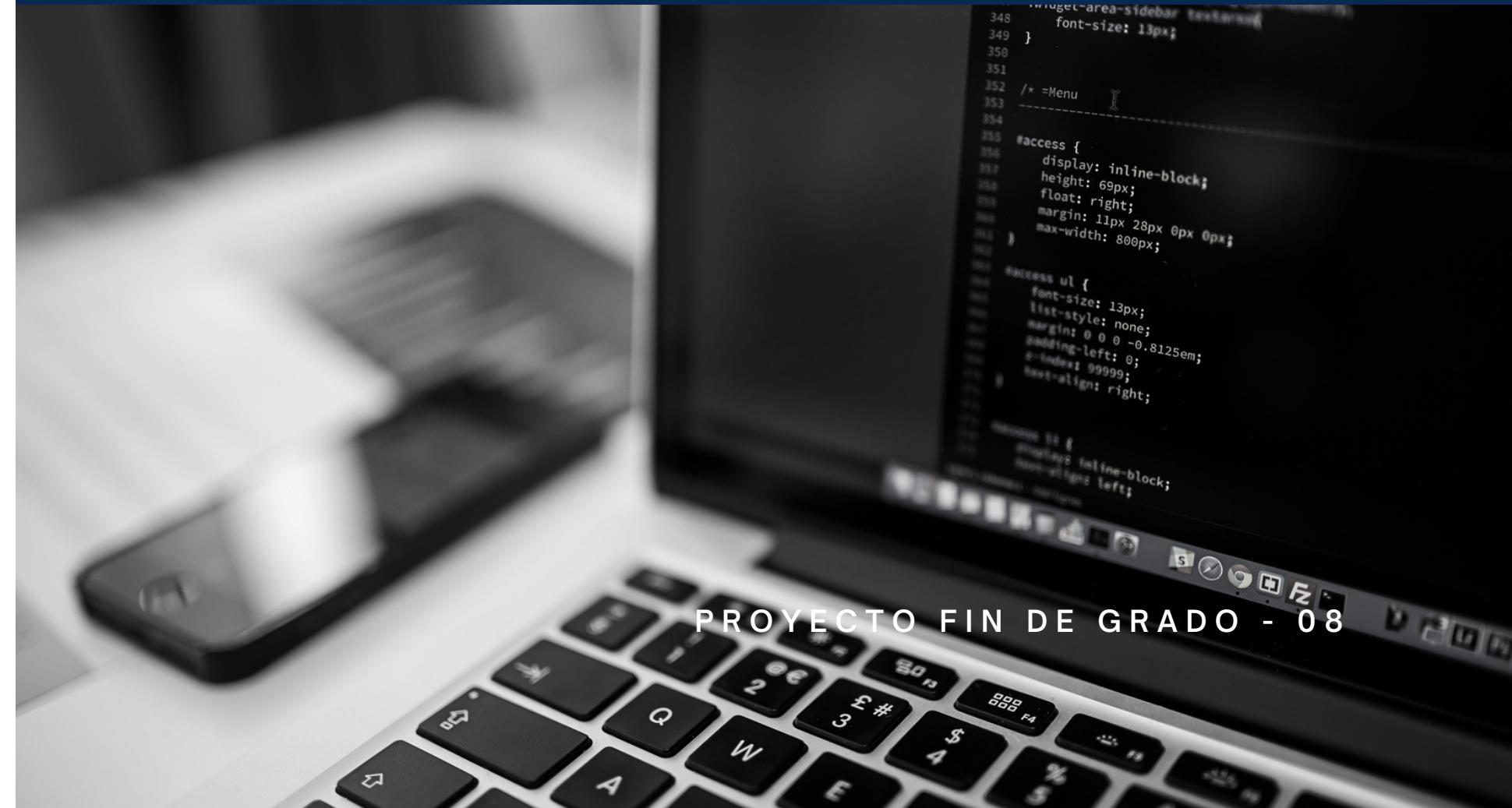
CONSULTAS AVANZADAS INCLUYEN OPERACIONES JOIN PARA COMBINAR DATOS DE MÚLTIPLES TABLAS Y FUNCIONES AGREGADAS (SUM, AVG, COUNT) PARA REALIZAR CÁLCULOS EN LOS DATOS.

FUNCIONES Y PROCEDIMIENTOS ALMACENADOS:

SQL PERMITE DEFINIR Y UTILIZAR FUNCIONES Y PROCEDIMIENTOS ALMACENADOS PARA ENCAPSULAR LÓGICA DE NEGOCIOS Y REUTILIZAR CÓDIGO.

