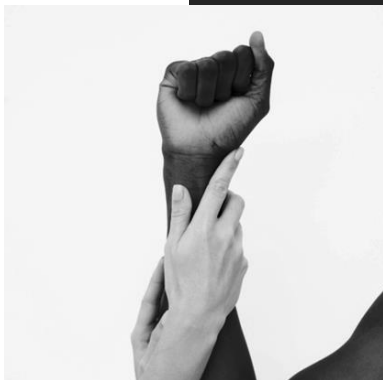


Base de Datos y SQL 1





CONTENIDO

01 Bases de datos 1

02 SQL

03 Ejercicios

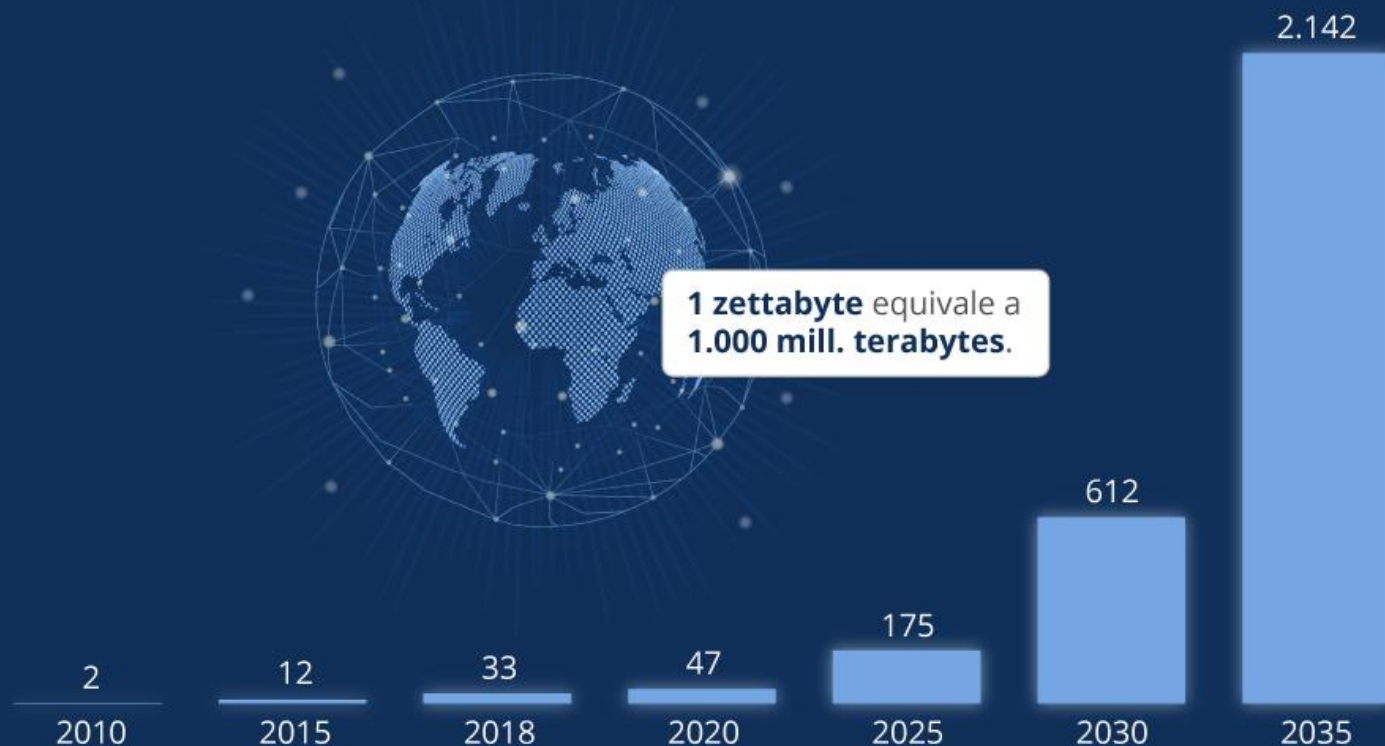


01

1. ¿Qué es una Base de Datos?

La creación de datos, a punto de explotar

Cantidad real y prevista de datos generados en todo el mundo (en zettabytes)



¿Qué es una base de datos?

Una base de datos es una recopilación organizada de información o datos estructurados, que normalmente se almacena de forma electrónica en un sistema informático.

- Oracle

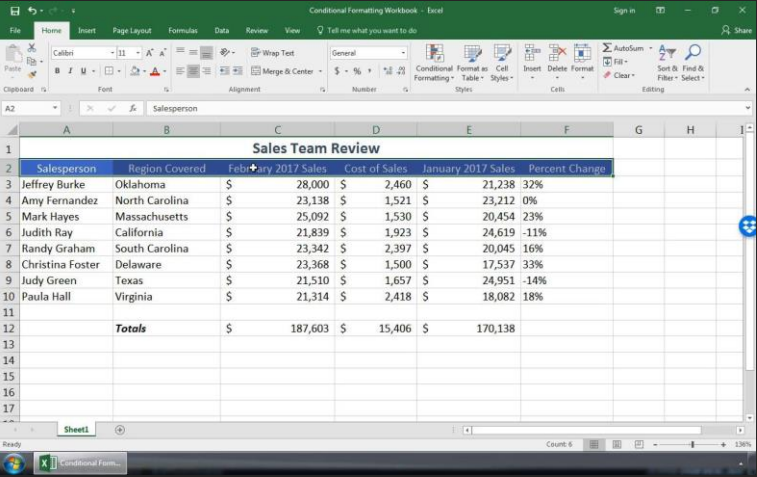
¿Qué es una base de datos?

Es un conjunto de datos organizados y relacionados entre sí, con una estructura independiente del programa que los acceda.

¿Excel es una base de datos?

