Bases de Datos y SQL

1. **Tareas del Analista de Datos**
   1. Establece metas en equipos de tecnología
   2. Minería de datos de fuetes primarias y secundarias
   3. Limpieza y disección de datos
   4. Análisis e interpretación de resultados
   5. Tendencias y patrones de un conjunto de datos
   6. Nuevas oportunidades para mejorar procesos
   7. Informes de datos
   8. Diseña, crear y mantener DB.
   9. Solucionar problemas de código
2. **Tipos de análisis de datos: descriptivo, diagnostico, predictivo.**

* **Análisis Descriptivo**: responde, *¿qué paso?,* describe los patrones clave en los datos existentes y permite observar situaciones y comportamientos habituales. Son reportes.
* **Análisis Diagnostico**: responde, *¿por qué paso?,* define conclusiones basadas en los datos históricos identificados en el análisis descriptivo.
* **Análisis predictivo**: responde, *¿qué pasará?,* identifican los comportamientos futuros en base a la información histórica.
* **Análisis prescriptivo**: responde, *¿qué debería hacer?,* con simulaciones y optimizaciones se compara la información resultante del análisis predictivo, para implementar una acción.

1. **Partes de una tabla**

FILAS

FILAS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N. Cliente | Nombre | Sexo | Telefono | ID\_EDAD |
| 398 | Cristian | M | 5512853695 | 1 |
| 562 | Fernando | M | 5587459562 | 3 |
| 817 | Silvana | F | 5574851235 | 5 |
| 345 | Juan | M | 5512354896 | 8 |
| 390 | Mariela | F | 5585996652 | 9 |

COLOUMNAS

COLOUMNAS

COLOUMNAS

**Ejemplo de resumen de tabla**:

***COLUMNAS:*** 6

***GRADO:*** 6

Las **COLUMNAS** son: *n. de cliente*, nombre, sexo, teléfono, id\_edad.

Se tienen 5 registros que describen a 5 usuarios distintos

El ***dominio*** de la columna “N. Clientes" es: 5

1. **Ciclo de vida de los datos**

**Ciclo de vida de los datos**: Recolección, Mantenimiento, Síntesis, Uso, Publicación y Depuración.

* **Recolección**: Identificación del origen de los datos. Almacenamiento de forma digital.
* **Mantenimiento:** revisión de la información. Se procesa para garantizar su calidad.
* **Síntesis**: se modela la información de acuerdo a los indicadores requeridos.
* **Uso**: su disponibilidad para su explotación, en la generación de reportes, informes, modelos, etc.
* **Publicación**: el análisis realizado es expuesto, para la toma de decisiones.
* **Depuración**: gracias a la exposición, se detecta si la información requiere agregaciones, eliminaciones o cambio.

1. **Conceptos básicos**

**Dato**: (1) Característica o atributo sin procesamiento, el cual *no informa nada por sí solo*.  
(2) Se considera *la unidad mínima de información*, simplemente es un ente que requiere de una interpretación para dar lugar a la información.

**Información**: (1) Unión de datos procesados, que se complementan para informar un hecho. (2) Conjunto de datos con que a través del procesamiento adecuado adquiere un significado en un contexto determinado.

**Base de Datos**: Conjunto de datos con alguna característica en común, contienes elementos que los relaciona, almacenados de forma tradicional o digital.

**No Relacionales**: *Cassandra,* *hadoop*, *mongoDB*. No tiene estructura.

**ETL** = *Extract*, *Transform*, *Load;* son sistemas que permite el procesamiento de los datos, y convertirlos en información.

**Lenguajes de programación:** *r*, *Python*, *Julia*

**Formatos**: *xls*, *csv*.

**Analista de Datos**: busca ***determinar cómo se pueden usar los datos*** para **responder preguntas** y **resolver problemas**, estudia el presente para **identificar tendencias** y **hacer predicciones**. (2) Proceso de examinar conjunto de datos para su transformación y visualización, sacan información para gestionar indicadores.

**Data Analytics**: son métodos y técnicas de medición, que permiten gestionar la información en la etapa de: **recolección**, **transformación** y **visualización**. Grupo de tareas para identificar las necesidades del negocio con el fin de propones una solución.

**Bi**: business intelligence, es la unión de técnicas de manipulación de datos y tecnologías de integración y almacenamiento, para gestionar información.

**Base de Datos relacional**: se basan en el *modelo relacional,* propuesto por **Edgar F.Codd**, en los 70’s , contiene *tablas, registros, campos, dominio, cardinalidad, grado.*

Relacionales: My Workbench, My SQL Server. Tiene una estructura

**Tabla:** o **relación** es una *entidad con atributos*, *campos o columnas y tuplas*, *registros o reglones*. Representa un número indefinido de elementos descritos con características en común, cada uno en un diferente renglón.

**Campos, columnas o atributos**: representan una característica particular de cada entidad, los campos son comunes a todos los registros de una tabla.

**Tuplas, registros o reglones**: se refieren a la descripción de cada ente de la tabla. Son instancias particulares.

**Dominio**: se refiere al número de valores que puede obtener una columna, ejemplo, m y f son la columna género, entonces su dominio sería 2.