LAS FUNCIONES PUEDEN DEVOLVER VALORES Y LOS PROCEDIMIENTOS REALIZAN ACCIONES ESPECÍFICAS EN LA BASE DE DATOS.

BD
SGBD/RDBMS
SQL



C R E A U N A T A B L A

CREATE TABLE:

LA INSTRUCCIÓN CREATE TABLE SE UTILIZA PARA CREAR UNA NUEVA TABLA EN UNA BASE DE DATOS.

```
CREATE TABLE NOMBREDELATABLA (
    NOM_COLUMNNA1 TIPO_DE_DATO,
    NOM_COLUMNNA2 TIPO_DE_DATO,
    NOM_COLUMNNA3 TIPO_DE_DATO,
);
```

```
CREATE TABLE Empleados (
    EmpleadoID INTEGER,
    Nombre TEXT,
    FechaContratacion DATE,
    Salario REAL
);
```

SELECT:
LA INSTRUCCIÓN
SELECT SE UTILIZA
PARA RECUPERAR
DATOS DE UNA TABLA

**SELECT COLUMNA1,COLUMNA2
FROM NOMBRE_DE_TABLA;**

SELECTOR UNIVERSAL ("*"):

EL SELECTOR UNIVERSAL, QUE SE
REPRESENTA COMO "", SE UTILIZA
PARA SELECCIONAR TODAS LAS
COLUMNAS DE UNA TABLA SIN
ESPECIFICAR CADA COLUMNA
INDIVIDUALMENTE

**SELECT Nombre, Apellido
FROM Empleados;**

Nombre	Apellido
Luis	Martínez
Ana	Rodríguez
Carlos	López
María	García
Pedro	Ramírez
Laura	Sánchez
Sofía	Hernández
Diego	Fernández
Elena	Díaz
Miquel	González
Maria	NULL

TIPOS DE DATOS EN SQL

LOS TIPOS DE DATOS DISPONIBLES EN SQL PUEDEN VARIAR SEGÚN EL SISTEMA DE GESTIÓN DE BASES DE DATOS (DBMS) QUE ESTÉS UTILIZANDO, YA QUE CADA DBMS PUEDE TENER SUS PROPIOS TIPOS DE DATOS ESPECÍFICOS. SIN EMBARGO, AQUÍ TIENES UNA LISTA DE ALGUNOS TIPOS DE DATOS COMUNES QUE SE ENCUENTRAN EN SQL ESTÁNDAR, QUE SON COMPATIBLES CON MUCHOS SISTEMAS DE BASES DE DATOS:

ES IMPORTANTE TENER EN CUENTA QUE ALGUNOS DE ESTOS TIPOS DE DATOS PUEDEN NO ESTAR DISPONIBLES EN TODOS LOS SISTEMAS DE BASES DE DATOS, Y ALGUNOS SISTEMAS PUEDEN TENER TIPOS DE DATOS ADICIONALES ESPECÍFICOS DE ESE SISTEMA. POR LO TANTO, ES IMPORTANTE CONSULTAR LA DOCUMENTACIÓN DEL SISTEMA DE BASES DE DATOS QUE ESTÁS UTILIZANDO PARA OBTENER INFORMACIÓN ESPECÍFICA SOBRE LOS TIPOS DE DATOS ADMITIDOS.

INTEGER O INT: PARA NÚMEROS ENTEROS.

SMALLINT: PARA NÚMEROS ENTEROS PEQUEÑOS.

BIGINT: PARA NÚMEROS ENTEROS GRANDES.

REAL O FLOAT: PARA NÚMEROS DE PUNTO FLOTANTE.

DOUBLE PRECISION: PARA NÚMEROS DE DOBLE PRECISIÓN (PUNTO FLOTANTE DE DOBLE PRECISIÓN).

DECIMAL O NUMERIC: PARA NÚMEROS DECIMALES CON PRECISIÓN FIJA.

CHAR(N): PARA CADENAS DE CARACTERES DE LONGITUD FIJA, DONDE "N" REPRESENTA LA LONGITUD MÁXIMA.

VARCHAR(N): PARA CADENAS DE CARACTERES DE LONGITUD VARIABLE, DONDE "N" REPRESENTA LA LONGITUD MÁXIMA.

TEXT: PARA CADENAS DE TEXTO LARGAS O DE LONGITUD VARIABLE.

DATE: PARA ALMACENAR FECHAS EN FORMATO "YYYY-MM-DD".

TIME: PARA ALMACENAR HORAS Y MINUTOS EN FORMATO "HH:MM:SS".

