

Bases de Datos I

SEGUNDO BIMESTRE

PROFESOR: Rodrigo Soto
rsoto@digitalhouse.com
[Cronograma](#)

Clase 1: Módulo I - Bienvenida

¿Por qué bases de datos?

Las bases de datos surgieron por diferentes motivos:

- Seguridad
- Control
- Escalabilidad

Esta herramienta nos permite:

- Modelar datos
- Guardar datos
- Consultar

¿Qué es una BBDD?

Es una pieza de software que nos va a permitir almacenar información.

¿Qué herramientas vamos a utilizar?

- Un gestor de base de datos.
- Una herramienta que nos permita manipular la base de datos.

MySQL Workbench es una herramienta multiplataforma que nos va a permitir trabajar con bases de datos en forma gráfica.

Clase 2: Módulo I - Introducción a base de datos

¿Qué es una Base de Datos?

Es un conjunto de datos pertenecientes a un mismo contexto organizados para un propósito específico.

Una base de datos nos permite:

- Almacenar (agregar, modificar y eliminar) datos.
- Acceder a los datos.
- Manipularlos y combinarlos.
- Analizar datos.
- Entre otras cosas más.

Las bases de datos las hay de dos tipos: relacionales y no relacionales.

Las bases de datos no solo almacenan información, también nos permiten realizar otras tareas:

- Evita la redundancia de datos: Evita duplicar datos
- Conserva la integridad de los datos: Evita inconsistencias

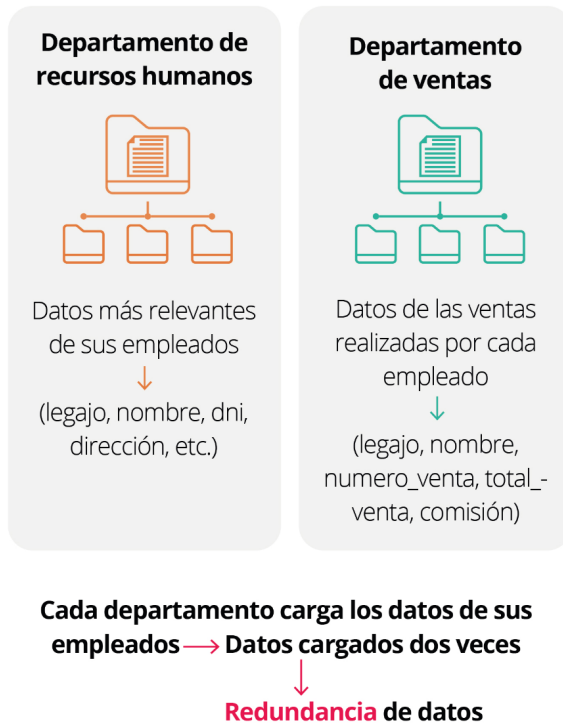
Bases de datos vs Archivos planos

¿Base de datos o archivos planos?

Un archivo está compuesto por registros y cada registro está compuesto por un conjunto de campos.

Algunos inconvenientes del uso de archivos planos:

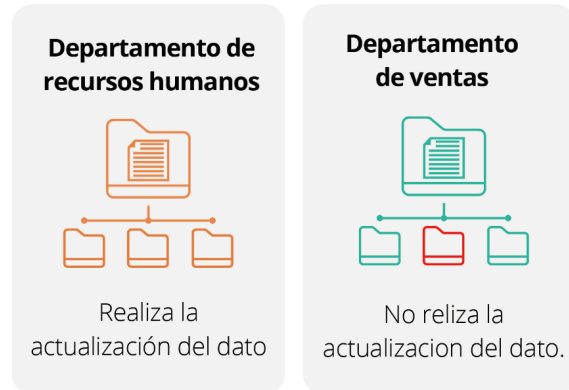
Redundancia



Consistencia e integridad de datos



Un empleado tiene mal grabado su apellido



La información sobre el empleado queda **inconsistente** y, como consecuencia, se pierde la **integridad** de los datos ya que para un mismo legajo se tienen dos personas distintas.

Además, sucede algo peor, como únicamente RR. HH. realizó la actualización del apellido, las ventas registradas no pertenecen a ningún empleado actual y podría haber errores en la liquidación de comisiones.

Motores de Bases de datos

Vimos que las bases de datos realizan chequeos de:

- Integridad
- Coherencia de los datos.
- Redundancia

Pero ¿Cómo realiza estas tareas? ¿Quién es el encargado de realizarlas?