**Cardinalidad**: es el número de registros que tiene una tabla.

**Grado**: es el número de campos que tiene una tabla.

1. **Tipo de modelos**

**Relacional**: Colección de tablas para representar tanto los datos como sus relaciones.

VENTAS

Folio

N. de cliente

Clave

Cantidad

Subtotal

Iva

Total

CLIENTES

N. Cliente

Nombre

Sexo

Teléfono

PRODUCTO

Clave

Descripción

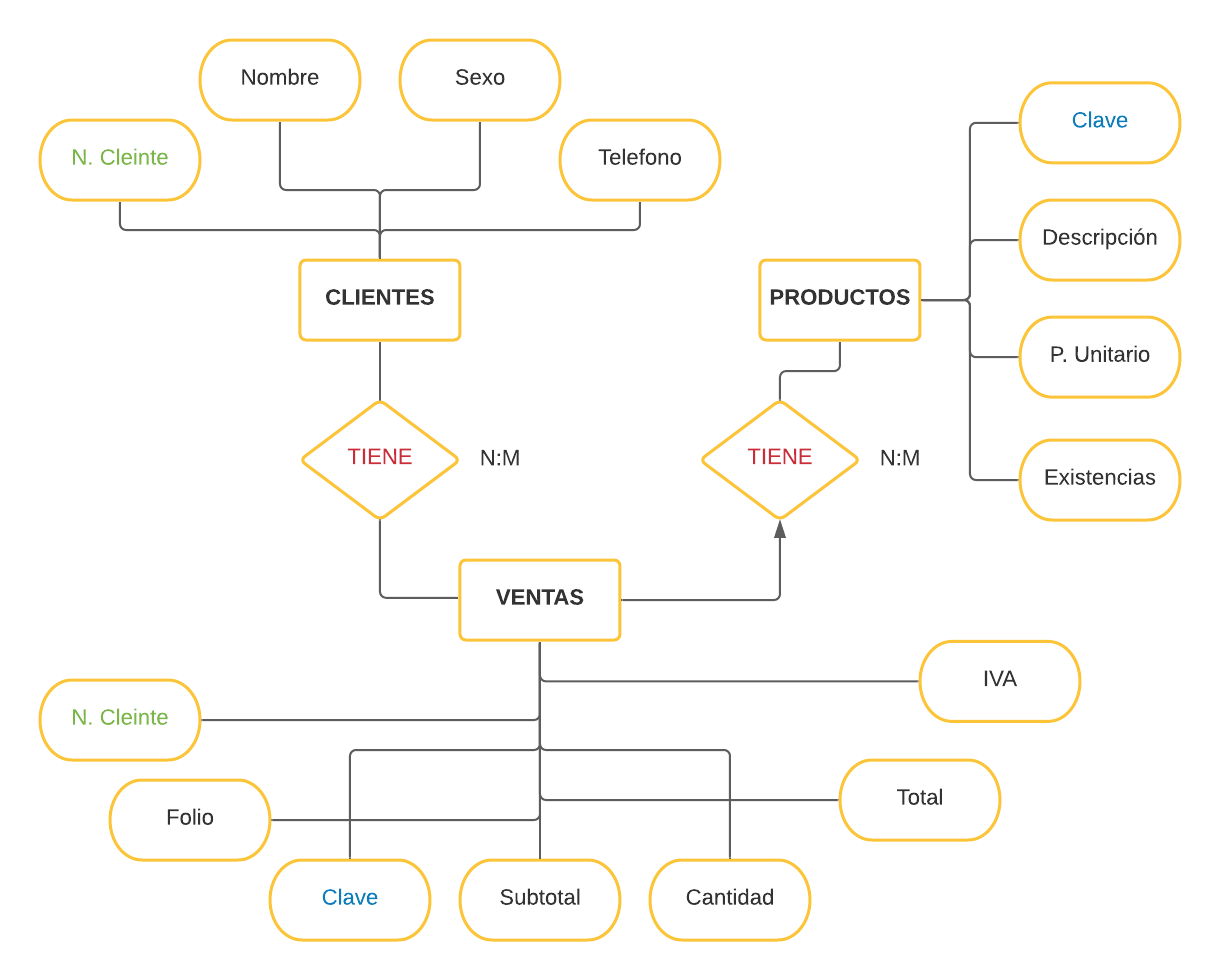
P. Unitario

Existencia

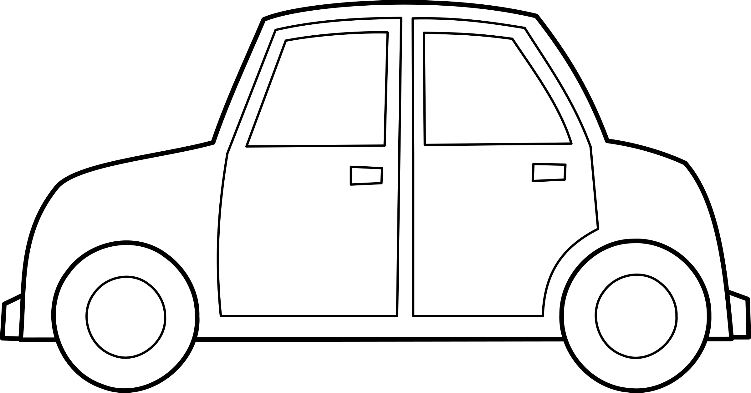
Características Modelo relacional: estructura de tablas, que se relaciona a su vez con otras tablas, NO POSEE INFORMACIÓN REPETIDA de forma innecesaria, lo que permite adicionar más información sin llegar a afectar la otra almacenada.