DATETIME: PARA ALMACENAR FECHAS Y HORAS COMBINADAS.

BOOLEAN: PARA VALORES VERDADEROS O FALSOS (1 O 0).

BINARY: PARA DATOS BINARIOS, COMO IMÁGENES O ARCHIVOS.

VARBINARY: PARA DATOS BINARIOS DE LONGITUD VARIABLE.

BLOB: PARA DATOS BINARIOS LARGOS.

CLOB: PARA TEXTO LARGO, COMO DOCUMENTOS O DESCRIPCIONES EXTENSAS.

ENUM: UN TIPO DE DATOS ESPECÍFICO DE ALGUNOS SISTEMAS DE BASES DE DATOS QUE PERMITE DEFINIR UN CONJUNTO FIJO DE VALORES POSIBLES.

SET: OTRO TIPO DE DATOS ESPECÍFICO DE ALGUNOS SISTEMAS DE BASES DE DATOS QUE PERMITE DEFINIR UN CONJUNTO DE VALORES, PERO A DIFERENCIA DE ENUM, SE PUEDEN SELECCIONAR VARIOS VALORES DE UN CONJUNTO EN UNA SOLA COLUMNA.

UUID: PARA IDENTIFICADORES ÚNICOS UNIVERSALES, ESPECIALMENTE EN BASES DE DATOS MODERNAS.

JSON: PARA ALMACENAR DATOS EN FORMATO JSON (JAVASCRIPT OBJECT NOTATION).

XML: PARA ALMACENAR DATOS EN FORMATO XML (EXTENSIBLE MARKUP LANGUAGE).

GEOMETRY: PARA DATOS GEOMÉTRICOS, UTILIZADOS EN SISTEMAS DE INFORMACIÓN GEOGRÁFICA (GIS).

ARRAY: PARA ALMACENAR ARRAYS O LISTAS DE VALORES.

INTERVAL: PARA REPRESENTAR UN INTERVALO DE TIEMPO.

SERIAL O AUTO_INCREMENT: PARA CREAR UNA COLUMNA AUTOINCREMENTAL QUE GENERA AUTOMÁTICAMENTE VALORES ÚNICOS.

**LISTA DE TIPOS DE DATOS COMUNES
PARA CADA UNO DE LOS SISTEMAS DE
GESTIÓN DE BASES DE DATOS:**

SQLITE3

MYSQL

SQL SERVER

POSTGRESQL

ORACLE

MONGODB

SQLITE3:

1. INTEGER: PARA NÚMEROS ENTEROS.
2. REAL: PARA NÚMEROS DECIMALES.
3. TEXT: PARA CADENAS DE TEXTO.
4. BLOB: PARA DATOS BINARIOS (COMO IMÁGENES O ARCHIVOS).
5. DATE: PARA ALMACENAR FECHAS EN FORMATO YYYY-MM-DD.
6. TIME: PARA ALMACENAR TIEMPOS EN FORMATO HH:MM:SS.
7. DATETIME: PARA ALMACENAR FECHAS Y HORAS COMBINADAS.
8. NUMERIC: ES UN TIPO DE DATO NUMÉRICO QUE SE UTILIZA PARA ALMACENAR NÚMEROS, YA SEAN ENTEROS O DECIMALES, SIN UNA PRECISIÓN ESPECÍFICA.

MYSQL:

- INT O INTEGER: PARA NÚMEROS ENTEROS.
- FLOAT: PARA NÚMEROS DE PUNTO FLOTANTE.
- VARCHAR(N): PARA CADENAS DE TEXTO DE LONGITUD VARIABLE, DONDE "N" REPRESENTA LA LONGITUD MÁXIMA.
- DATE: PARA ALMACENAR FECHAS EN FORMATO YYYY-MM-DD.
- TIME: PARA ALMACENAR TIEMPOS EN FORMATO HH:MM:SS.
- DATETIME: PARA ALMACENAR FECHAS Y HORAS COMBINADAS.
- BLOB: PARA DATOS BINARIOS.
- BOOLEAN: PARA VALORES VERDADEROS O FALSOS.

SQL SERVER:

1. INT: PARA NÚMEROS ENTEROS.
2. DECIMAL O NUMERIC: PARA NÚMEROS DECIMALES DE PRECISIÓN FIJA.
3. VARCHAR(N): PARA CADENAS DE TEXTO DE LONGITUD VARIABLE, DONDE "N" REPRESENTA LA LONGITUD MÁXIMA.
4. DATE: PARA ALMACENAR FECHAS EN FORMATO YYYY-MM-DD.
5. TIME: PARA ALMACENAR TIEMPOS EN FORMATO HH:MM:SS.
6. DATETIME: PARA ALMACENAR FECHAS Y HORAS COMBINADAS.
7. BINARY: PARA DATOS BINARIOS.
8. BIT: PARA VALORES VERDADEROS O FALSOS.