Las principales diferencias entre los dos son:

- Cómo se almacenan y se manipulan los datos
- Quién puede acceder a los datos
- Cuántos datos pueden almacenarse



Sales Team Review					
Salesperson	Region Covered	February 2017 Sales	Cost of Sales	January 2017 Sales	Percent Change
Jeffrey Burke	Oklahoma	\$ 28,000	\$ 2,460	\$ 21,238	32%
Amy Fernandez	North Carolina	\$ 23,138	\$ 1,521	\$ 23,212	0%
Mark Hayes	Massachusetts	\$ 25,092	\$ 1,530	\$ 20,454	23%
Judith Ray	California	\$ 21,839	\$ 1,923	\$ 24,619	-11%
Randy Graham	South Carolina	\$ 23,342	\$ 2,397	\$ 20,045	16%
Christina Foster	Delaware	\$ 23,368	\$ 1,500	\$ 17,537	33%
Judy Green	Texas	\$ 21,510	\$ 1,657	\$ 24,951	-14%
Paula Hall	Virginia	\$ 21,314	\$ 2,418	\$ 18,082	18%
Totals		\$ 187,603	\$ 15,406	\$ 170,138	

¿Qué es una base de datos?

Una **base de datos** es una colección organizada de información o datos que están estructurados de tal manera que permiten su acceso, gestión y actualización de manera eficiente.

Características clave:

1. **Estructurada:** Los datos están organizados en tablas, columnas y filas, lo que facilita su recuperación.
2. **Persistente:** Los datos permanecen almacenados incluso después de que la aplicación que los utiliza haya dejado de funcionar.
3. **Accesible:** Permite realizar consultas para buscar información específica o hacer modificaciones.

¿Por qué necesitamos una base de datos?

1. **Organización de la información:** En la mayoría de las aplicaciones modernas, como las tiendas online, redes sociales y sistemas de gestión empresarial, las bases de datos juegan un rol fundamental para organizar grandes volúmenes de datos.
2. **Facilitan el acceso a la información:** Mediante consultas eficientes, se pueden recuperar datos rápidamente para diferentes finalidades.
3. **Seguridad y control:** Las bases de datos permiten establecer niveles de permisos para que solo ciertas personas puedan acceder o modificar los datos.
4. **Escalabilidad:** Las bases de datos pueden crecer de manera eficiente, gestionando más datos a medida que una empresa o aplicación crece.

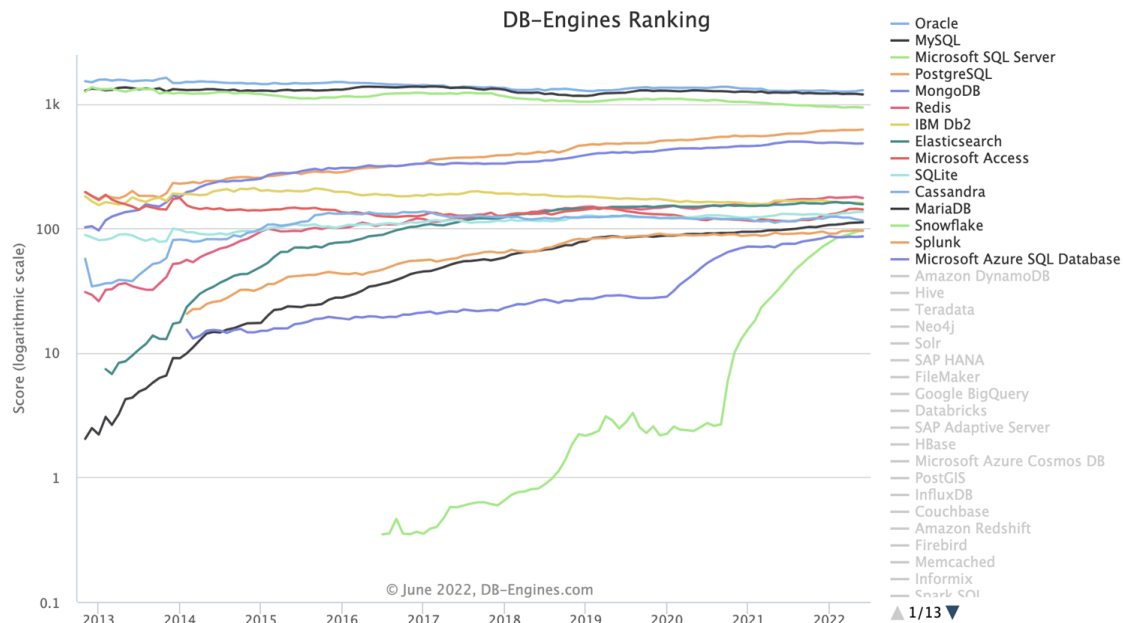
¿Qué es DBMS?

El DBMS esencialmente sirve como una interfaz entre la base de datos y los usuarios finales o programas de aplicación.

El DBMS gestiona tres cosas importantes: los datos, el motor de la base de datos que permite acceder a los datos, bloquearlos y modificarlos, y el esquema de la base de datos, que define la estructura lógica de la base de datos.

¿Qué es un motor de base de datos?

La pieza de *software* fundamental de un DBMS que se encarga de las bases de datos es lo que corrientemente llamamos el motor de bases de datos.



¿Qué es un motor de base de datos?



Tipos de Bases de Datos

Existen diferentes tipos de bases de datos que son utilizadas en función de las necesidades y del tipo de datos que se manejen:

Bases de Datos Relacionales (RDBMS)

Estas son las más comunes. En una base de datos relacional, los datos se organizan en tablas, y las relaciones entre los diferentes datos se definen mediante claves (primarias y foráneas).

Ejemplos: MySQL, PostgreSQL, SQL Server, Oracle.

