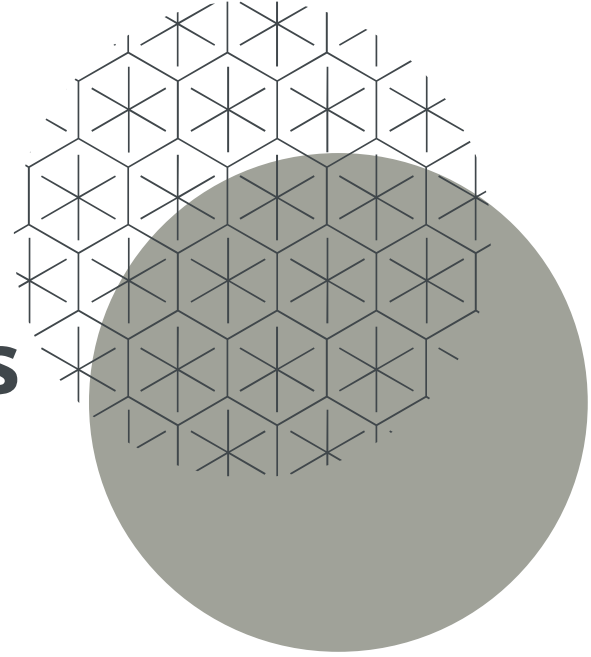




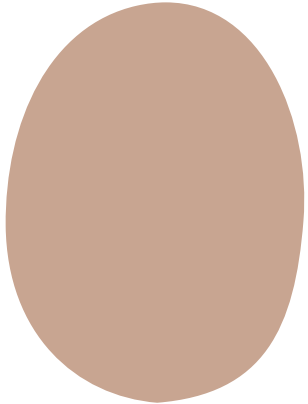
Sistemas de Almacenamiento.

Contenidos

- Introducción a las bases de datos
- Bases de datos relacionales
- Bases de datos no relacionales
- Comparación entre bases de datos relacionales y no relacionales



Introducción a las bases de datos

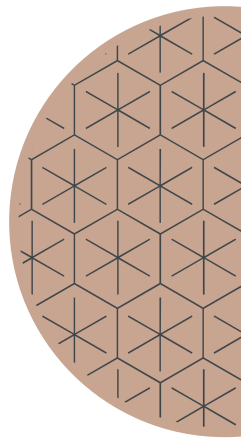


Definición de BBDD

Una **base de datos** es un conjunto organizado de datos que se almacenan y gestionan de manera estructurada para facilitar el acceso, la recuperación y la manipulación de la información. Las bases de datos son una herramienta fundamental en la informática y se utilizan en una amplia variedad de aplicaciones, desde sistemas de gestión de empresas hasta sitios web y aplicaciones móviles.

Importancia de las Bases de Datos en el Big Data

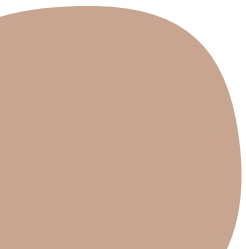
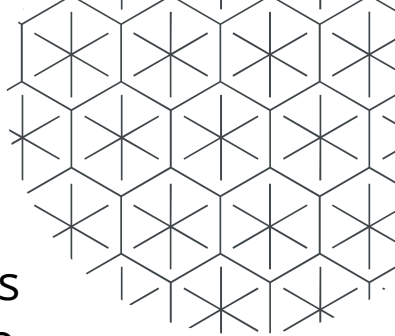
Las **bases de datos** desempeñan un papel fundamental en el ámbito del Big Data. Big Data se refiere al manejo, procesamiento y análisis de conjuntos de datos extremadamente grandes y complejos, y las bases de datos son esenciales para gestionar y acceder a estos datos de manera eficiente.



Tipos de Bases de Datos

Existen dos categorías principales de bases de datos: las bases de datos relacionales (**SQL**) y las bases de datos no relacionales (**NoSQL**).

Cada una de estas categorías tiene diferentes tipos y modelos de bases de datos. A continuación, se describen los principales tipos de bases de datos en cada categoría:

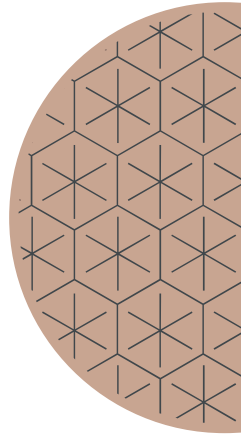


Bases de Datos No Relacionales (NoSQL)

- **Bases de Datos de Documentos:** Almacenan datos en documentos semi-estructurados (por ejemplo, JSON o XML) y son ideales para aplicaciones web y móviles. Ejemplos incluyen MongoDB y CouchDB.
- **Bases de Datos de Columnas:** Almacenan datos en columnas, lo que es eficiente para consultas analíticas. Pueden manejar grandes volúmenes de datos. Ejemplos incluyen Apache Cassandra y HBase.
- **Bases de Datos Clave-Valor:** Almacenan datos en pares clave-valor y son altamente escalables. Son adecuadas para cachés y sistemas de almacenamiento de alta velocidad. Ejemplos incluyen Redis y Amazon DynamoDB.