POSTGRESQL:

1. INTEGER: PARA NÚMEROS ENTEROS.
2. DECIMAL O NUMERIC: PARA NÚMEROS DECIMALES.
3. VARCHAR(N): PARA CADENAS DE TEXTO DE LONGITUD VARIABLE, DONDE "N" REPRESENTA LA LONGITUD MÁXIMA.
4. DATE: PARA ALMACENAR FECHAS EN FORMATO YYYY-MM-DD.
5. TIME: PARA ALMACENAR TIEMPOS EN FORMATO HH:MM:SS.
6. TIMESTAMP: PARA ALMACENAR FECHAS Y HORAS COMBINADAS.
7. BYTEA: PARA DATOS BINARIOS.
8. BOOLEAN: PARA VALORES VERDADEROS O FALSOS.

ORACLE:

1. **NUMBER**: PARA NÚMEROS ENTEROS O DECIMALES.
2. **VARCHAR2(N)**: PARA CADENAS DE TEXTO DE LONGITUD VARIABLE, DONDE "N" REPRESENTA LA LONGITUD MÁXIMA.
3. **DATE**: PARA ALMACENAR FECHAS Y HORAS.
4. **TIMESTAMP**: PARA ALMACENAR FECHAS Y HORAS DE MAYOR PRECISIÓN.
5. **BLOB**: PARA DATOS BINARIOS GRANDES.
6. **CLOB**: PARA TEXTO LARGO.
7. **BOOLEAN**: PARA VALORES VERDADEROS O FALSOS.

MONGODB:

MONGODB ES UNA BASE DE DATOS NOSQL, POR LO QUE NO TIENE UN ESQUEMA FIJO DE TIPOS DE DATOS COMO LAS BASES DE DATOS RELACIONALES. EN MONGODB, LOS DATOS SE ALMACENAN EN DOCUMENTOS BSON (BINARY JSON) Y LOS TIPOS DE DATOS SON DINÁMICOS, LO QUE SIGNIFICA QUE PUEDES ALMACENAR DIFERENTES TIPOS DE DATOS EN UN CAMPO.

STRING: PARA CADENAS DE TEXTO.

NUMBER: PARA NÚMEROS ENTEROS O DE PUNTO FLOTANTE.

DATE: PARA FECHAS Y HORAS.

BOOLEAN: PARA VALORES VERDADEROS O FALSOS.

ARRAY: PARA ALMACENAR LISTAS DE VALORES.

OBJECT: PARA OBJETOS INCRUSTADOS DENTRO DE DOCUMENTOS.

NULL: PARA VALORES NULOS.

BINARY DATA: PARA DATOS BINARIOS.

OBJECTID: PARA IDENTIFICADORES ÚNICOS DE DOCUMENTOS.

INSERT INTO:

LA INSTRUCCIÓN
INSERT INTO SE
UTILIZA PARA AGREGAR
NUEVOS REGISTROS A UNA TABLA

INSERT INTO NOMBRE_TABLA
(NOMBRE_COLUMNNA1,
NOMBRE_COLUMNNA2)

VALUES ('VALOR_TEXTO', VALOR_NUMERO);

```
INSERT INTO Empleados (Nombre, Edad, Departamento)  
VALUES ('Ana', 28, 'Ventas');
```

WHERE:

LA CLÁUSULA WHERE
SE UTILIZA PARA
FILTRAR RESULTADOS
EN UNA CONSULTA
SELECT O PARA
ESPECIFICAR
CONDICIONES EN
OTRAS OPERACIONES

SELECT COLUMNA/S
FROM TABLA
WHERE CONDICION;

```
SELECT Nombre  
FROM Empleados  
WHERE Edad > 30;
```

UPDATE:

LA INSTRUCCIÓN
UPDATE SE UTILIZA
PARA MODIFICAR
DATOS EXISTENTES EN
UNA TABLA.

```
UPDATE TABLA  
SET COLUMNA = REGISTRO_NUEVO  
WHERE CONDICION;
```

```
UPDATE Empleados
```

```
SET Departamento = 'Recursos Humanos'  
WHERE Nombre = 'Ana';
```

DELETE:

LA INSTRUCCIÓN
DELETE SE UTILIZA
PARA ELIMINAR
REGISTROS DE UNA
TABLA QUE CUMPLAN
CIERTAS CONDICIONES

DELETE
FROM TABLA
WHERE CONDICION;

```
DELETE FROM Empleados  
WHERE Edad < 25;
```

OPERADORES

OPERADORES COMUNES EN SQL:

=: OPERADOR DE IGUALDAD.