Características:

Los datos están organizados en tablas.

Utilizan el lenguaje SQL para realizar operaciones.

Soportan transacciones, lo que permite mantener la consistencia de los datos.

Tipos de Bases de Datos

Bases de Datos No Relacionales (NoSQL)

Estas bases de datos se utilizan para datos que no tienen una estructura fija o que requieren una escalabilidad más flexible. Se suelen usar en aplicaciones que manejan grandes volúmenes de datos y donde la estructura de los mismos puede cambiar con el tiempo.

Ejemplos: MongoDB, Cassandra, Redis.

Características:

Datos no estructurados o semiestructurados.

Escalabilidad horizontal, lo que significa que es fácil añadir más servidores para gestionar mayores volúmenes de datos.

No siguen el modelo de tablas rígidas.

Tipos de Bases de Datos

Bases de Datos en Memoria

Son bases de datos diseñadas para almacenar los datos en la memoria RAM en lugar de en el disco duro, lo que permite acceder a los datos de manera mucho más rápida.

Ejemplos: Redis, Memcached.

Características:

Alta velocidad de lectura y escritura.

Generalmente se utilizan para almacenar datos temporales o de sesión.

Tipos de Bases de Datos

Bases de Datos Orientadas a Grafos

Estas bases de datos almacenan datos en forma de nodos y relaciones (aristas), lo que es útil para gestionar datos con relaciones complejas, como redes sociales o mapas de rutas.

Ejemplo: Neo4j.

Características:

Ideal para aplicaciones que requieren un manejo complejo de relaciones.

Los datos se almacenan como nodos conectados por aristas.

Tipos de Bases de Datos

Bases de Datos Documentales

En lugar de almacenar los datos en tablas, estos se almacenan en formato de documentos, como JSON o BSON. Son muy útiles para aplicaciones web que requieren flexibilidad en la estructura de los datos.

Ejemplo: MongoDB, CouchDB.

Características:

Flexible: los documentos pueden tener una estructura diferente.

Adecuadas para aplicaciones que requieren cambios frecuentes en la estructura de los datos.

Tipos de bases de datos I

Centralized

Biblioteca central en una Universidad.

Distributed

Hadoop, big data.

Hierarchical

Relación padre-hijo entre datos.

Object

Datos representados y almacenados como objetos.

Cloud

Entorno virtual. GCP, AWS, Azure.

Network

Nodos conectados por aristas. Grafos.

Tipos de bases de datos II

Relational

Ordenan los datos en forma de
tablas, columnas y filas.

Non-Relational

Esquema de datos no claramente
definido.



Relacionales vs. No Relacionales

Bases de Datos Relacionales (SQL)

- **Estructura:** Utilizan tablas donde los datos están organizados en filas y columnas.
- **Relaciones:** Pueden establecer relaciones complejas entre diferentes tablas.
- **Consistencia:** Son ideales para aplicaciones donde la consistencia de los datos es fundamental.
- **Lenguaje:** Utilizan SQL para interactuar con los datos.

Relacionales vs. No Relacionales

Bases de Datos No Relacionales (NoSQL)

- **Estructura:** Los datos no tienen que seguir una estructura fija, lo que las hace más flexibles.
- **Relaciones:** No son ideales para aplicaciones que requieren relaciones complejas entre datos.
- **Escalabilidad:** Mejor escalabilidad horizontal, lo que significa que se puede añadir más capacidad mediante la adición de más servidores.
- **Lenguaje:** No siguen un lenguaje de consulta estándar como SQL, suelen usar APIs o formatos como JSON.

Tipos de bases de datos III

Ventajas de una base de datos relacional

1. Los datos se estructuran fácilmente en categorías.
2. Los datos son consistentes en cuanto a su entrada y significado, y son fáciles de navegar.
3. Se pueden definir fácilmente las relaciones entre los puntos de datos.

Ventajas de una base de datos no relacional

1. Los datos no se limitan a un grupo estructurado.
2. Se pueden realizar funciones que permiten una mayor flexibilidad.
3. Sus datos y análisis pueden ser más dinámicos y permitir más variantes de entrada.

Bases Relacionales: ACID

Todas las transacciones de base de datos deben ser conformes a ACID (atómicas, coherentes, aisladas y duraderas) para garantizar la integridad de los datos.

- La **atomicidad** requiere que la transacción se ejecute correctamente como un todo o, si una parte de la transacción falla, que toda ella quede invalidada.
- La **consistencia** exige que los datos escritos en la base de datos como parte de la transacción cumplan todas las reglas definidas, así como las restricciones, incluidos los desencadenadores, las limitaciones y las cascadas.
- El **aislamiento** es fundamental para lograr el control de concurrencia y asegurarse de que cada transacción sea independiente por sí misma.
- La **durabilidad** requiere que todos los cambios realizados en la base de datos sean permanentes luego de que la transacción se haya completado de forma correcta.

Tipos de datos I

Data type	Description	Max size	Storage
char(n)	Fixed width character string	8,000 characters	Defined width
varchar(n)	Variable width character string	8,000 characters	2 bytes + number of chars
varchar(max)	Variable width character string	1,073,741,824 characters	2 bytes + number of chars
text	Variable width character string	2GB of text data	4 bytes + number of chars
nchar	Fixed width Unicode string	4,000 characters	Defined width x 2
nvarchar	Variable width Unicode string	4,000 characters	
nvarchar(max)	Variable width Unicode string	536,870,912 characters	
ntext	Variable width Unicode string	2GB of text data	
binary(n)	Fixed width binary string	8,000 bytes	
varbinary	Variable width binary string	8,000 bytes	
varbinary(max)	Variable width binary string	2GB	
image	Variable width binary string	2GB	