El motor de BBDD también conocido como Sistema de gestión de base de datos. Es una capa de software, cuyo objetivo es:

- Almacenar datos

-
- Procesar datos
 - Proteger datos

Su rol es:

- Dirigir
- Generar políticas
- Velar por la seguridad

Poseen distintas:

- Consistencia de los datos
- Respaldo
- Integridad
- Seguridad
- Acceso concurrente
- Tiempo de respuesta
- Capacidades



Todos los motores poseen un estándar común que no permite trabajar en una base de datos independientemente de su motor: *SQL*.

SQL es un lenguaje que nos asegura el acceso a cualquiera de los motores de búsqueda disponibles de forma transparente. Nos permite:

- Crear BBDD
- Modificarlas

-
- Escribir datos
 - Consultar y analizar datos

Características de MySQL

- Gratuita
- Código abierto
- Confiable
- Compatible con la mayoría de los SO y sistemas de hosting existentes.

Modelos de bases de datos

¿Cómo se comienza a desarrollar una base de datos?

1. Normalmente se inicia con un requerimiento o necesidad de una aplicación o sistema.
Lo importante es tener en claro qué es lo que se tiene que almacenar, y por ello que este requerimiento tiene que estar lo más detallado posible.
2. Una vez que tenemos el requerimiento, se inicia la etapa de modelado.
Este documento nos permite validar que entendimos el requerimiento y empezar a pensar cómo lo vamos a implementar.
3. Una vez implementada la base de datos, ¡este documento no se elimina! Sino que sirve de documentación para volver a consultar y entender el porqué se implementó la base de datos de una determinada forma.

Modelado de datos

Un modelo es un conjunto de herramientas conceptuales para describir datos, sus relaciones, su significado y sus restricciones de consistencia.

El modelado de datos es una manera de estructurar y organizar los datos para que se puedan utilizar fácilmente por las bases de datos.

Beneficios

- Registrar los requerimientos de datos de un proceso de negocio.
- Se puede descomponer un proceso complejo en partes.

-
- Permite observar patrones.
 - Sirve de plano para construir la base de datos física.
 - El modelo de datos ayuda a las empresas a comunicarse dentro y entre las organizaciones.
 - Proporciona soporte ante los cambios de requerimientos del negocio o aplicaciones.

Tipos de modelos

Existen 3 tipos de modelos que podrían implementarse:

- **Conceptual:** Es un modelo con un diseño muy general y abstracto, cuyo objetivo es explicar la visión general del negocio o sistema.
- **Lógico:** El modelo lógico es una versión completa que incluye todos los detalles acerca de los datos. Explica qué datos son importantes, su semántica, relaciones y restricciones. Explica el “qué”.

Un modelo de datos lógico describe los datos con el mayor detalle posible, independientemente de cómo se implementarán físicamente en la base de datos.

Describe los elementos importantes del negocio, qué significa cada objeto, su nivel de detalle, cómo se relacionan entre sí y sus restricciones.

Las características de un modelo de datos lógicos incluyen:

- Se definen cuáles son los conceptos importantes sobre los que hay que almacenar información. Estos elementos se denominan entidades.
 - Se especifican todos los atributos para cada una de las entidades.
 - Se conectan las entidades mediante relaciones.
 - Se especifica la clave principal para cada entidad.
 - Se especifican las claves externas (claves que identifican la relación entre diferentes entidades).
 - La normalización ocurre en este nivel.
- **Físico:** Es un modelo que implementa el modelo lógico. Es un esquema que se va a implementar dentro de un sistema de gestión de bases de datos. Explica el “cómo”.

El modelo de datos físicos representa cómo se construirá el modelo en la base de datos.

Un modelo de base de datos física muestra todas las estructuras de tablas, incluidos el nombre de columna, el tipo de datos de columna, las restricciones de columna, la clave principal, la clave externa y las relaciones entre las tablas.

Las características de un modelo de datos físicos incluyen:

- Especificación de todas las tablas y columnas.
- Las claves externas se usan para identificar relaciones entre tablas.
- La desnormalización puede ocurrir según los requisitos del usuario.

Las consideraciones físicas pueden hacer que el modelo de datos físicos sea bastante diferente del modelo de datos lógicos.

El modelo físico puede diferir de un motor de bases de datos a otro —no es lo mismo implementar en Ms. SQL Server que en MySQL—.