Ventajas

* Favorece el proceso de normalización, el cual permite eliminar la redundancia de los datos.
* Permite realizar consultas y obtener reporte de forma ágil y rápida por medio de SQL u otro lenguaje de base de datos estructurado.
* Se puede crear una o varias relaciones
* Ayuda evitar la duplicidad de los registros
* Garantiza la integridad referencial, si un registro tiene una relación con otra tabla, no permite que el mismo sea eliminado. Si se quiere borrar, también borrara todos los datos relacionados.

**Entidad-Relación**: percepción del mundo real que consiste en una colección de objetos básicos, denominados entidades, y de las relaciones entre ellos

|  |  |
| --- | --- |
| Componentes |  |
|  | Rectángulos: representan conjuntos de entidades. |
|  | Elipses: representan atributos, en la mayoría de los casos la clave candidata. |
|  | Rombos: representan acciones que permiten relaciones entre tablas. |
|  | Líneas: representan la unión entre acciones y entidades |

**Orientado a objetos**: La información se representa mediante objetos como los presentes en la programación orientada a objetos.

Objeto: Carro

**Encapsulados**

Neumáticos

Chasis

Motor

**Funciones**

Encender

Acelerar

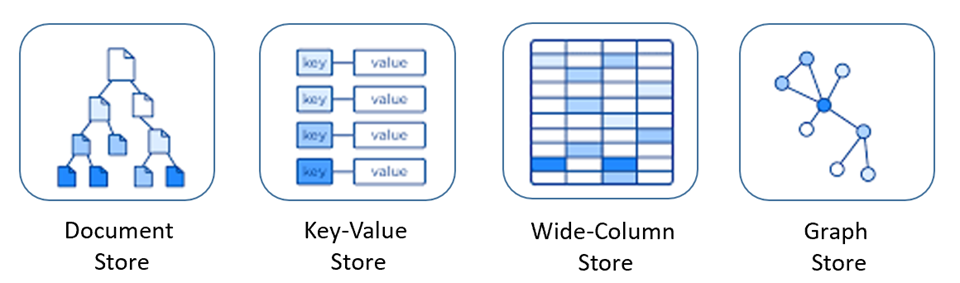
Apagar

**Atributos**

Color: Blanco

Año: 2000

Trans: Manual

**No Relacional**: No tiene un identificado que sirva de relación entre un conjunto de datos y otros.

**Jerárquico**: los datos son organizados en una estructura parecida a un árbol.

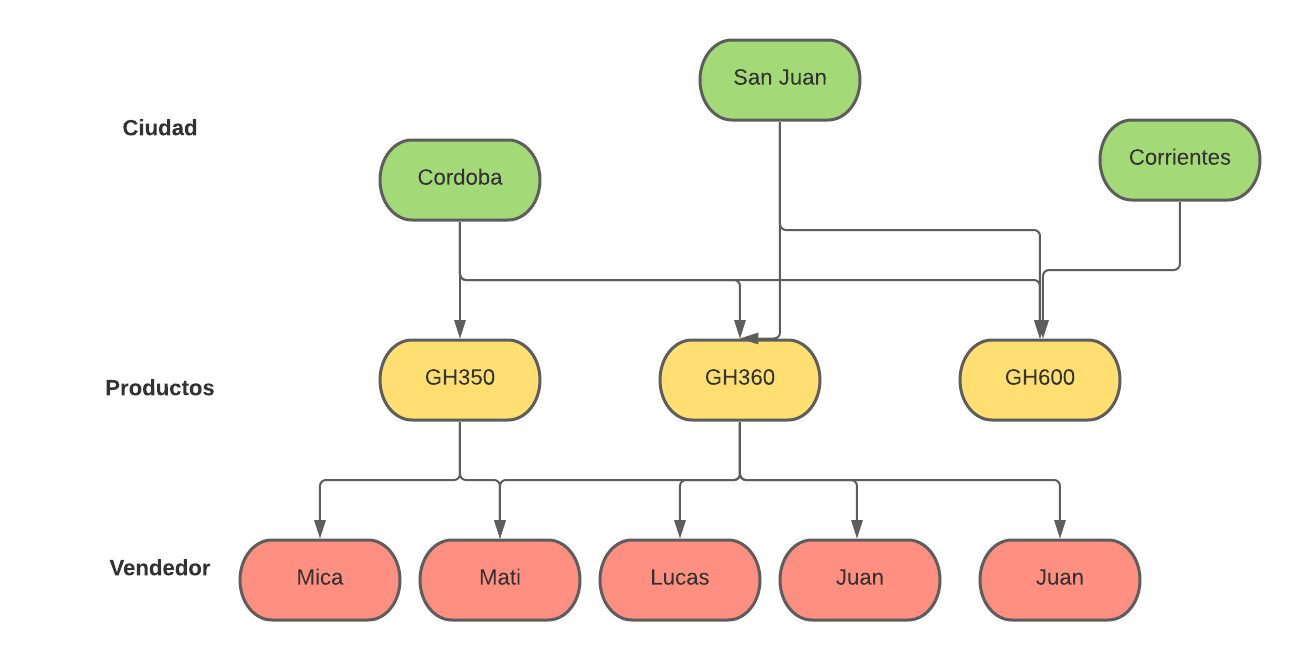
Salida

Destino

Vuelo

Pasajero

**Red**: set de registros, los cuales están conectados entre sí por medio de una red



1. **Tipos de relaciones**