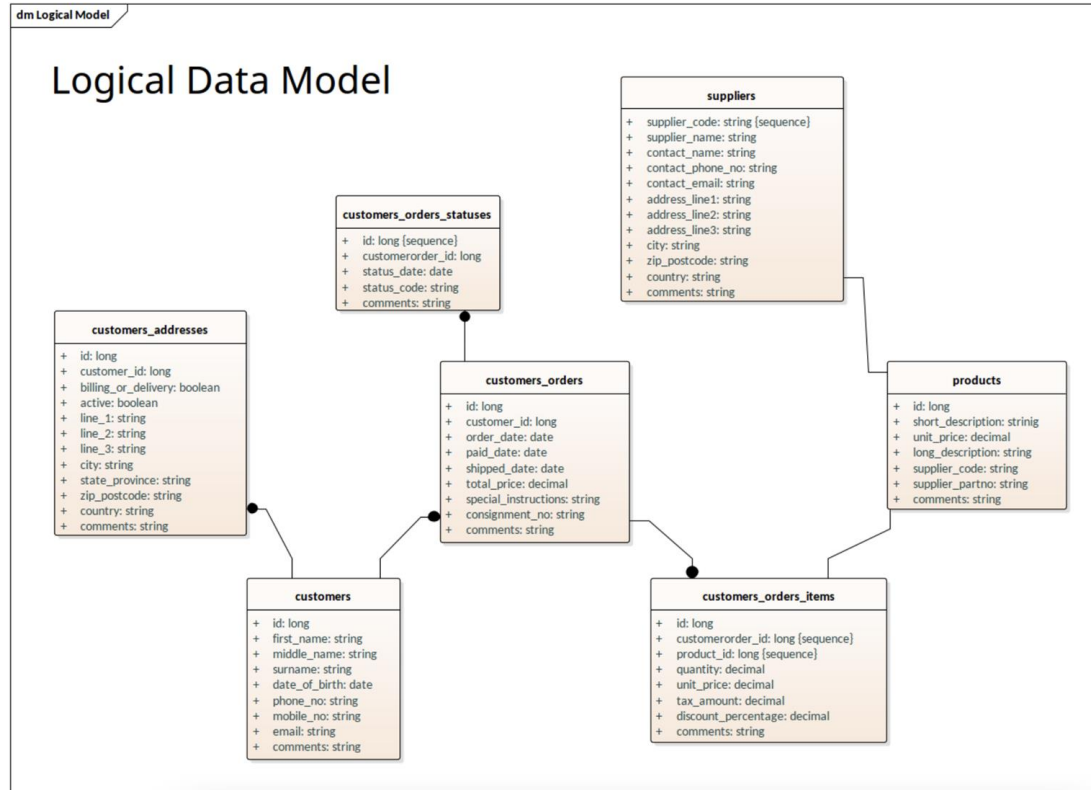
Tipos de datos II

Data type	Description	Storage
bit	Integer that can be 0, 1, or NULL	
tinyint	Allows whole numbers from 0 to 255	1 byte
smallint	Allows whole numbers between -32,768 and 32,767	2 bytes
int	Allows whole numbers between -2,147,483,648 and 2,147,483,647	4 bytes
bigint	Allows whole numbers between -9,223,372,036,854,775,808 and 9,223,372,036,854,775,807	8 bytes
decimal(p,s)	<p>Fixed precision and scale numbers.</p> <p>Allows numbers from $-10^{38} + 1$ to $10^{38} - 1$.</p> <p>The p parameter indicates the maximum total number of digits that can be stored (both to the left and to the right of the decimal point). p must be a value from 1 to 38. Default is 18.</p> <p>The s parameter indicates the maximum number of digits stored to the right of the decimal point. s must be a value from 0 to p. Default value is 0</p>	5-17 bytes
numeric(p,s)	<p>Fixed precision and scale numbers.</p> <p>Allows numbers from $-10^{38} + 1$ to $10^{38} - 1$.</p>	5-17 bytes