Bases de Datos No Relacionales (NoSQL)

- **Bases de Datos de Grafo:** Almacenan datos en estructuras de grafo y son eficaces para representar y consultar relaciones complejas. Ejemplos incluyen Neo4j y Amazon Neptune.
- **Bases de Datos de Tiempo Real (Time-Series Databases):** Están diseñadas para datos de series temporales, como registros de sensores y métricas. Ejemplos incluyen InfluxDB y Prometheus.
- **Bases de Datos de Búsqueda:** Están optimizadas para la búsqueda de texto completo y la indexación de contenido. Ejemplos incluyen Elasticsearch y Solr.





Bases de datos relacionales



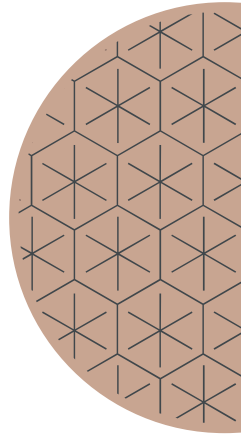
Comprensión de los modelos de datos relacionales

Los **modelos de datos relacionales** son una parte fundamental de la gestión de bases de datos en sistemas de gestión de bases de datos relacionales (RDBMS). Estos modelos proporcionan una forma estructurada de organizar y almacenar datos en tablas con filas y columnas.

Relaciones

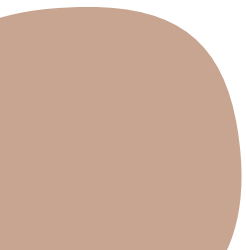
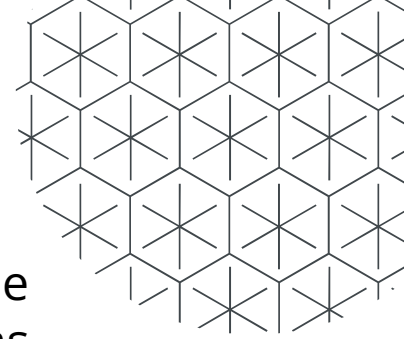
La estructura del modelo relacional se basa en el concepto de tablas, que también se conocen como relaciones. Cada tabla tiene un identificador único denominado clave primaria, y las tablas pueden relacionarse entre sí mediante el uso de claves externas.

Las relaciones entre tablas en un modelo relacional pueden ser de uno a uno, de uno a muchos o de muchos a muchos.



Atributos y grado

Los atributos de una tabla son las columnas, o campos, que representan las características o propiedades de las entidades que la tabla representa. El grado de una tabla es el número de atributos que tiene. Por ejemplo, una tabla de empleados puede tener un grado de 5, si tiene columnas para el ID, el nombre, el cargo, el salario y el departamento del empleado.

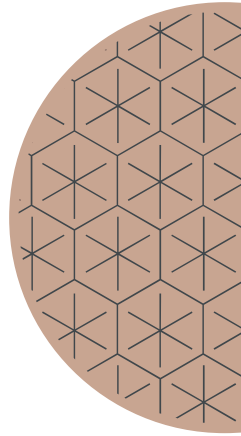


Tuplas y cardinalidad

Las tuplas de una tabla son las filas, o registros, que representan las entidades individuales que representa la tabla. La cardinalidad de una tabla es el número de tuplas que tiene. Por ejemplo, si una tabla de empleados tiene 10 filas, su cardinalidad es 10.

Valores nulos

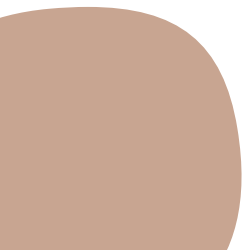
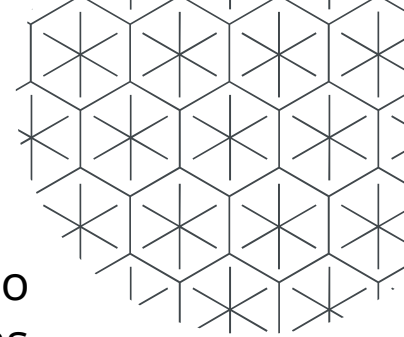
En un modelo relacional, los valores nulos se utilizan para representar datos perdidos o desconocidos. Un valor nulo es diferente de una cadena vacía o un valor cero, ya que indica que los datos no se conocen o no son aplicables, en lugar de simplemente estar ausentes.