<> O !=: OPERADOR DE DESIGUALDAD.

> Y <: OPERADORES DE MAYOR Y MENOR QUE.

>= Y <=: OPERADORES DE MAYOR O IGUAL QUE Y MENOR O IGUAL QUE.

AND, OR: OPERADORES LÓGICOS PARA COMBINAR CONDICIONES.

LIKE: OPERADOR UTILIZADO EN LAS CLÁUSULAS WHERE PARA BUSCAR PATRONES EN TEXTO.

BETWEEN: OPERADOR PARA BUSCAR VALORES DENTRO DE UN RANGO.

IN: OPERADOR PARA BUSCAR VALORES QUE COINCIDAN CON UNA LISTA DE VALORES ESPECÍFICOS.

IS NULL Y IS NOT NULL: PARA VERIFICAR SI UN VALOR ES NULO O NO NULO.

FUNCIONES

FUNCIONES DE AGREGACIÓN:

ESTAS FUNCIONES SE UTILIZAN PARA REALIZAR CÁLCULOS EN CONJUNTOS DE DATOS Y SUELEN UTILIZARSE EN COMBINACIÓN CON LA CLÁUSULA GROUP BY PARA RESUMIR DATOS.

- **SUM()**: CALCULA LA SUMA DE LOS VALORES EN UNA COLUMNA.
- **AVG()**: CALCULA EL PROMEDIO DE LOS VALORES EN UNA COLUMNA.
- **COUNT()**: CUENTA EL NÚMERO DE FILAS O VALORES EN UNA COLUMNA.
- **MAX()**: ENCUENTRA EL VALOR MÁXIMO EN UNA COLUMNA.
- **MIN()**: ENCUENTRA EL VALOR MÍNIMO EN UNA COLUMNA.

FUNCIONES DE TEXTO:

- LENGTH(): DEVUELVE LA LONGITUD DE UNA CADENA DE TEXTO.
- UPPER(): CONVIERTE UNA CADENA DE TEXTO A MAYÚSCULAS.
- LOWER(): CONVIERTE UNA CADENA DE TEXTO A MINÚSCULAS.
- SUBSTR(): DEVUELVE UNA SUBCADENA DE UNA CADENA DE TEXTO.
- TRIM(): ELIMINA ESPACIOS EN BLANCO DE LOS EXTREMOS DE UNA CADENA DE TEXTO.

FUNCIONES DE FECHA Y HORA:

CURRENT_TIMESTAMP(): DEVUELVE LA FECHA Y HORA ACTUALES.

DATE(): EXTRAÉ LA PARTE DE FECHA DE UNA COLUMNA DATETIME.

TIME(): EXTRAÉ LA PARTE DE HORA DE UNA COLUMNA DATETIME.

STRFTIME(): FORMATEA FECHAS Y HORAS SEGÚN UN PATRÓN ESPECÍFICO.

BETWEEN:

EL OPERADOR BETWEEN SE UTILIZA PARA FILTRAR RESULTADOS DENTRO DE UN RANGO ESPECIFICADO. PUEDES USARLO EN COMBINACIÓN CON LOS OPERADORES AND Y NOT PARA DEFINIR EL RANGO DE VALORES.

```
SELECT NOMBRE, EDAD  
      FROM EMPLEADOS  
 WHERE EDAD BETWEEN 25 AND 35;
```

EN ESTE EJEMPLO, SE SELECCIONAN LOS NOMBRES Y
EDADES DE LOS EMPLEADOS CUYAS EDADES ESTÁN ENTRE
25 Y 35 AÑOS, INCLUSIVE.

LIMIT Y OFFSET:

- LA CLÁUSULA **LIMIT** SE UTILIZA PARA LIMITAR EL NÚMERO DE FILAS DEVUELTAS POR UNA CONSULTA.

```
SELECT NOMBRE  
FROM EMPLEADOS  
LIMIT 10;
```

ESTO DEVOLVERÁ SOLO LOS PRIMEROS 10 REGISTROS DE LA TABLA "EMPLEADOS".