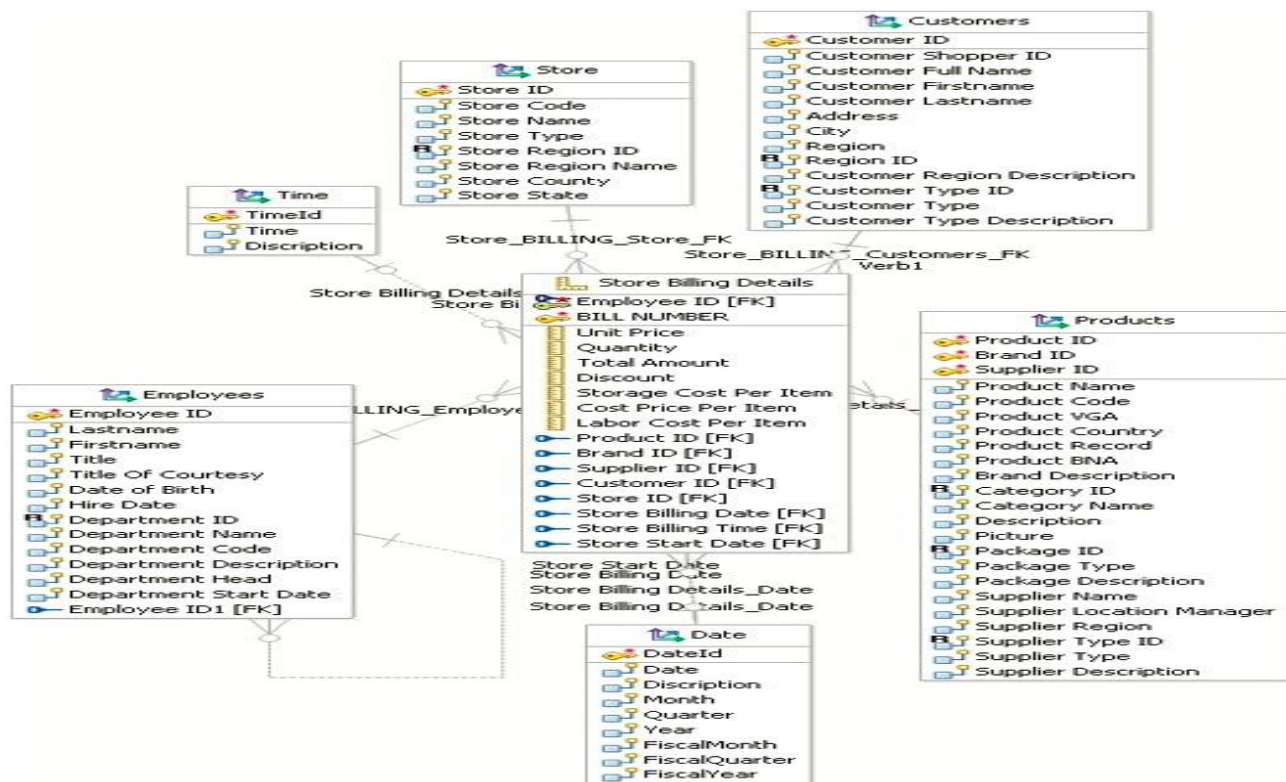
Tipos de datos III

Data type	Description	Storage
datetime	From January 1, 1753 to December 31, 9999 with an accuracy of 3.33 milliseconds	8 bytes
datetime2	From January 1, 0001 to December 31, 9999 with an accuracy of 100 nanoseconds	6-8 bytes
smalldatetime	From January 1, 1900 to June 6, 2079 with an accuracy of 1 minute	4 bytes
date	Store a date only. From January 1, 0001 to December 31, 9999	3 bytes
time	Store a time only to an accuracy of 100 nanoseconds	3-5 bytes
datetimeoffset	The same as datetime2 with the addition of a time zone offset	8-10 bytes
timestamp	Stores a unique number that gets updated every time a row gets created or modified. The timestamp value is based upon an internal clock and does not correspond to real time. Each table may have only one timestamp variable	

Diseño y creación de una base de datos relacional III



Diseño y creación de una base de datos relacional IV



Diseño y creación de una base de datos relacional V

- Sin un buen diseño de la base de datos, es probable que ésta no sea satisfactoria.
- Un buen diseño de base de datos debe implementarse de tal manera que las consultas se escriban de forma sencilla y fácil.
- Un buen diseño de base de datos no tiene redundancias de datos (la redundancia de datos se refiere a la duplicación de datos).
- Generar documentación del sistema.

Diseño y creación de una base de datos relacional VI

- **Entidad:** Cualquier cosa sobre la que se mantiene información.
- **Atributo:** Una propiedad de una entidad.
- **Instancia:** Una ocurrencia simple de una entidad.

ID	Students	Hair color	Height (in)	Weight (lbs)
DO1234	Jess	brown	61	140
DO2345	Ash	brown	52	117
DO3456	Morgan	red	55	125

Diseño y creación de una base de datos relacional VII

Relacion: Es un vínculo que nos permite definir una dependencia entre varias entidades, es decir, nos permite exigir que varias entidades compartan ciertos atributos de forma indispensable.



Diseño y creación de una base de datos relacional VII

Empleados

Nombre	DNI	Cargo
Carlos Sánchez	45338600L	001
Pepe Sánchez	02405068K	002
Juan Sánchez	40588860J	002

Cargo del empleado

ID del cargo	Descripción
001	Jefe de taller
002	Mecánico

Diseño y creación de una base de datos relacional VIII

Cardinalidad: Define el número de instancias de una entidad que se pueden relacionar con un número de instancias de otra entidad.

Modalidad: Indica si una instancia debe participar en la Relación. Es decir, si la relación es obligatoria u opcional.

Diseño y creación de una base de datos relacional IX

1. Uno a uno.
2. Uno a muchos.
3. Muchos a muchos

También hay casos donde es optativa la relación.

Diseño y creación de una base de datos relacional X

Clave primaria: valores únicos en una tabla que identifican un registro específico.

Clave foránea: registros en una tabla separada que se usan para hacer una conexión con las claves primarias.

La razón más importante para tener claves primarias y foráneas es la identificación de registros únicos en cada tabla de la base de datos.

De transaccional a analítica II

	Analytic	Transactional
Use case	Analyzing huge volumes of data for business analytics	Processing huge volumes of transactions in real-time
Optimized for	Fast inserts and selects over huge numbers of rows.	Real-time inserts, updates, selects, and deletes over fewer rows.
Example SELECT query	SELECT GENDER, COUNT(*) FROM CUSTOMERS GROUP BY GENDER;	SELECT GENDER FROM CUSTOMERS WHERE ID = 1742;
Query response times	Seconds for an analytical query	Milliseconds for a transactional query
Example databases	Vertica, Redshift, Greenplum, Teradata, ParAccel	MySQL, PostgreSQL, Microsoft SQL Server

02

SQL



LEARN SQL

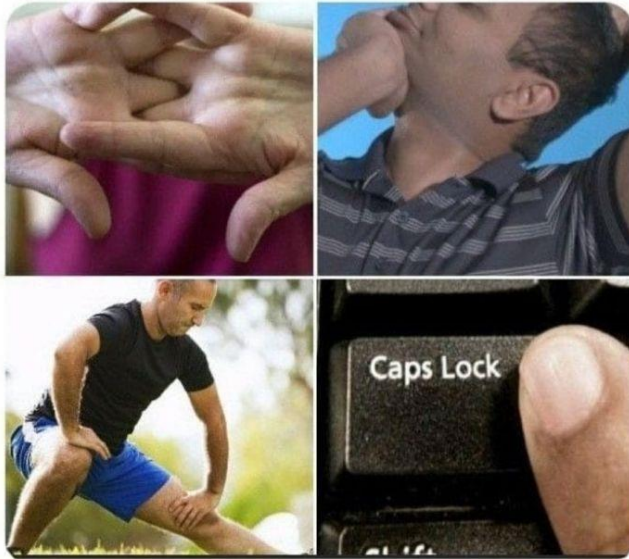
WE MUST

¿Qué es SQL?