Dominio y Atributo

En un modelo relacional, un atributo es una columna, o campo, que representa una característica o propiedad de las entidades que representa la tabla. El dominio de atributos de una tabla es el conjunto de valores permitidos para un atributo determinado.

Por ejemplo, considere una tabla de empleados con un atributo para el salario. Si el dominio del atributo salario es simple, cada empleado puede tener como máximo un valor de salario.



Claves candidatas y primarias

En un modelo relacional, una clave candidata es una columna o conjunto de columnas que pueden identificar de forma única cada fila de una tabla. Una clave candidata debe cumplir las siguientes propiedades

- **Unicidad:** Cada valor de la clave candidata debe ser único dentro de la tabla.
- **No nulidad:** La clave candidata no puede contener valores nulos.
- **Irreducibilidad:** La clave candidata no puede ser un subconjunto de ninguna otra clave candidata.

Claves candidatas y primarias

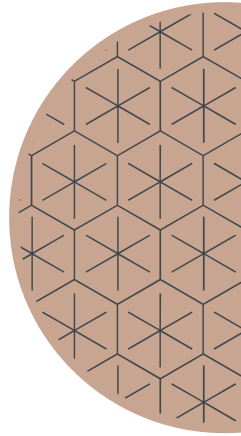
Una tabla puede tener una o **varias** claves **candidatas**, pero sólo se puede elegir una como clave principal. La **clave primaria** es la clave candidata que se utiliza para **identificar** de forma exclusiva cada fila de la **tabla**, y se suele utilizar como base para las relaciones de clave externa con otras tablas.

Las **claves alternativas** son claves candidatas que no se eligen como clave principal, pero que pueden utilizarse para identificar filas de la tabla.

Claves candidatas y primarias

Las **claves externas** son columnas o conjuntos de columnas de una tabla que hacen referencia a la clave principal de otra tabla y se utilizan para establecer una relación entre las dos tablas.

Por ejemplo, considere una tabla de empleados con una clave primaria de ID de empleado. La tabla de empleados también puede tener una clave alternativa de número de la seguridad social, y una clave externa que haga referencia al ID de departamento en una tabla de departamentos. Esto permite buscar y recuperar datos de forma eficaz en varias tablas.



Reglas de integridad del modelo relacional

En un modelo relacional, las reglas de integridad se utilizan para garantizar la corrección y coherencia de los datos de la base de datos.

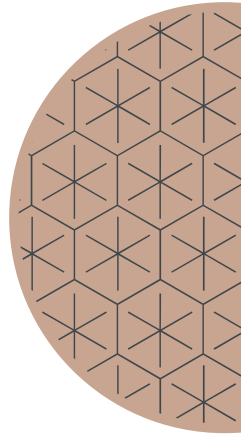
- **Integridad de entidad:** Esta regla garantiza que cada fila de una tabla tiene una clave primaria única, y que la clave primaria no puede ser nula.
- **Integridad referencial:** Esta regla garantiza que las claves externas de una tabla se refieren a claves primarias válidas de la tabla relacionada.

Reglas de integridad del modelo relacional

- **Integridad de dominio:** Esta regla garantiza que los valores de una columna se ajustan al dominio de atributo de la columna.
- **Integridad definida por el usuario:** Esta regla permite a los usuarios especificar restricciones y reglas personalizadas que deben cumplir los datos de la base de datos.

Normalización

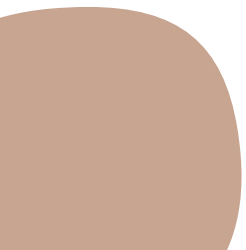
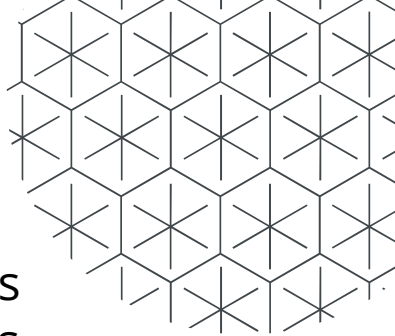
La **normalización** es el proceso de organizar los datos en una base de datos relacional para minimizar la redundancia y la dependencia, y para mejorar la integridad y el rendimiento de la base de datos. La normalización suele realizarse como parte del proceso de diseño de la base de datos, y consiste en descomponer estructuras de datos complejas en estructuras más simples y atómicas.