LA CLÁUSULA **OFFSET** SE UTILIZA JUNTO CON LIMIT PARA PAGINAR RESULTADOS. DEFINE EL NÚMERO DE FILAS QUE SE DEBEN OMITIR ANTES DE APLICAR LIMIT.

```
SELECT NOMBRE  
FROM EMPLEADOS  
LIMIT 10 OFFSET 10;
```

ESTO DEVOLVERÁ LOS REGISTROS DEL 11 AL 20 DE LA TABLA "EMPLEADOS".

ORDER BY:

LA CLÁUSULA ORDER BY SE UTILIZA PARA ORDENAR LOS RESULTADOS DE UNA CONSULTA EN FUNCIÓN DE UNA O MÁS COLUMNAS. PUEDES ESPECIFICAR SI DESEAS ORDENAR EN ORDEN ASCENDENTE (ASC) O DESCENDENTE (DESC).

```
SELECT NOMBRE, EDAD  
FROM EMPLEADOS  
ORDER BY EDAD DESC;
```

EN ESTE EJEMPLO, LOS RESULTADOS SE ORDENARÁN EN FUNCIÓN DE LA COLUMNA "EDAD" EN ORDEN DESCENDENTE, LO QUE MOSTRARÁ A LOS EMPLEADOS MÁS JÓVENES PRIMERO.

GROUP BY:

LA CLÁUSULA GROUP BY SE UTILIZA PARA AGRUPAR FILAS DE DATOS EN FUNCIÓN DE LOS VALORES DE UNA O MÁS COLUMNAS. ESTO ES ÚTIL CUANDO DESEAS REALIZAR CÁLCULOS AGREGADOS, COMO SUMAS O PROMEDIOS, EN GRUPOS DE DATOS

```
SELECT DEPARTAMENTO, COUNT(*) AS  
CANTIDADEMPLEADOS  
FROM EMPLEADOS  
GROUP BY DEPARTAMENTO;
```

EN ESTE EJEMPLO, SE AGRUPAN LOS EMPLEADOS POR DEPARTAMENTO Y SE CUENTA CUÁNTOS EMPLEADOS HAY EN CADA DEPARTAMENTO.

NORMALIZACION

LA NORMALIZACIÓN ES UN PROCESO EN LA GESTIÓN DE BASES DE DATOS RELACIONALES QUE TIENE COMO OBJETIVO ORGANIZAR LA ESTRUCTURA DE UNA BASE DE DATOS PARA ELIMINAR LA REDUNDANCIA DE DATOS Y MEJORAR LA INTEGRIDAD DE LOS DATOS. LA NORMALIZACIÓN SE DIVIDE EN VARIAS FORMAS NORMALES (1NF, 2NF, 3NF, BCNF, 4NF, 5NF, ETC.), Y CADA UNA DE ELLAS ESTABLECE REGLAS ESPECÍFICAS PARA DISEÑAR TABLAS DE BASES DE DATOS DE MANERA EFICIENTE. AQUÍ TE PROPORCIONO UNA BREVE INTRODUCCIÓN A LAS TRES PRIMERAS FORMAS NORMALES:

LA NORMALIZACIÓN SE REALIZA PARA REDUCIR LA REDUNDANCIA DE DATOS Y EVITAR PROBLEMAS COMO LA INSERCIÓN, ACTUALIZACIÓN Y ELIMINACIÓN DE DATOS INCORRECTOS O INCONSISTENTES. SIN EMBARGO, ES IMPORTANTE TENER EN CUENTA QUE LA NORMALIZACIÓN EXCESIVA PUEDE LLEVAR A UN AUMENTO EN LAS OPERACIONES DE UNIÓN (JOIN) EN LAS CONSULTAS, LO QUE PODRÍA AFECTAR EL RENDIMIENTO. POR LO TANTO, EL DISEÑO DE LA BASE DE DATOS DEBE EQUILIBRAR LA NORMALIZACIÓN CON LA NECESIDAD DE CONSULTAS EFICIENTES.

CABE MENCIONAR QUE EXISTEN MÁS FORMAS NORMALES, COMO LA FORMA NORMAL DE BOYCE-CODD (BCNF) Y LA CUARTA Y QUINTA FORMA NORMAL (4NF Y 5NF), QUE ABORDAN CASOS MÁS ESPECÍFICOS DE REDUNDANCIA DE DATOS Y DEPENDENCIAS FUNCIONALES. EL PROCESO DE NORMALIZACIÓN SE ADAPTA SEGÚN LAS NECESIDADES ESPECÍFICAS DE LA BASE DE DATOS Y SU DISEÑO.