Los pasos para el diseño del modelo de datos físicos son los siguientes:

1. Convertir entidades en tablas.
2. Convertir relaciones en claves externas.
3. Convertir atributos en columnas.
4. Modificar el modelo de datos físicos en función de las restricciones / requisitos físicos.

Modelo lógico vs Modelo físico

Modelo Lógico	Modelo Físico
Describe el “qué”.	Describe el “cómo”.
Explica el negocio.	Explica técnicamente cómo se van a almacenar los datos.
Es independiente de la implementación.	Explica la implementación en el sistema de gestión de bases de datos.

Responsable: el analista.	Responsable: administrador de bases de datos o símil.
---------------------------	---

Test

1. **¿Qué es una base de datos?** Es un “almacén” que nos permite guardar grandes cantidades de información de forma organizada.
2. **¿Cuál de las siguientes opciones es un ejemplo de base de datos?** La historia clínica de un paciente.
3. **Las ventajas de usar una base de datos frente a archivos planos son** Integridad de datos, consistencia y redundancia de datos.
4. **¿Qué se entiende por integridad de datos?** Se refiere a la corrección y exactitud de los datos en una base de datos.
5. **¿Cuál de los siguientes ejemplos explica un problema de redundancia en un formulario de inscripción?** Que permita registrarse más de una vez.
6. **Cuando modelamos una base de datos, ¿qué modelado se realiza primero?**
Modelado lógico
7. **Un modelo lógico está compuesto por:** Entidades, atributos y relaciones.

Clase 3: Módulo I - Cierre de semana

Clase 4: Módulo II - Entidades

El objetivo de cualquier sistema de información es representar mediante abstracciones del mundo real toda la información necesaria para el cumplimiento de los fines.

Modelo de bases de datos: es una colección de herramientas conceptuales para describir datos, relaciones entre ellos, semántica asociada a los datos y restricciones de consistencia.

Modelos

Tipos

- **Modelo conceptual basado en objetos:** Fue definido por Peter Chen en 1976. Se utiliza para la representación de la realidad No comprometida con ningún entorno informático: Sería el Modelo Entidad-Relación propiamente dicho.
- **Modelo lógico basado en objetos:** determinan algunos criterios de almacenamiento y de operaciones de manipulación de los datos dentro de un entorno informático.
- **Modelo entidad-relación:** se basa en una percepción del mundo real, que consiste en un conjunto de objetos básicos llamados entidades y de relaciones entre ellos. Se emplea para interpretar, especificar y documentar los requerimientos para los sistemas de bases de datos.

Es un método de representación abstracta del mundo real centrado en las restricciones o propiedades lógicas de una base de datos.

Entidades

Una entidad es un objeto, real o abstracto, acerca del cual se recoge información de interés para la base de datos.

Tipos

- **Entidades fuertes:** tienen existencia por sí mismas. (alumnos, empleados, departamento.)
- **Entidades débiles:** dependen de otra entidad para su existencia (hijos de empleados)

Se define como **ocurrencia de entidad** al conjunto de datos para una entidad en particular.

Por ejemplo: 125, Juan Pérez, casado, 23 años.

Cada entidad tiene propiedades particulares llamadas **atributos**.

Atributos

Los atributos describen las características de una entidad.

Por ejemplo:

Entidad: clientes

Atributos: legajo, nombre, domicilio, etc.

Tipos

- **Atributo con simple valor:** Cuando un atributo tiene un simple valor para una identidad particular.
Por ejemplo: Una persona que tiene un valor por su fecha de nacimiento y la fecha de nacimiento es un simple valor de la persona.
- **Atributo multivalor:** Cuando un atributo tiene una serie de valores para identificarse.
Por ejemplo: El atributo teléfonos de un cliente que puede contener uno o más números de teléfono.
- **Atributos derivados:** Cuando los valores de un atributo son afines y el valor para este tipo de atributo se puede derivar de los valores de otros atributos.
Por ejemplo: La edad y fecha de nacimiento de una persona; si conocemos la fecha de nacimiento, podemos calcular su edad, en este caso se dice que la edad es un atributo derivado del atributo fecha de nacimiento.
- **Atributo clave:** Las entidades pueden contener un atributo que identifica cada una de las ocurrencias de la entidad. Es decir, usualmente contienen un atributo que diferencia los ítems entre sí.
Por ejemplo: En la entidad clientes el atributo documento puede ser un atributo clave. No necesariamente el atributo clave debe ser un solo atributo, hay casos en que varios atributos forman una llave. Por ejemplo: tipo más número de factura.
- **Atributos nulos:** Se usa cuando una entidad no tiene valor para un atributo o que el valor es desconocido.