Normalización

Existen varios niveles de normalización, cada uno de los cuales se basa en los anteriores y añade restricciones y reglas adicionales para organizar los datos.

- **Primera forma normal (1NF):** En 1NF, cada tabla debe tener una clave primaria y cada columna debe contener valores atómicos (es decir, valores que no pueden dividirse posteriormente).
- **Segunda forma normal (2NF):** En 2NF, cada tabla debe satisfacer 1NF, y todas las columnas de clave no primaria deben depender de la clave primaria.



Normalización

- **Tercera forma normal (3NF):** En 3NF, cada tabla debe satisfacer 2NF, y no pueden existir dependencias entre los atributos que no forman parte de la clave primaria.
- **La forma normal Boyce-Codd (BCNF):** es un nivel de normalización superior a la tercera forma normal (3NF). En BCNF, una tabla está en tercera forma normal y todos los determinantes son claves candidatas. Un determinante es un atributo del que depende funcionalmente otro atributo.

Origen SQL

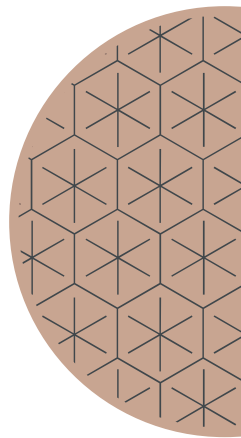
SQL (Structured Query Language) es un lenguaje de programación desarrollado a principios de los años 70 para gestionar datos almacenados en sistemas de gestión de bases de datos relacionales (RDBMS).

SQL ha evolucionado a lo largo de los años, con diversas versiones publicadas y estandarizadas por organizaciones como el American National Standards Institute (ANSI) y la International Organization for Standardization (ISO).

Tipos de datos

Estos son algunos de los tipos de datos más comunes que se utilizan en SQL:

- **INTEGER:** Un número entero (sin punto decimal).
- **DECIMAL:** Un número con punto decimal.
- **CHAR:** Cadena de longitud fija (255 caracteres como máximo).
- **VARCHAR:** Cadena de longitud variable (máx. 255 caracteres).
- **TEXT:** Una cadena de longitud variable con una longitud máxima de 65.535 caracteres.
- **DATE:** Un valor de fecha en el formato AAAA-MM-DD.



Lenguaje de definición de datos

El lenguaje de definición de datos (**DDL**) es un conjunto de comandos que se utiliza para definir la estructura de una base de datos. Se utiliza para crear, modificar y eliminar objetos de base de datos como tablas, índices y vistas.

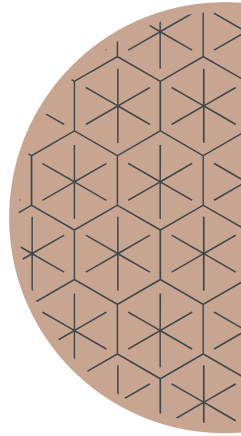
- **CREATE/DROP TABLE:** Se utiliza para crear una nueva tabla en la base de datos. Drop para borrar.
- **ALTER TABLE:** Se utiliza para modificar la estructura de una tabla existente, como añadir o eliminar columnas.

Lenguaje de definición de datos

- **CREATE/DROP INDEX:** sirve para crear un índice en una tabla, lo que puede mejorar el rendimiento de las consultas que acceden a esa tabla. Drop lo elimina.
- **CREATE/DROP VIEW:** Permite crear/borrar una tabla virtual a partir de una sentencia SQL.
- **TRUNCATE TABLE:** sirve para eliminar todos los datos de una tabla, pero a diferencia de DROP TABLE, no elimina la tabla en sí.

Lenguaje de manipulación de datos

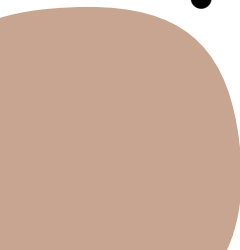

El lenguaje de manipulación de datos (**DML**) es un tipo de lenguaje de programación utilizado para manipular y recuperar datos en una base de datos. Los comandos DML se utilizan para insertar, actualizar y eliminar datos en una base de datos, así como para recuperar datos de una base de datos.