PRIMERA FORMA NORMAL (1NF):

- EN 1NF, LOS DATOS EN CADA COLUMNA DE UNA TABLA DEBEN SER ATÓMICOS, ES DECIR, INDIVISIBLES. NO DEBE HABER MÚLTIPLES VALORES EN UNA SOLA CELDA.
- CADA FILA EN LA TABLA DEBE TENER UN IDENTIFICADOR ÚNICO, GENERALMENTE UNA CLAVE PRIMARIA.

SEGUNDA FORMA NORMAL (2NF):

- PARA CUMPLIR CON 2NF, UNA TABLA DEBE ESTAR EN 1NF Y, ADEMÁS, TODOS LOS ATRIBUTOS NO CLAVE DEBEN DEPENDER COMPLETAMENTE DE LA CLAVE PRIMARIA.
- ESTO SIGNIFICA QUE SI TIENES UNA CLAVE PRIMARIA COMPUESTA (VARIAS COLUMNAS), CADA ATRIBUTO EN LA TABLA DEBE DEPENDER DE TODAS LAS COLUMNAS DE LA CLAVE PRIMARIA Y NO DE UNA PARTE DE ELLA.

TERCERA FORMA NORMAL (3NF):

- UNA TABLA QUE CUMPLE CON 2NF Y DONDE LOS ATRIBUTOS NO CLAVE NO DEPENDEN TRANSITIVAMENTE DE LA CLAVE PRIMARIA CUMPLE CON LA 3NF.
- ESTO SIGNIFICA QUE NO DEBE HABER DEPENDENCIAS TRANSITIVAS ENTRE LOS ATRIBUTOS NO CLAVE. SI UN ATRIBUTO NO CLAVE DEPENDE DE OTRO ATRIBUTO NO CLAVE, DEBE SEPARARSE EN UNA TABLA DIFERENTE.

DEPENDENCIA FUNCIONAL:

LA DEPENDENCIA FUNCIONAL SE REFIERE A UNA RELACIÓN ENTRE DOS CONJUNTOS DE ATRIBUTOS EN UNA TABLA DE BASE DE DATOS.

SE DICE QUE UN CONJUNTO DE ATRIBUTOS A DEPENDE FUNCIONALMENTE DE OTRO CONJUNTO DE ATRIBUTOS B SI, PARA CADA VALOR ÚNICO DE B, HAY UN SOLO VALOR CORRESPONDIENTE EN A.

EN NOTACIÓN, ESTO SE DENOTA COMO $B \rightarrow A$, LO QUE SIGNIFICA QUE A DEPENDE FUNCIONALMENTE DE B.

EJEMPLO:

SI TIENES UNA TABLA DE EMPLEADOS CON COLUMNAS {ID, NOMBRE, DEPARTAMENTO}, PUEDES DECIR QUE EL ATRIBUTO "NOMBRE" DEPENDE FUNCIONALMENTE DEL ATRIBUTO "ID" SI CADA EMPLEADO TIENE UN NOMBRE ÚNICO PARA SU ID.

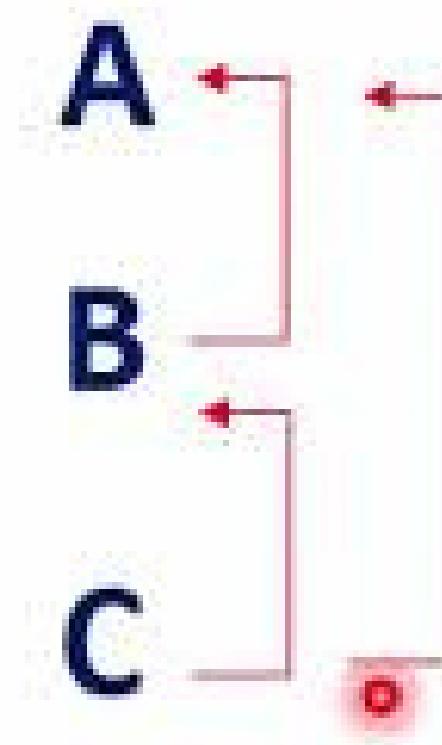
DEPENDENCIA TRANSITIVA:

LA DEPENDENCIA TRANSITIVA ES UN TIPO ESPECÍFICO DE DEPENDENCIA FUNCIONAL. SE PRODUCE CUANDO UN ATRIBUTO A DEPENDE FUNCIONALMENTE DE OTRO ATRIBUTO B A TRAVÉS DE UN TERCER ATRIBUTO C.

ESTO SIGNIFICA QUE SI CONOCES EL VALOR DE C, PUEDES DETERMINAR EL VALOR DE B Y, POR LO TANTO, TAMBIÉN PUEDES DETERMINAR EL VALOR DE A.