Claves

Tipos

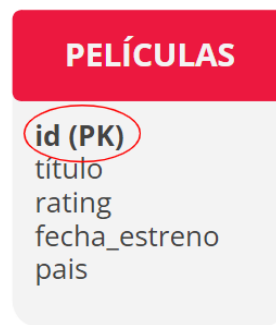
- **Clave candidata:** Se compone por uno o más atributos cuyos valores identifican unívocamente a cada ocurrencia de la entidad, sin que ningún subconjunto de ellos pueda realizar esta misma función. Una clave candidata es una posible clave

primaria. Pueden definirse varias claves candidatas para luego seleccionar la más adecuada.

- **Clave primaria:** Primary Key es un campo que identifica a cada fila de una tabla de forma única. Es decir que no puede haber dos filas en una tabla que tengan la misma PK.

Está compuesta por uno o más atributos cuyos valores identifican unívocamente a cada ocurrencia de la entidad. No pueden contener valores nulos ni repetidos. Esta clave es una de aquellas que anteriormente se seleccionaron como candidata.

Para identificar la clave primaria en una entidad, podemos escribir el atributo en negrita seguido de las iniciales PK entre paréntesis.



- **Superclave:** Es el conjunto de uno o más atributos que, tomados colectivamente, permiten identificar de forma unívoca a la ocurrencia de una entidad. Se utiliza generalmente en las tablas de relación, este concepto se desarrollará en las próximas clases.

Convención de nombres

En los nombres de entidades y atributos siempre se debe utilizar sustantivos en singular o plural. No se puede utilizar eñes, espacios ni acentos. Si el nombre se compone por más de una palabra, se deben reemplazar los espacios con guiones bajos (snake case) o eliminar dicho espacio y colocar una mayúscula en la inicial de cada palabra (camel case).

Ej. Así podríamos asignar un nombre para la siguiente frase:

“costos anuales” → costo_anual → costoAnual

“costos anuales” → costos_anuales → costosAnuales

Cabe aclarar que, debido a que MySQL utiliza directorios y archivos para almacenar bases de datos y tablas, los nombres se distinguen entre mayúsculas y minúsculas solo si el sistema operativo posee un sistema de archivo sensible al tipo.

Windows: No distingue entre mayúsculas y minúsculas.

Algunas versiones de Unix y Linux: Distinguen entre mayúsculas y minúsculas. (Case Sensitive)

Datos

Los datos son los valores que pueden tener los atributos. Nos permiten entender si son datos numéricos, textos, fechas, si tienen un formato en particular, si son obligatorios u opcionales.

No se modelan, pero nos permite entender mejor las entidades.

[Caso de estudio: Uber](#)

[Caso Práctico: Playground](#)

Clase 5: Módulo II - Datos

Tipos de datos

Los datos o atributos de cada registro de una tabla tienen que ser de un tipo de dato concreto.

Cuando diseñamos una base de datos tenemos que pensar qué tipo de datos requerimos para nuestro modelo.

Cada tipo de dato tiene un tamaño determinado y cuanto más precisión apliquemos en su definición, más rápido y performante va a funcionar MySQL.

- **Datos numéricos:** Donde tenemos varios subtipos:
 - **INT:** -2.147.483.648 a 2.147.483.647 (sin signo de 0 a 4.294.967.295)
 - **TINYINT:** -128 a 127 (sin signo de 0 a 255)
 - **SMALLINT:** -32.768 a 32.767 (sin signo de 0 a 65.535)

-
- **MEDIUMINT**: -8.388.608 a 8.388.607 (sin signo de 0 a 16.777.215)
 - **BIGINT** (DNI): -9.223.372.036.854.775.808 a 9.223.372.036.854.775.807 (sin signo de 0 a 18.446.744.073.709.551.615)
**Varía el máximo y mínimo de valores que podemos ingresar.*
 - **DECIMAL**
 - **FLOAT(n,d)**: Almacenan números de coma flotante pequeño. Tienen precisión simple para la parte decimal (máx. 7 dígitos).
n → 1 a 24 dígitos (incluyendo la parte decimal).
d → 0 a 7 dígitos dependiendo de cuánto se asigne en n.
 - **DOUBLE(n,d)**: Almacenan números de coma flotante grande. Tienen precisión doble para la parte decimal (máx. 15 dígitos).
n → 0 a 53 dígitos (incluyendo la parte decimal).
d → 0 a 15 dígitos dependiendo de cuánto se asigne en n.
**Para aclarar números con coma (,). Se sugiere usar DECIMAL.*
 - **BOOLEAN**: Se recomienda en MySQL no utilizarlo.
**Reemplazarlo con un TINYINT (1: True, 0:False)*

Unsigned: Propiedad que usamos para tener ese negativo en positivo.