Lenguaje de manipulación de datos



Los comandos DML más comunes son:

- **INSERT:** El comando INSERT se utiliza para insertar nuevos registros en una tabla de base de datos.
 - **UPDATE:** El comando UPDATE se utiliza para actualizar datos existentes en una tabla de base de datos.
 - **DELETE:** El comando DELETE se utiliza para eliminar registros de una tabla de base de datos.
 - **SELECT:** El comando SELECT se utiliza para recuperar datos de una tabla o varias tablas de la base de datos.
- 
- 

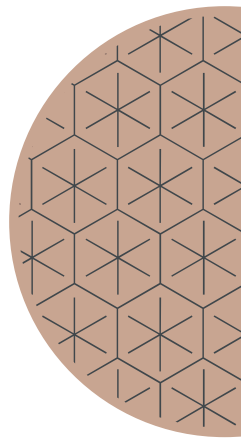
Lenguaje de control de datos

El lenguaje de control de datos (**DCL**) es un tipo de lenguaje informático utilizado para controlar el acceso a los datos de una base de datos y su manipulación. DCL incluye comandos que se utilizan para conceder y revocar permisos sobre objetos de base de datos, como tablas y vistas, así como para controlar transacciones.

Lenguaje de control de datos

Algunos comandos comunes de DCL incluyen:

- **GRANT:** El comando GRANT se utiliza para dar acceso a usuarios o roles a objetos o privilegios de la base de datos.
- **REVOKE:** El comando REVOKE se utiliza para eliminar el acceso de usuarios o roles a objetos o privilegios de la base de datos.
- **SAVEPOINT:** El comando SAVEPOINT se utiliza para crear un punto dentro de una transacción en el que los cambios realizados hasta ese momento se pueden revertir si es necesario.



Lenguaje de control de datos

- **ROLLBACK:** El comando ROLLBACK se utiliza para deshacer todos los cambios realizados en una transacción hasta un punto de guardado especificado o hasta el inicio de la transacción.
- **COMMIT:** El comando COMMIT se utiliza para hacer permanentes los cambios realizados en una transacción y para finalizar la transacción.

Escritura de consultas SQL

La escritura de consultas SQL (Structured Query Language) es esencial para interactuar con bases de datos relacionales y recuperar información de ellas.

La consulta SELECT se utiliza para recuperar datos de una tabla.

```
SELECT columna1, columna2
```

```
FROM tabla
```

```
WHERE condición;
```

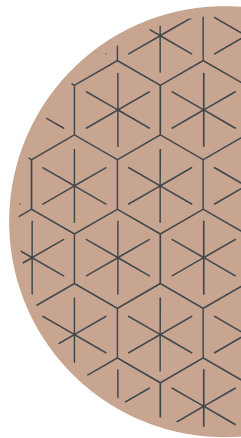
Escritura de consultas SQL

Cuando deseas combinar datos de múltiples tablas, utilizas JOIN.

```
SELECT t1.columna1, t2.columna2  
FROM tabla1 AS t1  
JOIN tabla2 AS t2 ON t1.clave = t2.clave;
```

La consulta INSERT se usa para agregar filas a una tabla.

```
INSERT INTO tabla (columna1, columna2) VALUES (valor1, valor2);
```



Lenguaje de control de datos

La consulta UPDATE se utiliza para actualizar registros en una tabla.

```
UPDATE tabla
```

```
SET columna = nuevo_valor
```

```
WHERE condición;
```

Más información: <https://www.w3schools.com/sql/>

Escritura de consultas SQL

La consulta DELETE se utiliza para eliminar registros de una tabla.

```
DELETE FROM tabla
```

```
WHERE condición;
```

Puedes utilizar GROUP BY para agrupar filas por un valor específico y aplicar funciones de agregación como SUM, AVG, COUNT, etc.

```
SELECT columna, función_agregación(columna)
```

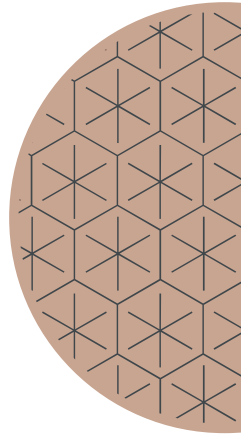
```
FROM tabla
```

```
GROUP BY columna;
```

Escritura de consultas SQL

Puede utilizar GROUP BY para agrupar filas en función de los valores de una columna y filtrar los grupos con la cláusula HAVING en función de una condición agregada.

```
SELECT Ciudad, COUNT(*) AS TotalClientes  
FROM Clientes  
GROUP BY Ciudad  
HAVING TotalClientes > 1;
```



Lenguaje de control de datos

Puede utilizar ORDER BY para ordenar el conjunto de resultados en orden ascendente (ASC) o descendente (DESC) en función de una o varias columnas.

```
SELECT Nombre, Apellido  
FROM Clientes  
ORDER BY Apellido ASC;
```

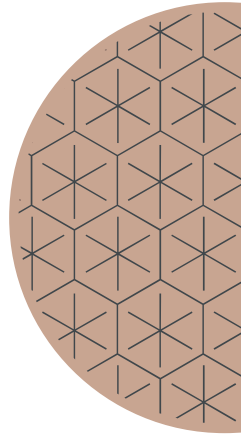
RDBMS

Un RDBMS, o Sistema de Gestión de Bases de Datos Relacionales es un tipo de software de gestión de bases de datos que se utiliza para administrar y almacenar datos en un formato estructurado y relacionado.