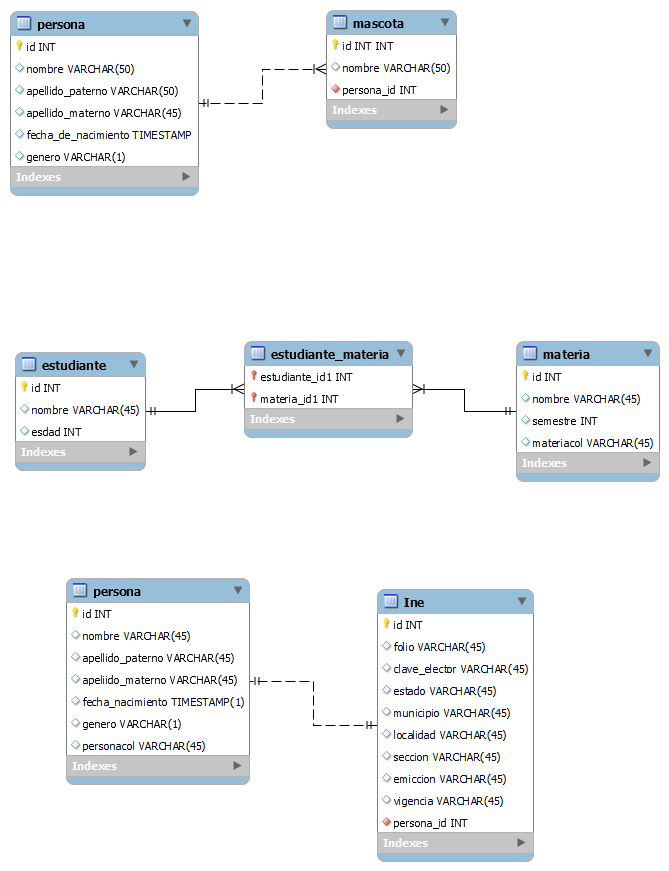
**Uno a muchos**: un registro en una tabla puede tener relación con varios elementos de otra tabla

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| N. Cliente | Folio | Clave | Cantidad | Total | Subtotal | Iva |
| 398 | 3 | 9 | 27 | 4367 | 369 | 19% |
| 562 | 35 | 12 | 68 | 915 | 536 | 6% |
| 817 | 185 | 5 | 17 | 5749 | 927 | 20% |
| 345 | 15 | 82 | 22 | 24 | 3 | 2% |
| 390 | 39 | 11 | 18 | 512 | 74 | 10% |
| 398 | 542 | 36 | 65 | 9472 | 327 | 21% |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N. Cliente | Nombre | Sexo | Telefono | ID\_EDAD |
| 398 | Cristian | M | 5512853695 | 1 |
| 562 | Fernando | M | 5587459562 | 3 |
| 817 | Silvana | F | 5574851235 | 5 |
| 345 | Juan | M | 5512354896 | 8 |
| 390 | Mariela | F | 5585996652 | 9 |

N

1



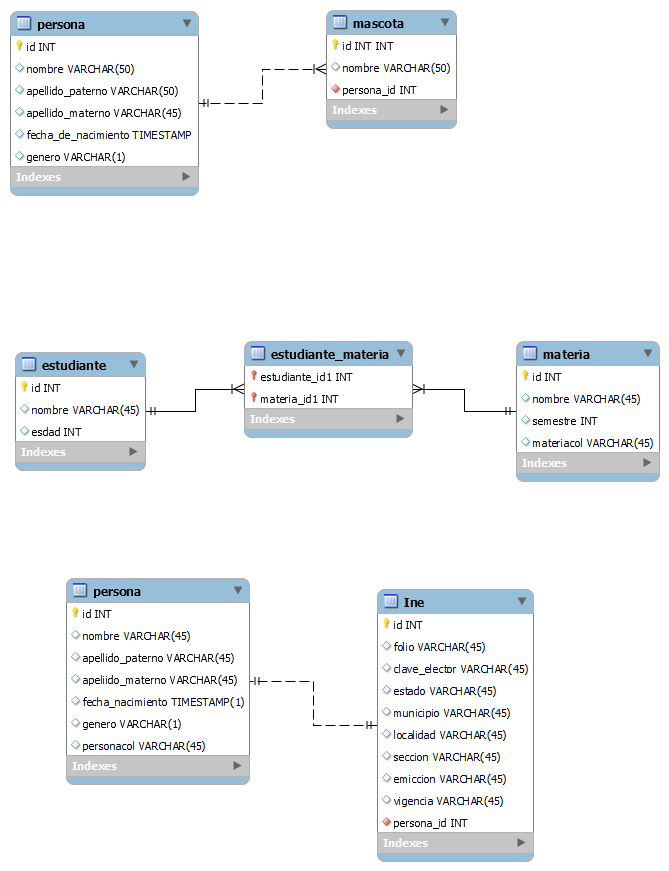
**Muchos a muchos**: permite unir tablas que en un principio no tienen relación directa. (1) Cuando uno o más registros en una tabla puede tener una relación con uno o más elementos de otra tabla.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| N. Cliente | Folio | Clave | Cantidad | Total | Subtotal | Iva |
| 398 | 3 | 9 | 27 | 4367 | 369 | 19% |
| 562 | 35 | 12 | 68 | 915 | 536 | 6% |
| 817 | 185 | 5 | 17 | 5749 | 927 | 20% |
| 345 | 15 | 82 | 22 | 24 | 3 | 2% |
| 390 | 39 | 11 | 18 | 512 | 74 | 10% |
| 398 | 542 | 36 | 65 | 9472 | 327 | 21% |

|  |  |  |  |
| --- | --- | --- | --- |
| Clave | Descripción | P. Unitario | Existencia |
| 9 | Escritorios | 35 | 4 |
| 12 | Silla | 43 | 9 |
| 5 | Ipad Mouse | 75 | 1 |
| 82 | Microfonos | 95 | 2 |
| 11 | Parlantes | 23 | 1 |
| 11 | Parlantes | 52 | 8 |