- **Datos de tipo texto**: MySQL nos ofrece 3 variantes distintas:
 - **CHAR(n)**: n → 1 a 255 caracteres.
Viene de la mano de un número que indica la cantidad de caracteres exacta que va a tener nuestro texto. Tiene pocos casos de uso, ya que es riguroso.
Uso: Código postal.
 - **VARCHAR(n)**: n → 1 a 21.845 caracteres.
Lleva un número asociado, pero no es un número exacto de caracteres, representa el número máximo.
Uso: Nombre - VARCHAR(100)
 - **TEXT**: Entradas que no tienen límite.
 - TINYTEXT: 0 a 255 caracteres.
 - TEXT: 0 a 65,535 caracteres.
 - MEDIUMTEXT: 0 a 16.777.215 caracteres.

-
- **LONGTEXT:** 0 a 4.294.967.295 caracteres.

- **Datos de tipo fecha:** A la hora de almacenar fechas, hay que tener en cuenta que MySQL no comprueba de una manera estricta si una fecha es válida o no.

Donde hay 3 subtipos:

- **DATE:** Para aclarar sólo fechas. Respetando el formato YYYY-MM-DD.
Valores permitidos: '0001-01-01' a '9999-12-31'.
La fecha se debe colocar entre comillas simples o dobles y se separa por guiones. Ejemplo: '2021-05-15'.
- **TIME:** Para aclarar sólo horarios. Respetando el formato HH:MM:SS.
Valores permitidos: '00:00:00' a '23:59:59'.
La hora se debe colocar entre comillas simples o dobles y se separa por dos puntos. Ejemplo: '11:50:55'.
- **DATETIME:** Para aclarar ambas cosas. Respetando el formato YYYY-MM-DD HH:MM:SS.
Valores permitidos: '0001-01-01 00:00:00' a '23:59:59 9999-12-31'.
La fecha se debe colocar entre comillas simples o dobles, un espacio y la hora que se debe separar por dos puntos. Ejemplo: '2021-05-15 11:50:55'.

Restricciones

- **NOT NULL:** Columnas que no pueden quedar vacías. Dato obligatorio.
- **UNIQUE:** No se puede repetir, aunque no sea clave primaria.
- **DEFAULT y AUTO_INCREMENT:** Como si fuera una columna id que va a ir autoincrementando en cada una de las filas.



Test

1. **¿Qué tipos de datos existen?** Texto, fechas y números.
2. **En un tipo de datos numérico, tenemos que tener en cuenta:** La cantidad de dígitos de longitud.
3. **Para el stock de una fábrica de muebles, ¿qué tipo de datos utilizarías?** INT
4. **Para un título de un libro, si no se conoce la longitud, conviene utilizar:**
VARCHAR

Clase 6: Módulo II - Cierre de semana

[Ejercicios prácticos](#)

Clase 7: Módulo II - Relaciones

Cuando estamos modelando y detectamos las entidades o tablas, es normal que se relacionen entre sí.

Las [relaciones](#) indican cómo se van a relacionar dos tablas. Existen 3 tipos de relaciones:




- 1 a 1
- 1 a Muchos: Sumar FK (id) del 1 a la tabla de Muchos
- Muchos a Muchos: Requiere una tabla intermedia o pivot.

¿Cómo podemos saber cómo se relaciona una entidad con otra?

Planteando un ejemplo concreto que nos ayude a definir cómo interactúan esas dos entidades entre sí.

Cardinalidad

Es la forma en que se relacionan las entidades.