SQL (Structured Query Language) es un lenguaje de programación utilizado para gestionar y manipular bases de datos relacionales. Se encarga de realizar operaciones como consultas, inserciones, actualizaciones y eliminación de datos, así como la definición y control de estructuras de datos.

¿Qué es SQL?

SQL programmers be like



¿Qué es SQL?

Características clave:

- **Declarativo:** Dices qué quieres obtener, no cómo obtenerlo.
- **Estándar:** Aunque diferentes bases de datos tienen sus propias extensiones, SQL es un estándar internacional.
- **Popularidad:** Amplio uso en empresas de todas las industrias.

¿Qué es SQL?

- SQL es un acrónimo en inglés para **Structured Query Language**.
- Es un **Lenguaje de Consulta Estructurado**.
- Es un tipo de lenguaje de programación que te permite manipular y consultar datos de una base de datos.
- Tiene capacidad de hacer cálculos avanzados y álgebra.
- Es utilizado en la mayoría de empresas que almacenan datos en una base de datos.
- Ha sido y sigue siendo el lenguaje de programación más usado para bases de datos relacionales.

¿Cómo surge SQL?

- La primera publicación basada en la manipulación de los datos propuesta por Codd (creador del modelo relacional), fue desarrollada por un grupo de investigación de IBM, Chamberlain y Boyce en 1974.
- Dio origen a lo que se convirtió en SQL, llamado originalmente SEQUEL (Structured English QUERY Language)
- Proveedores de diferentes DBMS desarrollaron variantes de aquel lenguaje, hasta que finalmente ANSI e ISO generaron la versión estándar en 1986, llamada SQL-1 o SQL-86.

¿Para qué sirve SQL?

- Hacer consultas y mantenimiento.
- Recuperar datos.
- Insertar, actualizar y eliminar registros.
- Crear nuevas bases de datos.
- Crear nuevas tablas
- Crear procedimientos
- Crear vistas
- Establecer permisos en tablas, procedimientos y vistas.

Tipos de comandos SQL

Los comandos de SQL son instrucciones que se usan para comunicarse con la base de datos.

- Data Definition Language (DDL).
- Data Manipulation Language (DML).
- Data Control Language (DCL).
- Transaction Control Language (TCL).
- Data Query Language (DQL).

Data Definition Language I

- Los comandos DDL cambian la estructura de la tabla tales como la creación de una tabla, la eliminación de una tabla, la modificación de una tabla, etc.
- Todos los comandos de DDL son auto-comprometidos, lo que significa que guardan permanentemente todos los cambios en la base de datos.
- Los comandos más usados son:
 - CREATE
 - ALTER
 - DROP
 - TRUNCATE

Data Definition Language II

```
CREATE TABLE EMPLOYEE(Name VARCHAR2(20), Email VARCHAR2(100), DOB DATE);
```

```
DROP TABLE EMPLOYEE;
```

```
ALTER TABLE STU_DETAILS ADD(ADDRESS VARCHAR2(20));
```

```
ALTER TABLE STU_DETAILS MODIFY (NAME VARCHAR2(20));
```

```
TRUNCATE TABLE EMPLOYEE;
```


Data Manipulation Language I

- Los comandos **DML** se utilizan para modificar la base de datos. Es responsable de todo tipo de cambios en la base de datos.
- El comando de **DML** no es auto-comprometido, lo que significa que no puede guardar permanentemente todos los cambios en la base de datos. Se pueden revertir.
- Los comandos más usados son:
 - **INSERT**
 - **UPDATE**
 - **DELETE**

Data Manipulation Language II

```
INSERT INTO javatpoint (Author, Subject) VALUES ("Sonoo", "DBMS");
```

```
UPDATE students SET User_Name = 'Sonoo' WHERE Student_Id = '3'
```

```
DELETE FROM javatpoint WHERE Author="Sonoo";
```

Data Control Language I

- Los comandos DCL se utilizan para conceder y retirar la autoridad a cualquier usuario de la base de datos.
- Los comandos más usados son:
 - **Grant:** usado para darle a los usuarios permisos para la base de datos.
 - `GRANT SELECT, UPDATE ON MY_TABLE TO SOME_USER, ANOTHER_USER;`
 - **Revoke:** usado para quitar permisos a usuarios.
 - `REVOKE SELECT, UPDATE ON MY_TABLE FROM USER1, USER2;`

Transaction Control Language I

- Los comandos TCL sólo pueden utilizarse con comandos DML como INSERT, DELETE y UPDATE.
- Estas operaciones se consignan automáticamente en la base de datos, por lo que no pueden utilizarse al crear o eliminar tablas.
- Los comandos más usados son:
 - **COMMIT**: usado para guardar la operación en la db.
 - **ROLLBACK**: usado para cancelar la operación que no fue guardada en la db.
 - **SAVEPOINT**: usado para volver atrás con alguna operación sin volver del todo atrás. Retornar a un *checkpoint*.