M

N



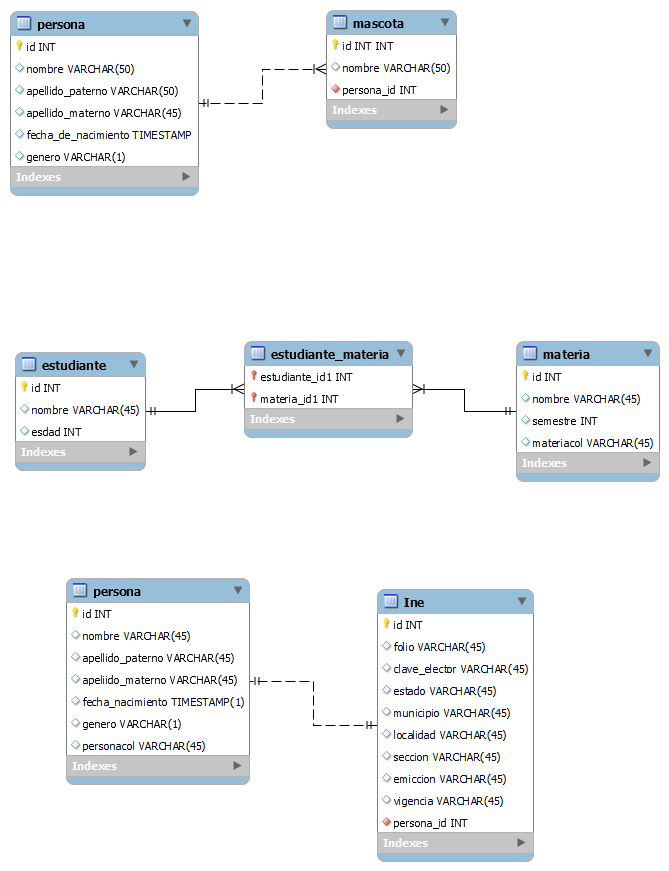
**Uno a Uno**: es la relación que menos se utiliza, pues quiere decir que ninguna tabla puede repetir registros de la otra. De esta firma, ambas tablas podrían combinarte en una sola, son para fines específicos. (1) Cada registre en cada tabla solo aparece una vez.

1

1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N. Cliente | Nombre | Sexo | Telefono | ID\_EDAD |
| 398 | Cristian | M | 5512853695 | 1 |
| 562 | Fernando | M | 5587459562 | 3 |
| 817 | Silvana | F | 5574851235 | 5 |
| 345 | Juan | M | 5512354896 | 8 |
| 390 | Mariela | F | 5585996652 | 9 |

|  |  |
| --- | --- |
| DID\_EDAD | EDAD |
| 1 | 25 |
| 3 | 64 |
| 5 | 43 |
| 8 | 30 |
| 9 | 14 |



1. **Conexión a una Base de Datos**

Para almacenas una base de datos con muchos registros se necesita alojar en un servidor.

A los servidores que almacenan bases de datos se llama Servidores de Bases de Datos.

Para conectarnos a un servidor se necita:

* Dirección del servidor o host.
* Usuario con el que se conectara a la BD
* Contraseña de acceso
* Puerto a través del cual realizará la conexión

Ejemplo de cliente: Workbench es un cliente de MySQL

Workbench permite conectarse con servidores bases de datos: para crear nuevas BD, hacer consultas crear tablas, entre muchas otras tareas.

**Servidor**: equipo de cómputo potente que forma parte de una red de computadoras y provee a los clientes.

**Clientes:** equipo a los que el servidor provee de distintos servicios

1. **Básicos de SQL**

Las bases de datos relacionales tienen su propio lenguaje con este podemos hacer que se realizasen algunas acciones, este lenguaje es el SQL, es un estándar para BD relaciónales. Cada SGBD tiene su propia implantación de SQL.

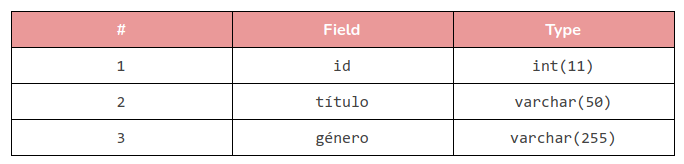
**Definición de la estructura de una Tabla**

SHOW TABLES;

DESCRIBE <nombre\_tabla>; para conocer los campos de una tabla, y tipos de datos a almacenar.

Ejemplo:

DESCRIBE movies;



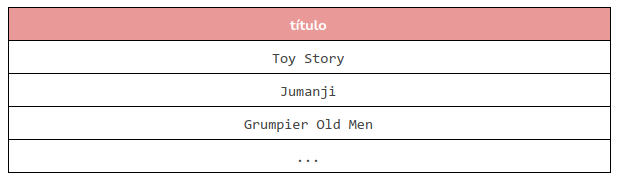
**Estructura básica de una consulta**

Ejemplo de una consulta

Los resultados de una consulta siempre se devuelven en tablas.