Los RDBMS se basan en el modelo de datos relacional, que organiza los datos en tablas compuestas por filas y columnas, y establece relaciones entre ellas.

MySQL

MySQL es un sistema de gestión de bases de datos relacionales de código abierto ampliamente utilizado. Es conocido por su rendimiento, escalabilidad y facilidad de uso. MySQL es una opción popular para aplicaciones web y empresariales.

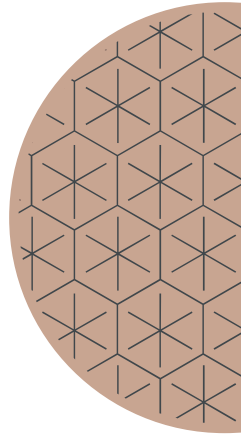


Características clave MySQL

- Soporta múltiples motores de almacenamiento, incluyendo InnoDB y MyISAM.
- Es altamente escalable y adecuado para aplicaciones de todos los tamaños.
- Ofrece una amplia comunidad de usuarios y una amplia gama de recursos en línea.
- Disponible en ediciones de código abierto y comerciales.

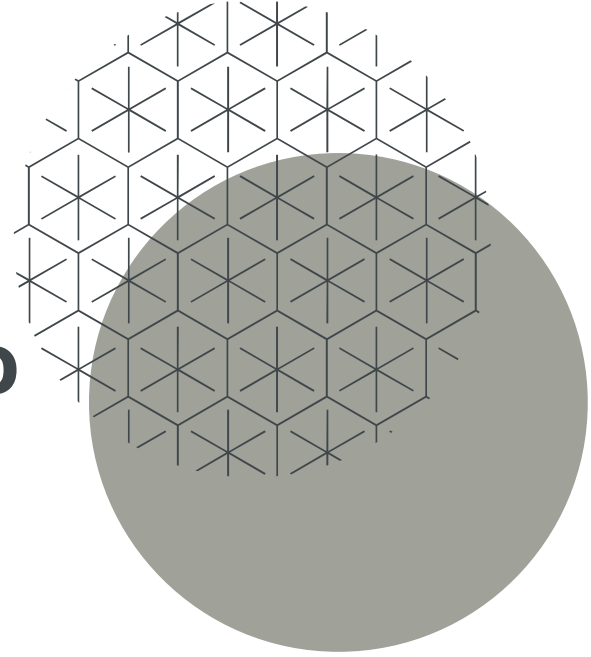
SQL Server

Microsoft SQL Server es un sistema de gestión de bases de datos relacionales desarrollado por Microsoft. Ofrece una suite completa de herramientas de gestión y es ampliamente utilizado en el entorno empresarial.

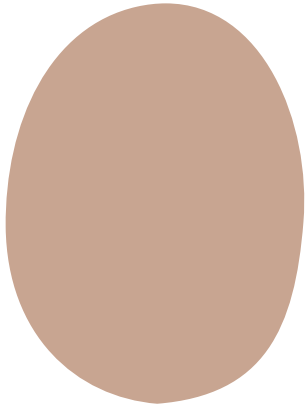


Características clave SQL Server

- Ofrece una variedad de ediciones, desde Express (gratuita) hasta Enterprise (de alto rendimiento).
- Integración con otras tecnologías de Microsoft, como .NET y Azure.
- Características avanzadas, como análisis de datos, informes y procesamiento en memoria.
- Seguridad y gestión de bases de datos avanzadas.



Bases de datos no relacionales



Introducción a las bases de datos no relacionales (NoSQL)

Las bases de datos no relacionales, comúnmente conocidas como bases de datos NoSQL (Not Only SQL), son un tipo de sistema de gestión de bases de datos que difieren de las bases de datos relacionales tradicionales en su enfoque de almacenamiento y recuperación de datos.

A diferencia de las bases de datos relacionales, que se basan en tablas y esquemas predefinidos, las bases de datos NoSQL están diseñadas para manejar datos no estructurados o semiestructurados de manera más flexible.