Transaction Control Language II

```
DELETE FROM CUSTOMERS
```

```
WHERE AGE = 25;
```

```
COMMIT;
```

```
DELETE FROM CUSTOMERS
```

```
WHERE AGE = 25;
```

```
ROLLBACK;
```


```
SAVEPOINT SAVEPOINT_NAME;
```

Data Query Language I

- DQL se utiliza para obtener los datos de la base de datos.
- Utiliza solo un tipo de comando:
 - **SELECT:** utilizado para obtener, seleccionar, los datos que quieras en el formato que desees y con las condiciones que precises.
 - `SELECT emp_name FROM employee WHERE age > 20;`

Data Query Language II

```
1 Select * from Apps
```

	Query Favorites	Query History		
AppID	AppName	CreatorName	AppCategory	AppPrice
1	AppDividend	Krunal	Software	50
2	Escrow	LVVM	Fashion	60
3	KGB	MJ	Music	70
4	Moscow	Mayor	Area	80
5	MoneyControl	Mukesh	Investment	90
6	Investing	Bill	Stocks	100

Data Query Language III

```
WITH tmp_1 AS
(
    SELECT Calc1 =
    (
        (SELECT TOP 1 DataValue
        FROM (
            SELECT TOP 50 PERCENT DataValue
            FROM SOMEDATA
            WHERE DataValue IS NOT NULL
            ORDER BY DataValue
        ) AS A
        ORDER BY DataValue DESC
        ) +
        (SELECT TOP 1 DataValue
        FROM (
            SELECT TOP 50 PERCENT DataValue
            FROM SOMEDATA
            WHERE DataValue IS NOT NULL
            ORDER BY DataValue DESC
        ) AS A
        ORDER BY DataValue ASC)
    )/2
),
tmp_2 AS
(
    SELECT AVG(DataValue) Mean, MAX(DataValue) - MIN(DataValue) AS MaxMinRange
    FROM SOMEDATA
),
tmp_3 AS
(
    SELECT TOP 1 DataValue AS Calc2, COUNT(*) AS Calc2Count
    FROM SOMEDATA
    GROUP BY DataValue
    ORDER BY Calc2Count DESC
)
SELECT Mean, Calc1, Calc2, MaxMinRange AS [Range]
FROM tmp_1 CROSS JOIN tmp_2 CROSS JOIN tmp_3;
```


Anatomía de un comando SELECT - Cláusulas I

- La sentencia **SELECT** se compone de varias cláusulas. Éstas son:
 - **SELECT**
 - **FROM**
 - **WHERE**
 - **GROUP BY**
 - **ORDER BY**
 - **HAVING**

Anatomía de un comando SELECT - Cláusulas II

- Cada una de estas cláusulas tiene información que le sigue. Esto es:
 - **SELECT** column(s)
 - **FROM** table / view
 - **WHERE** condition / expression
 - **GROUP BY** column(s)
 - **ORDER BY** column(s)
 - **HAVING** aggregate function and condition / expression
- Las dos únicas cláusulas necesarias en una sentencia SELECT son **SELECT** y **FROM**.

Anatomía de un comando SELECT - Cláusulas III

SELECT lastname, firstname

FROM customer

LastName	FirstName
Adams	Susan
Johnson	JoAnne
Smith	Ron
Baker	Pete
Smith	Bill

Anatomía de un comando SELECT - Cláusulas IV

SELECT lastname, firstname

FROM customer

WHERE lastname = 'Smith'

CustomerID	LastName	FirstName	Address	City	State	Zip
1002	Smith	Ron	2201 Thomas Dr	Panama City	FL	32401
1004	Smith	Bill	4407 12th Avenue	Austin	TX	78746

Anatomía de un comando SELECT - Cláusulas V

SELECT State, count(state) as 'Count of Customers by State'

FROM customer

GROUP BY state

State	Count of Customers by State
FL	1
NC	1
OK	1
TX	2

Anatomía de un comando SELECT - Cláusulas VI

SELECT State, count(state) as 'Count of Customers Greater Than 1'

FROM customer

GROUP BY state

HAVING count(state) > 1

State	Count of Customers Greater Than 1
TX	2

Anatomía de un comando SELECT - Cláusulas VII

SELECT lastname, firstname

FROM customer

ORDER BY lastname

LastName	FirstName
Adams	Susan
Baker	Pete
Johnson	JoAnne
Smith	Ron
Smith	Bill

Anatomía de un comando SELECT - Cláusulas VII

SELECT lastname, firstname

FROM customer

ORDER BY lastname, firstname

LastName	FirstName
Adams	Susan
Baker	Pete
Johnson	JoAnne
Smith	Bill
Smith	Ron

Anatomía de un comando SELECT - Operadores

Operador	Significado
=	Igual a
>	Mayor que
>=	Mayor o igual que
<	Menor que
<=	Menor o igual que
<>	No es igual que
BETWEEN " AND "	Entre dos valores (Incluidos los extremos)
IN (Valores)	incluidos en una lista de valores
LIKE	busca patron
IS NULL	Valida si es nulo
IS NOT NULL	Valida si no es nulo

Anatomía de un comando SELECT - Operadores II

SQLQuery1.sql - D:\...EGO-PC\diego (60))

```
values ('Martin Fierro','Jose Hernandez','Emece',16.00);
insert into libros (titulo,autor,editorial,precio)
values ('Aprenda PHP','Mario Molina','Emece',35.40);
insert into libros (titulo,autor,editorial,precio)
values ('Cervantes y el quijote','Borges','Paidós',50.90);

-- Seleccionamos los registros cuyo autor sea diferente de 'Borges'
select * from libros
where autor<>'Borges';

-- Seleccionamos los registros cuyo precio supere los 20 pesos, sólo el título y precio
select titulo,precio
from libros
where precio>20;

-- Recuperamos aquellos libros cuyo precio es menor o igual a 30
select *from libros
where precio<=30;
```

100 %

Results Messages

	titulo	autor	editorial	precio
1	Martin Fierro	Jose Hernandez	Emece	16
2	Aprenda PHP	Mario Molina	Emece	35.4

	titulo	precio
1	El aleph	24.5
2	Aprenda PHP	35.4
3	Cervantes y el quijote	50.9

	titulo	autor	editorial	precio
1	El aleph	Borges	Emece	24.5
2	Martin Fierro	Jose Hernandez	Emece	16