SELECT titulo

FROM movies;

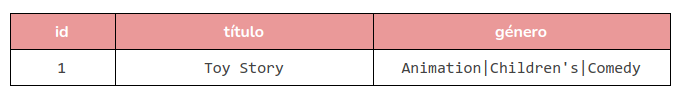


**Filtrado de información con WHERE**

Se puede utilizar operadores relacionales, lógicos y algunos otros para realizar combinaciones de filtros.

Ejemplo

SELECT \*  
FROM movies  
WHERE titulo = ‘Toy Story’;

Resultado

**Tabla de operadores relacionales y lógicos**

|  |  |  |
| --- | --- | --- |
| Operador | Nombre | Ejemplo |
| = | Igual | SELECT \* FROM persona WHERE edad =26; |
| <>  ¡= | Distinto, ambos operadores son validos | SELECT \* FROM persona WHERE edad ¡=26; |
| < | Menor que | SELECT \* FROM persona WHERE edad <26; |
| <= | Menor o igual que | SELECT \* FROM persona WHERE edad <=26; |
| > | Mayor que | SELECT \* FROM persona WHERE edad >26; |
| >= | Mayor o igual que | SELECT \* FROM persona WHERE edad >=26; |
| AND | Conjunción lógica | SELECT \* FROM persona WHERE edad =26  ANDESTATURA =1.75; |
| OR | Disyunción lógica | SELECT \* FROM persona WHERE edad =26  ORESTATURA =1.75; |
| NOT | Negación lógica | SELECT \* FROM persona WHERE NOT edad =26; |
| IN | Permite buscar si un campo se encuentra en un campo  de valores. La comparación se realiza mediante el operador OR | SELECT \* FROM persona WHERE edad IN (18,28); |

**Ordenamientos y límites**

**ORDER BY:** indica a la consulta la forma en que debe mostrar los datos ordenando los mismos por un campo de forma ascendente (**ACD**) o descendente (**DESC**), por defecto los resultados siempre son ordenados de forma ascendente.  
**LIMIT**: indica el número de registros máximos y limitando los registros a “n”.

Ejemplo: *Ordenamiento descendente, limitado a 3 filas*.

SELECT \*   
FROM persona  
WHERE estatura = 1.75  
ORDER BY edad DESC  
LIMIT 3;

Ejemplo: *Ordenamiento ascendente, limitado a 3 filas, dos formas de expresarlo*

|  |  |
| --- | --- |
| SELECT \*  FROM persona WHERE estatura = 1.75;  ORDER BY edad ASC LIMIT 3; | **SELECT \*  FROM persona WHERE estatura = 1.75;**  **ORDER BY edad  LIMIT 3;** |

**Búsqueda de patrones**: técnica ampliamente utilizada por distintos leguajes de programación y SGBD, consiste en dado un patrón representado mediante cadenas y encontrar un registro dentro de la BD en dónde el campo coincida con el patrón buscado.

Cuando una cadena caza con el parón a buscar, se dice que: “*caza con el patrón buscado*”

En MySQL se hace la búsqueda mediante la palabra reservada **LIKE** que se puede usar en combinación de la cláusula **WHERE**.

<expresión> LIKE <patrón>

La expresión representa el campo sobre el cual queremos realizar la búsqueda y el patrón es una cadena que incluye el patrón correspondiente

% Caza con cualquier cade, de cualquier longitud, incluso cadenas separadas

\_Caza con un solo carácter

Ejemplos

|  |  |
| --- | --- |
| SELECT nombre  FROM persona  WHERE apellido\_paterno LIKE ‘S%’ | Busca los registro dónde el apellido paterno empiece con ‘S’, ejemplo, Suarez, Sánchez, Silva |
| SELECT nombre  FROM persona  WHERE apellido\_materno LIKE ‘%am%’ | Busca los registros del campo apellido materno que en alguna parte llegan la cadena ‘am’, ejemplo Ramírez, o Samaniego |
| SELECT nombre  FROM persona  WHERE apellido\_paterno LIKE ‘Mendoz\_’ | Busca los registros del campo apellido materno que lleve la cadena ’Mendoz\_’, buscando un solo carácter, ejemplo Mendoza, Mendoza. |
| SELECT nombre  FROM persona  WHERE apellido\_paterno LIKE ‘M\_n\_oz\_’ | Busca los registros del campo apellido materno que en alguna parte lleve las letras que se le entregan al patrón para buscar, Mandoza, Mendoza, Mindiza. |
| SELECT nombre  FROM persona  WHERE apellido\_paterno NOT LIKE ‘%a’ | Se puede aprovechar el operador lógico y no buscar como tal cadena. |

**Funciones de agrupamiento**

* **SUM** suma todos los valores de una columna
* COUNT cuenta los valores de una columna
* **MIN** obtiene el valor mínimo de una columna
* **MAX** obtiene el valor máximo de una columna
* **AVG** obtiene el promedio de los valores de una columna

Ejemplos

|  |  |
| --- | --- |
| SELECT sum(salario) AS “Salario Total”  FROM empleados  WHERE salario > 5000; |  |
| SELECT count(ocupación) AS “Número de ocupaciones”  FROM persona; |  |
| SELECT max(edad) AS “Mayor”  FROM persona; |  |
| SELECT min(edad) AS “Mayor”  FROM persona; |  |
| SELECT avg(calificacion) “Promedio”  FROM estudiantes  WHERE calificacion >= 6.0; |  |

**Agrupaciones**: toma valores distintos de una columna y permite hace un análisis detallado mediante las funciones de agrupamiento.