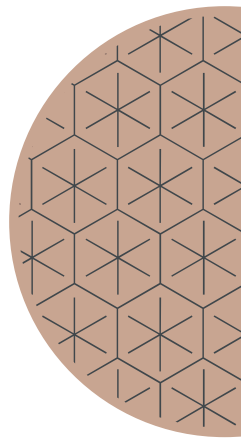
Característica Principal

Una de las características clave de las bases de datos NoSQL es que **no requieren un esquema fijo** o predefinido. Esto significa que puedes agregar nuevos campos a tus datos sin necesidad de modificar el esquema de la base de datos. Esto es particularmente útil en situaciones donde los requisitos de datos pueden cambiar con frecuencia.

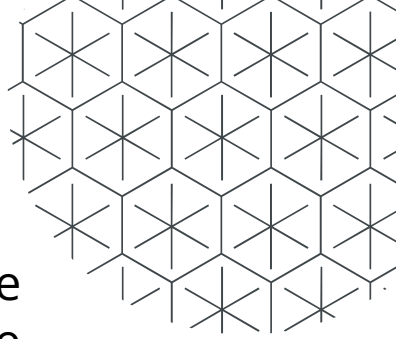
Modelado de datos en MongoDB

El modelado de datos en MongoDB se basa en el uso de documentos BSON (Binary JSON) para representar la información de una manera flexible y escalable.

Supongamos que estás diseñando una base de datos para un sistema de biblioteca.

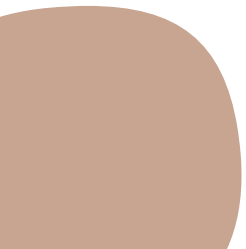


JSON



JSON, acrónimo de "JavaScript Object Notation" (Notación de objetos JavaScript), es un formato ligero de intercambio de datos fácil de leer y escribir para los humanos, y fácil de analizar y generar para las máquinas.

JSON se utiliza a menudo para la transmisión de datos entre un servidor y una aplicación web, o entre distintas partes de un sistema de software.

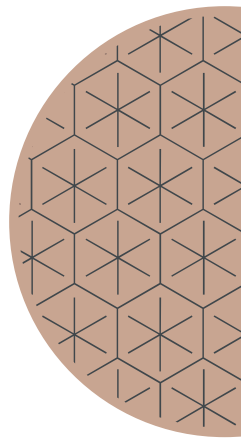


JSON

Los datos JSON se representan como una colección de pares clave-valor encerrados entre llaves {}. Cada clave es una cadena entre comillas dobles, seguida de dos puntos y, a continuación, un valor. Los valores pueden ser cadenas, números, objetos, matrices, booleanos, nulos u otros objetos JSON.

Modelado de datos en MongoDB

```
{  
  "name": "John Doe",  
  "age": 30,  
  "isStudent": false,  
  "hobbies": ["reading", "hiking", "gaming"],  
  "address": {  
    "street": "123 Main St",  
    "city": "Anytown",  
    "zipCode": "12345"  
  },  
  "email": null  
}
```



Modelado de datos en MongoDB

Paso 1: Definir las Entidades Principales

Identifica las entidades principales de tu sistema. En este caso, podrían ser Libros, Autores, Usuarios y Préstamos.

Paso 2: Diseñar el Schema

Cada entidad será representada por una colección en MongoDB. A continuación, diseñaremos los esquemas para cada una.

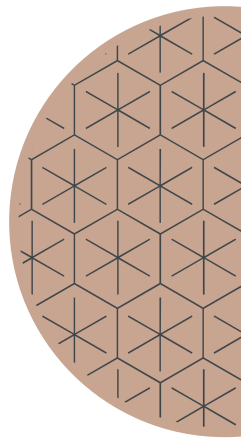
Característica Principal

Libros:

En MongoDB, no necesitas un esquema rígido, pero puedes definir un esquema flexible. Puedes diseñar un documento BSON para representar un libro de la siguiente manera:

Modelado de datos en MongoDB

```
{  
  "_id": ObjectId("unique_book_id"),  
  "titulo": "Título del Libro",  
  "autor_id": ObjectId("author_id"),  
  "ISBN": "Número ISBN",  
  "copias_disponibles": 5,  
  "prestamos": [  
    {  
      "usuario_id": ObjectId("user_id"),  
      "fecha_prestamo": ISODate("2023-10-11"),  
      "fecha_devolucion": ISODate("2023-10-25")  
    }  
  ]  
}
```



Modelado de datos en MongoDB

Autores:

```
{  
  "_id": ObjectId("unique_author_id"),  
  "nombre": "Nombre del Autor",  
  "nacionalidad": "Nacionalidad del Autor",  
  "libros_escritos": [  
    ObjectId("book_id_1"),  
    ObjectId("book_id_2")  
  ]  
}
```

Modelado de datos en MongoDB

Paso 3: Establecer las Relaciones

En MongoDB, puedes representar relaciones utilizando referencias a otros documentos a través del campo `_id`. Por ejemplo, el campo `autor_id` en el documento de libro hace referencia al documento del autor. Esto establece una relación entre libros y autores.