Cardinalidad	Se lee	Representación
1:1	Uno a uno	
1:M	Uno a muchos	
N:M	Muchos a muchos	

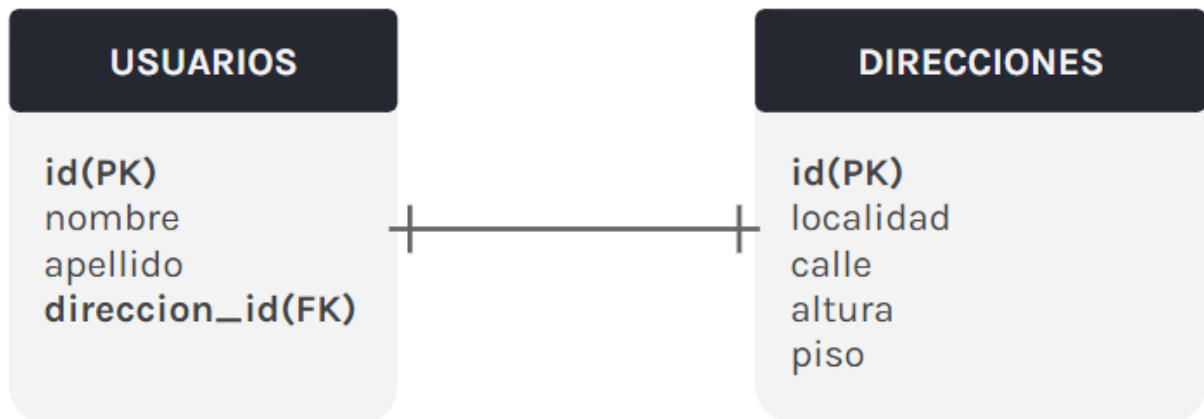
** Nota: Muchas veces vemos las notaciones como 1 a N o 1 a M. Son iguales, la letra se utiliza para representar "Muchos".*

***Preguntar en clase:** En el caso de relacionar 2 tablas, si una de ellas ya estaba relacionada con otras, debo incluir la relación anterior? ¿O no hace falta?
Por otro lado, en qué cambia la línea punteada en SQL?

Uno a uno (1:1)

Un usuario tiene solo una dirección. Una dirección pertenece solo a un usuario.

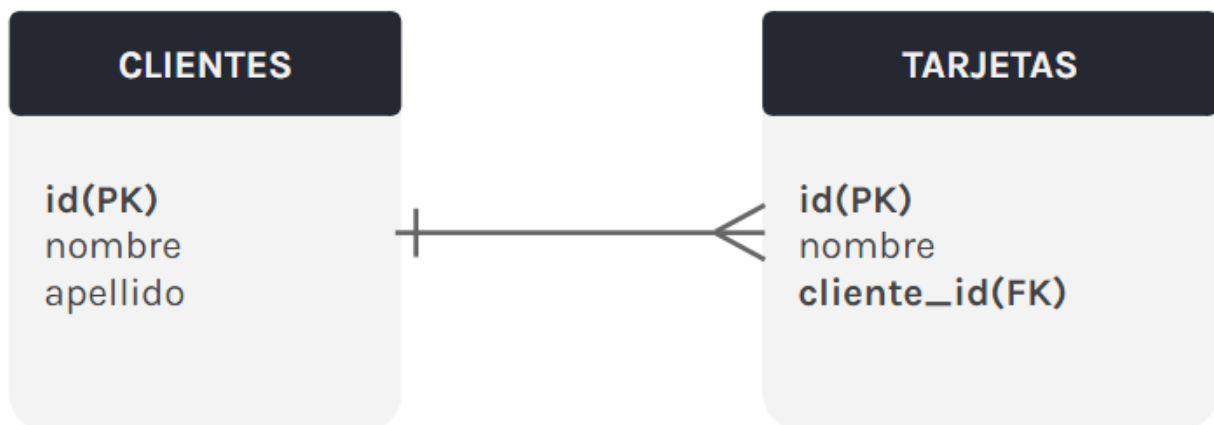
Para establecer la relación colocamos la clave primaria de la dirección en la tabla de usuarios, indicando que esa dirección está asociada a ese usuario (Clave foránea).



Uno a muchos (1:N)

Un cliente puede tener muchas tarjetas. Una tarjeta pertenece solo a un cliente.

Para establecer la relación colocamos la clave primaria del cliente en la tabla de tarjetas, indicando que esas tarjetas están asociadas a un usuario en particular.



Muchos a muchos (N:M)

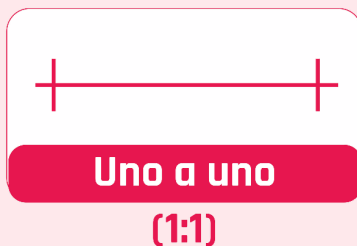
Un cliente puede comprar muchos productos. Un producto puede ser comprado por muchos clientes.