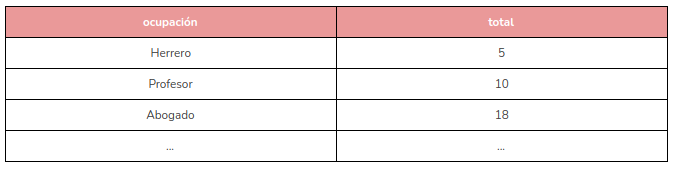
Para usar un agrupamiento se debe usar la restricción **GROUP BY**, que indica los campos por los cuales se realizará el agrupamiento. Los campos por los cuales se agrupará, deben aparecer en la consulta y debe haber al menos una función de agrupamiento proyectada.

Para saber cuántas personas hay por ocupación o agrupación por ocupación.

SELECT ocupacion, count(\*) “Total”

FROM persona

GROUP BY ocupacion;



**Subconsultas**

Una subconsulta consiste de consultas que incluyen otras consultas en su interior. Una subconsulta puede aparecer en la parte SELECT, en la parte FROM o en la parte WHERE de una consulta.

Ejemplo:

SELECT nombre, correo

FROM contactos

WHERE direccion IN

(SELECT direccion

FROM ubicaciones

WHERE codigo\_postal = ‘01060’);

En esta consulta tomamos las columnas resultantes de la subconsulta. En este tipo de consultas MySQL pide poner un nombre a la subconsulta para poder referenciar, en caso de que sea necesario. Esta consulta primero calcula la suma de las calificaciones y el total de alumnos de una tabla y posteriormente se usan estos datos para calcular el promedio.

Notamos también que es posible usar operaciones aritméticas, como es el caso de la suma, resta, división y multiplicación.

SELECT (Calificacion / Alumnos) AS “Promedio”

FROM

(SELECT sum(calif) AS “Calificacion”, count(\*) AS “Alumnos”

FROM alumnos) AS subconsulta;

Claves en las tablas: Llaves primarias y foráneas

Para poder relacionar tablas, se necesita identificar cada registro de forma única, para no tener registros duplicados.

Con el uso de llaves se evita el duplicado de datos, solo se usa el campo que lo identifica o llave

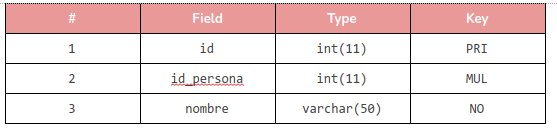
Esta forma de almacenar la información, es útil pues se tienen todos los datos a la mano, sin embargo, si el dueño Manuel, tuviera más de una mascota, se duplicaría su información en varios registros.

La forma en la que SQL evita esta duplicidad de datos, es justamente con el uso de llaves, de esta forma, en lugar de almacenar todos los datos de una persona almacenamos únicamente el campo que lo identifica de manera única, es decir, su llave.

Lógicas

Claves primarias PK (primary key): o llaves primarias, hace que el registro sea unívoco y obligatoriamente no nulo. Puede usarse un campo que no se repita. Se usan para números de cuenta, correos, c. p. entre otros. Se le puede agregar un identificador de tipo entero el cual se aumenta en uno cada que un registro nuevo se añade.

La PK se puede identificar en una tabla con la instrucción **DESCRIBE** en SQL.



Clave foránea FK (foreing jey): o clave secundaria o clave externa, puede ser, o no, una clave primaria dentro de la tabla. Su característica es que es el punto de enlace con otra tabla dónde está, es PK.(1) Representa la realción que pudiera haber entre dos tablas, ejmplo asocialr la tabla Mascotas con su dueño, es decir con un regitro de la tabla personas.

Clave índice: es un campo que facilita la búsqueda dentro de una tabla. Generalmente son campos PK.

Conceptuales

Clave concatenada CK: ayuda a encontrar la singularidad en una tabla combinando dos campos, ya que no hay una llave primaria.

Claves candidatas: cuando existe más de una clave primaria dentro de la tabla.

Join

Hay 3 tipos de join

INNER JOIN (o solo JOIN)

LEFT JOIN (o solo LEFT JOIN)

RIGTH JOIN (o solo RIGHT JOIN)

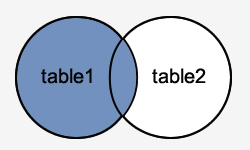
Join: relaciona dos tablas, trayendo todos los campos de ésta siempre y cuando se cumpla la condición de relación.

Ejemplos:

SELECT \*  
FROM persona  
JOIN mascota  
ON persona.id = mascota.id\_persona;

Obtiene una tabla con todos los campos de la tabla persona, todos los campos de la tabla mascota y agrega únicamente aquellos registros donde el id de la persona sea igual al id de la persona asociada a la mascota.

Un join izquierdo, relaciona dos tablas siempre y cuando se cumpla la condición de relación, sin embargo, traerá todos los registros de la tabla izquierda y únicamente aquellos registros que cumplan con la condición de relación de la otra tabla.

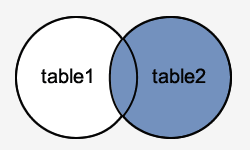


Ejemplo de consulta LEFT JOIN

SELECT \*  
FROM persona  
LEFT JOIN mascota  
 ON persona.id = mascota.id\_persona;

Obtiene una tabla con todos los registros de la tabla persona, pero sólo aquellos de la tabla mascota en donde el id de persona sea igual.

Un join derecho es inverso al join izquierdo, relaciona dos tablas siempre y cuando se cumpla la condición de relación, sin embargo, traerá todos los registros de la tabla derecha y únicamente aquellos registros que cumplan con la condición de relación de la otra tabla.