Query executed successfully: DIEGO-PC (14.0 RTM)

SQLQuery2.sql - NC8430\... (Francisco...2))

```
select *
from Usuarios
where Usuarios.Edad between '20' and '25'
```

Resultados Mensajes

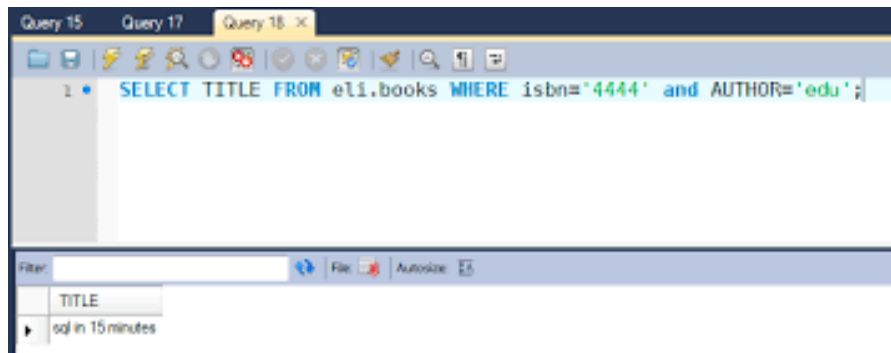
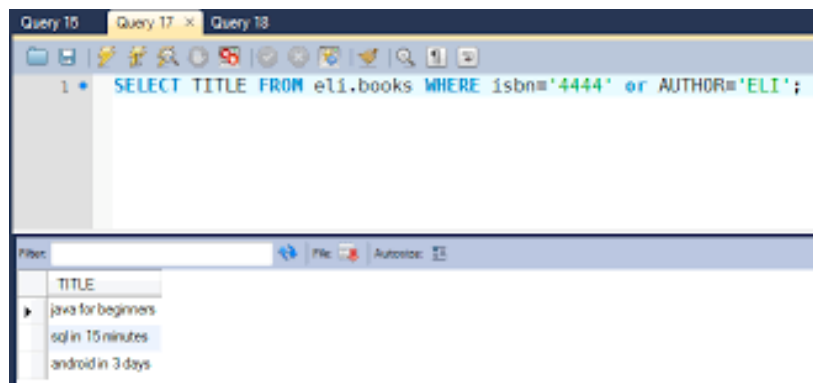
	UsuarioID	NombreUsuario	ApellidoP	ApellidoM	Edad	Sexo
1	1	Jose Francisco	Olivares	Martinez	22	M
2	2	Jorge	Castañeda	Morales	20	M
3	3	German	Lopez	Laguna	25	M
4	4	Rocio	Morales	Magallon	21	F
5	6	Maria Dolores	Gomez	Garcia	25	F
6	7	Soledad	Perez	Hurtado	23	F
7	11	Maria de Lourdes	Jimenez	Franco	23	F

Con... NC8430\SQLEXPRESS (10.50 RTM) nc8430\Francisco (52) MiPrimeraBD 00:00:00 7 filas

Anatomía de un comando SELECT - Condiciones lógicas I

Operadores Lógicos	
Operador	Uso
AND	Es el "y" lógico. Evalúa dos condiciones y devuelve un valor de verdad sólo si ambas son ciertas.
OR	Es el "o" lógico. Evalúa dos condiciones y devuelve un valor de verdad si alguna de las dos es cierta.
NOT	Negación lógica. Devuelve el valor contrario de la expresión.

Anatomía de un comando SELECT - Condiciones lógicas II



Anatomía de un comando SELECT - Funciones de agregación I

Las funciones agregadas proporcionan a SQL utilidades de cálculo sobre los datos de las tablas. Y se usan dentro de una cláusula **SELECT** en grupos de registros para devolver un único valor que se aplica a un grupo de registros.

Funciones de Agregado	
Función	Descripción
AVG	Utilizada para calcular el promedio de los valores de un campo determinado
COUNT	Utilizada para devolver el número de registros de la selección
SUM	Utilizada para devolver la suma de todos los valores de un campo determinado
MAX	Utilizada para devolver el valor más alto de un campo especificado
MIN	Utilizada para devolver el valor más bajo de un campo especificado

Anatomía de un comando SELECT - Funciones de agregación II

```
318 -- Agrupando y aplicando funciones de agregación
319 SELECT
320     id_marca,
321     count(*) AS "núm. productos contando nulos",
322     count(id_marca) AS "núm. productos"
323 FROM
324     productos
325 GROUP BY id_marca;
```

	id_marca	núm. productos contando nulos	núm. productos
1	NULL	1	0
2	2	1	1
3	5	28	28
4	6	143	143
5	8	252	252
6	11	36	36
7	14	7	7
8	15	106	106

Execution finished without errors.
Result: 250 rows returned in 39ms
At line 319:
SELECT

Anatomía de un comando SELECT - Orden de evaluación de cláusulas

Dada una sentencia SQL de selección que incluye todas las posibles cláusulas, el orden de ejecución de las mismas es el siguiente:

1. Cláusula **FROM**: obtener los registros de todas las tablas fuentes.
2. Cláusula **WHERE**: filtrar las combinaciones que cumplen las condiciones.
3. Cláusula **GROUP BY**: construir los grupos basados en las expresiones.
4. Cláusula **HAVING**: filtrar los grupos que cumplen las condiciones.
5. Cláusula **ORDER BY**: ordenar las filas
6. Cláusula **SELECT**: evaluar las expresiones de las columnas seleccionadas.



```
mysql> exit  
Bye
```