En las relaciones N:M, en la base de datos, la relación en sí pasa a ser una tabla. Esta tabla intermedia —también conocida como tabla pivot— puede tener 3 datos: una clave primaria (PK) y dos claves foráneas (FK), cada una haciendo referencia a cada tabla de la relación.

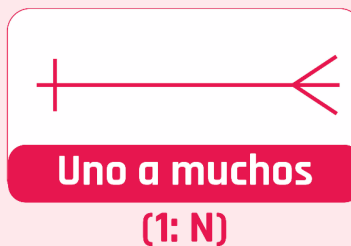


Glosario de Relaciones

Tipos de relaciones entre las entidades de una base de datos



Cada elemento de una entidad solo puede relacionarse con un elemento de otra entidad.



Un elemento de una entidad A puede relacionarse con varios elementos de una entidad B.



Muchos elementos se relacionan con más de un elemento en otra tabla. Las relaciones N a N se modifican creando una tercera entidad para la relación. Entidad que lleva entre sus datos las dos claves primarias de A y B en rol de claves foráneas y una clave primaria propia como clave de la relación.

Test

1. ¿Qué tipos de relaciones existen en las bases de datos relacionales? 1 a 1, 1 a Muchos, Muchos a Muchos.
2. ¿Qué es una clave foránea? Una referencia a la clave primaria de otra tabla.
3. Si queremos modelar que una persona tiene muchas mascotas, pero cada mascota tiene un dueño y utilizando como guía los siguientes modelos:



¿Qué cambios se pueden proponer? Agregar la columna persona_id en la tabla Mascotas.

4. Supongamos que estamos trabajando en el sistema de reservas de un hotel. En este mismo sistema tenemos una entidad Habitaciones y una entidad Huéspedes.

Ahora queremos agregar la relación de que un huésped puede reservar varias habitaciones y que una habitación puede ser reservada por varios huéspedes. Utilizando como guía los siguientes modelos:



¿Qué cambios se pueden proponer? Crear una tabla intermedia.

Clase 8: Módulo III - SQL

Las sentencias SQL se agrupan en dos categorías según su funcionalidad o propósito:

- **Lenguaje de definición de datos (DDL):** son sentencias para la creación de tablas y registros. Es decir, se utilizan para realizar modificaciones sobre la estructura de la base de datos.

-
- **Lenguaje de manipulación de datos (DML):** son sentencias para la consulta, actualización y borrado de datos. Es decir, se utilizan para realizar consultas y modificaciones sobre los registros almacenados dentro de cada una de las tablas.

BBDD: Musimundos

Cómo construir una base de datos: Reverse Engineer e Insert Into

Create, drop, alter

Las directrices create, drop y alter nos van a permitir llevar a cabo cada una de estas acciones que, vale la pena mencionar, son bastante habituales dentro del proceso de trabajo con bases de datos.

- **Create table:** podemos crear una tabla desde cero, junto con sus columnas, tipos y constraints., debemos aclarar:

```
CREATE TABLE new_tbl
```

- nombre de la tabla
- sus columnas
- sus constraints: Las limitaciones o (Constraints) de SQL se utilizan para especificar reglas para los datos de una tabla. Si hay alguna violación entre la restricción y acción de datos, la acción se aborta por la restricción.

```
SQL CREATE TABLE nombre_de_la_tabla (  
      nombre_de_la_columna_1 TIPO_DE_DATO CONSTRAINT,  
      nombre_de_la_columna_2 TIPO_DE_DATO CONSTRAINT  
    )
```

```
SQL CREATE TABLE post (  
      id INT PRIMARY KEY AUTO_INCREMENT,  
      titulo VARCHAR(200)  
    )
```

SQL

```
CREATE TABLE peliculas (  
    id INT PRIMARY KEY AUTO_INCREMENT,  
    title VARCHAR(500) NOT NULL,  
    rating DECIMAL(3,1) NOT NULL,  
    awards INT DEFAULT 0,  
    release_date DATE NOT NULL,  
    length INT NOT NULL  
);
```

- **Drop table:** Borrar tablas, debemos aclarar

```
DROP TABLE    tbl_name
```

- nombre de la tabla a borrar

```
SQL DROP TABLE IF EXIST peliculas;
```

- **Alter table:** Permite modificar la tabla

```
ALTER TABLE tbl_name
```

Permite alterar una tabla ya existente y va a operar con tres comandos:

- **ADD:** para agregar una columna.
- **MODIFY:** para modificar una columna.
- **DROP:** para borrar una columna.

```
SQL ALTER TABLE peliculas
ADD rating DECIMAL(3,1) NOT NULL;
```

Agrega la columna **rating**, aclarando tipo de dato y constraint.

```
SQL ALTER TABLE peliculas
MODIFY rating DECIMAL(4,1) NOT NULL;
```

Modifica el decimal de la columna **rating**. Aunque el resto de las configuraciones de la tabla no se modifiquen, es necesario escribirlas en la sentencia.

```
SQL ALTER TABLE peliculas
DROP rating;
```

Borra la columna **rating**.