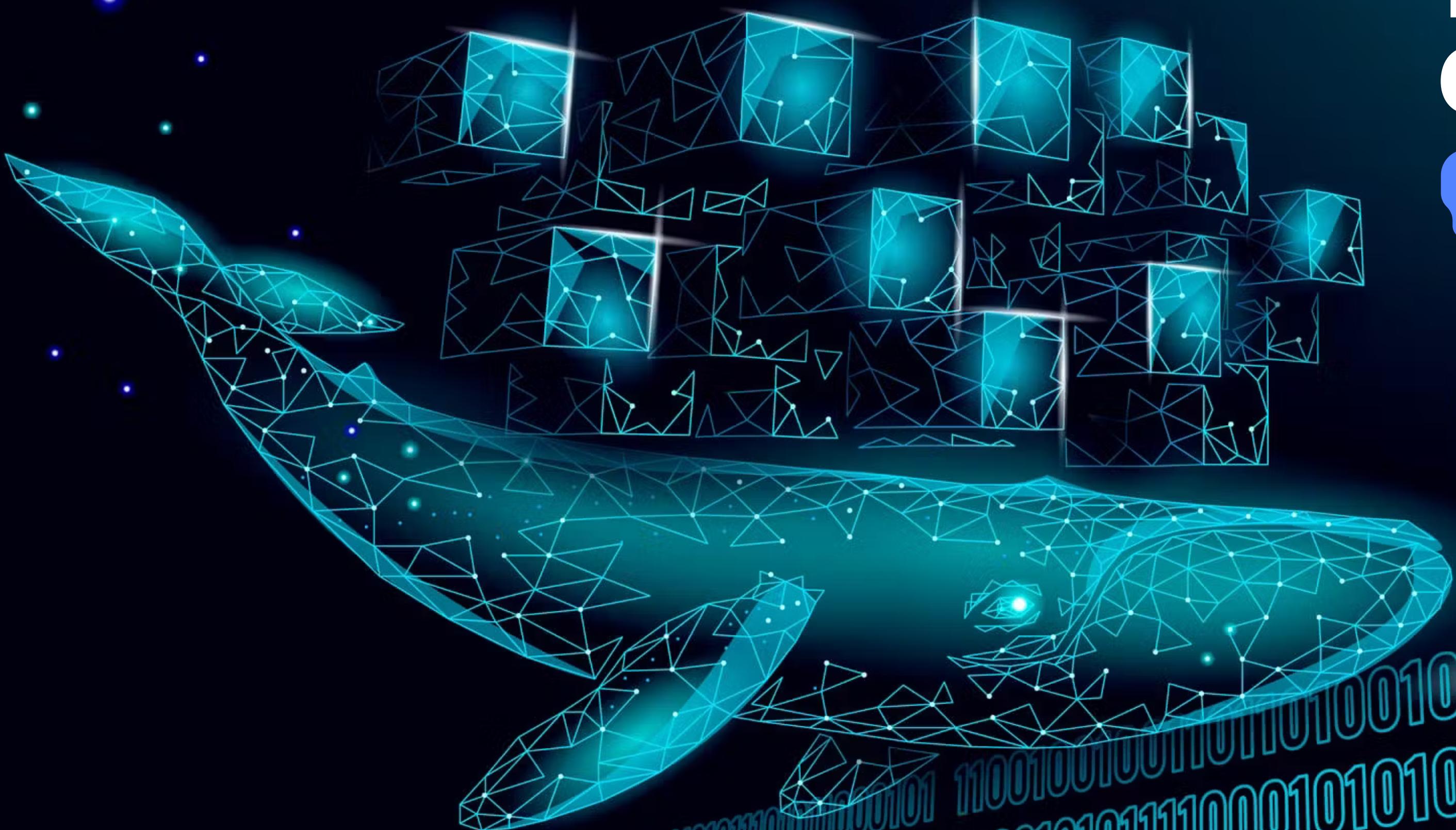
EN NOTACIÓN, ESTO SE DENOTA COMO $B \rightarrow C$ Y $C \rightarrow A$, LO QUE IMPLICA $B \rightarrow A$.

Dependencia Transitiva



Muchas Gracias!

↓ NOS VEMOS EN
LA 2DA PARTE ↓



101001010110 1000101 110010010110 10010101010001100101
100010010110 101001010111000101010010011 0110 101001010101
10001010111 001010 010011101101100101 101001010101
101001010101 11000101110010010011 011010010101