Ejemplo de consulta RIGTH JOIN

SELECT \*  
FROM persona  
RIGHT JOIN mascota  
 ON persona.id = mascota.id\_persona;

Obtiene una tabla con todos los registros de la tabla mascota, pero sólo aquellos de la tabla persona en donde el id de persona sea igual.

Vistas: son la representación virtual de una consulta en formato de tabla o dicho de otro modo, es el resultado de guardar una consulta para poder consultarla como a cualquier otra tabla. Son útiles cuando el tiempo de procesamiento de una consulta es alto y por lo tanto no tenemos que ejecutarla una y otra vez pues estará disponible.

Para darle un nombre a la consulta que estamos tomando como base, por ejemplo, la siguiente vista toma la consulta que asocia mascotas con su dueño, llamamos a esta vista, duenios. }

Ejemplo de crear vistas

CREATE VIEW duenios AS  
 SELECT \*  
 FROM persona  
 JOIN mascota  
 ON persona.id = mascota.id\_persona;

Una vez creada, podemos consultar los campos que arroja la consulta asociada a una vista, pero en lugar de llamar a la consulta, llamamos a la vista con el nombre que se le dio.

Ejemplo de cómo consultar una vista

SELECT \*  
FROM duenios  
WHERE id\_persona = 2;

COMANDOS SQL

USE <nombre\_db> ; ahora conectarnos a la base datos

SHOW DATABASES ; Para saber que bases de datos hay en el servidor

SHOW TABLES; Para saber las tablas contenidas en la bse de datos

DESCRIBE <nombre\_tabla>; Para conocer los campos de la tabla, el tipo de dato(type), key.

SELECT <campo\_uno>, <campo\_dos> Estructura básica de una consulta, obtiene los datos de los campos o columnas consultadas. Tambien se se conce como proyectar registros

FROM <tabla>;

SELECT \* devuelve todos los campos o columnas de la tabla

FROM <nombre\_tabla>

SELECT \* Estructura WHERE

FROM <nombre\_tabla>

WHERE <campo> <operador\_compración> <objetivo\_a\_coparar>;

<expresión> LIKE <patron> Estrucuta de búsqueda de patrones

SELECT \* Ejemplo del uso de ORDER BY y LIMIT  
FROM persona De la tabla persona filtra las personas  
WHERE estatura = 1.75 que su estatura se igual a 1.75.  
ORDER BY edad DESC Las ordena de acuerdo al campo edad  
LIMIT 3; De forma descendente y solo 3 registros.

SELECT nombre Ejemplo de búsqueda de patrones  
FROM persona  
WHERE apellido\_paterno LIKE ‘S%’

SELECT nombre Ejemplo de búsqueda de patrones  
FROM persona  
WHERE apellido\_materno LIKE ‘%am%’

SELECT nombre Ejemplo de búsqueda de patrones  
FROM persona  
WHERE apellido\_paterno LIKE ‘Mendoz\_’

SELECT sum(salario) Ejemplo de agrupaciones sumando  
AS “Salario Total”  
FROM empleados  
WHERE salario > 5000;

SELECT count(ocupación) Ejemplo de agrupaciones contando totales  
AS “Número de ocupaciones”  
FROM persona;

SELECT max(edad) Ejemplo de agrupaciones máximo dato del  
AS “Mayor” campo  
FROM persona;

SELECT min(edad) Ejemplo de agrupaciones minimo dato del   
AS “Mayor” campo  
FROM persona;

SELECT avg(calificacion) Ejemplo de agrupaciones promedio de las   
“Promedio” calificaciones  
FROM estudiantes  
WHERE calificacion >= 6.0;

SELECT ocupacion, count(\*) Ejemplo del uso de GROUP BY  
“Total”  
FROM persona  
GROUP BY ocupacion;

SELECT nombre, correo Ejemplo de subconsulta  
FROM contactos  
WHERE direccion IN   
 (SELECT dirección  
 FROM ubicaciones  
 WHERE codigo\_postal = ‘01060’);

SELECT \* Ejemplo de join inner  
FROM persona  
JOIN mascota  
ON persona.id = mascota.id\_persona;

SELECT \* Ejemplo de consulta LEFT JOIN  
FROM persona  
LEFT JOIN mascota  
 ON persona.id = mascota.id\_persona;

SELECT \* Ejemplo de consulta RIGTH JOIn  
FROM persona  
RIGHT JOIN mascota  
 ON persona.id = mascota.id\_persona;

CREATE VIEW duenios AS Ejemplo de crear vistas

SELECT \*  
 FROM persona  
 JOIN mascota  
 ON persona.id = mascota.id\_persona;

SELECT \* Ejemplo de cómo consultar una vista  
FROM duenios  
WHERE id\_persona = 2;

Herramientas

<https://www.lucidchart.com/pages/es> Diagramas

<https://docs.microsoft.com/es-es/office/troubleshoot/access/database-normalization-description>

<https://www.kaggle.com/> Bases de datos

color.tableaumagic.com paleta de colores, con una imgen

Fonts.google.com

App.moqups.com

PEXEL fotos

<https://iconos8.es/> iconos y fotos

<https://www.freepik.es/> fotos

<https://coolors.co/> generador de paleras de colores

<https://color.adobe.com/es/create/color-wheel> paletas de colores

<https://logomakr.com/> Logos

<https://www.flaticon.es/> iconoes