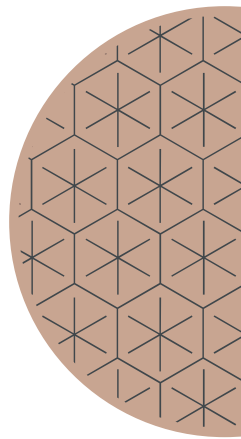
Consultas en MongoDB

Las consultas en MongoDB se realizan utilizando el lenguaje de consulta de MongoDB (MongoDB Query Language) y se ejecutan en colecciones que contienen documentos BSON.

Búsqueda de documentos:

```
db.libros.find({ autor: "Nombre del Autor" })
```

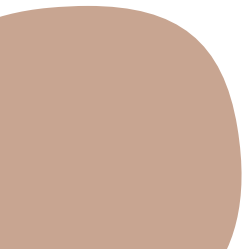
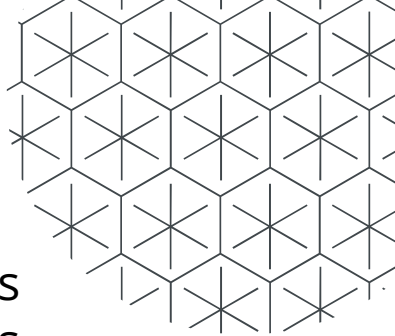
Esto recuperará todos los documentos de la colección "libros" donde el campo "autor" coincide con "Nombre del Autor".



Consultas en MongoDB

Puedes utilizar operadores para realizar consultas más específicas. Por ejemplo, para buscar todos los documentos en la colección "products" con un precio mayor a \$50, puedes hacer lo siguiente:

```
db.products.find({ price: { $gt: 50 } })
```



Consultas en MongoDB

Puedes combinar múltiples condiciones en una consulta. Por ejemplo, para encontrar documentos en la colección "orders" que tengan un valor total mayor a \$100 y estén marcados como "completados", puedes hacerlo de la siguiente manera:

```
db.orders.find({ total: { $gt: 100 }, status: "completado" })
```

Consultas en MongoDB

Puedes especificar qué campos deseas recuperar en el resultado de la consulta utilizando la opción de proyección. Por ejemplo, para obtener solo los campos "name" y "email" de los documentos de la colección "users", puedes hacerlo de la siguiente manera:

```
db.users.find({}, { name: 1, email: 1, _id: 0 })
```

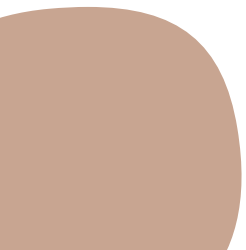
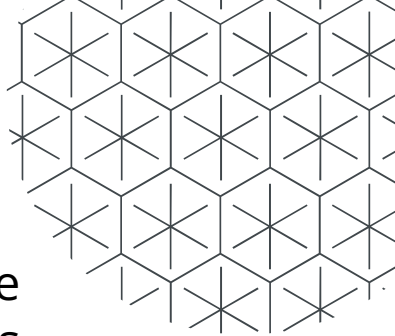
Aquí, { name: 1, email: 1, _id: 0 } significa que se incluirán los campos "name" y "email" pero se excluirá el campo "_id".

Consultas en MongoDB

Puedes ordenar los resultados de una consulta en función de un campo específico. Por ejemplo, para obtener todos los documentos de la colección "products" ordenados por el campo "price" de manera ascendente, puedes hacer lo siguiente:

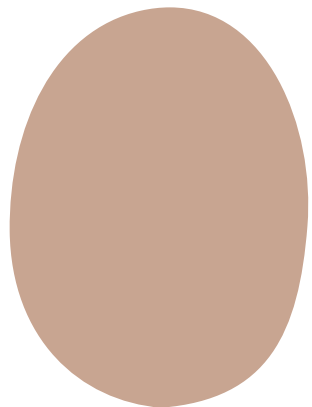
```
db.products.find().sort({ price: 1 })
```

En este caso, 1 representa un orden ascendente, mientras que -1 indicaría un orden descendente.



Consultas en MongoDB

Ejercicio: Buscar información sobre aggregators en <https://www.mongodb.com/docs/manual/aggregation/>.



FIN