- **Create database:** podemos crear una base de datos desde cero.

```
SQL CREATE DATABASE miprimerabasededatos;
USE miprimerabasededatos;
```

- **ForeignKey:** Cuando creamos una columna que contenga una id foránea, será necesario usar la sentencia FOREIGN KEY para aclarar a qué tabla y a qué columna hace referencia aquel dato.

Es importante remarcar que la tabla “clientes” deberá existir antes de correr esta sentencia para crear la tabla “ordenes”.

```
SQL CREATE TABLE ordenes (
    orden_id INT NOT NULL,
    orden_numero INT NOT NULL,
    cliente_id INT,
    PRIMARY KEY (orden_id),
    FOREIGN KEY (cliente_id) REFERENCES clientes(id)
);
```

Insert, update, delete

Al momento de trabajar con tablas, indefectiblemente vamos a querer insertar, actualizar o eliminar registros. Estas tres funciones son llevadas a cabo gracias a las tres directrices principales que tiene SQL para esta finalidad.

Insert nos va a permitir agregar datos, con **delete** podremos borrarlos y con **update** podremos modificar los registros existentes en una tabla.

- **Insert:** Existen dos formas de agregar datos en una tabla:
 - **Insertando datos en todas las columnas:** Si estamos insertando datos en todas las columnas, no hace falta aclarar los nombres de cada columna. Sin embargo, el orden en el que insertemos los valores, deberá ser el mismo orden que tengan asignadas las columnas en la tabla.

```
SQL  INSERT INTO table_name (columna_1, columna_2, columna_3, ...)
      VALUES (valor_1, valor_2, valor_3, ...);
```

```
SQL  INSERT INTO artistas (id, nombre, rating)
      VALUES (DEFAULT, 'Shakira', 1.0);
```

- **Insertando datos en las columnas que especifiquemos:** Para insertar datos en una columna en específico, aclaramos la tabla y luego escribimos el nombre de la o las columnas entre los paréntesis.

```
SQL  INSERT INTO artistas (nombre)
      VALUES ('Calle 13');
```

```
SQL  INSERT INTO artistas (nombre, rating)
      VALUES ('Maluma', 1.0);
```

- **Update:** modificará los registros existentes de una tabla.

**Es importante no olvidar el WHERE cuando escribimos la sentencia, aclarando la condición.*

```
SQL UPDATE nombre_tabla
    SET columna_1 = valor_1, columna_2 = valor_2, ...
    WHERE condición;
```

```
SQL UPDATE artistas
    SET nombre = 'Charly Garcia', rating = 1.0
    WHERE id = 1;
```

- **Delete:** podemos borrar información de una tabla.

**Es importante recordar utilizar siempre el WHERE en la sentencia para agregar la condición de cuáles son las filas que queremos eliminar. Si no escribimos el WHERE, estaríamos borrando toda la tabla y no un registro en particular.*

```
SQL DELETE FROM nombre_tabla WHERE condición;
```

```
SQL DELETE FROM artistas WHERE id = 4;
```

SELECT

Toda consulta a la base de datos va a empezar con la palabra SELECT.

Su funcionalidad es la de realizar consultas sobre una o varias columnas de una tabla.

Para especificar sobre qué tabla queremos realizar esa consulta usamos la palabra FROM seguida del nombre de la tabla.

SQL

```
SELECT nombre_columna, nombre_columna, ...  
FROM nombre_tabla;
```

id	título	rating	fecha_estreno	país
1001	Pulp Fiction	9.8	1995-02-16	Estados Unidos
1002	Kill Bill	9.5	2003-11-27	Estados Unidos

De esta tabla completa, para conocer solamente los títulos y ratings de las películas guardadas en la tabla películas, podríamos hacerlo ejecutando la siguiente consulta:

SQL

```
SELECT id, titulo, rating  
FROM peliculas;
```

WHERE y ORDER BY

WHERE

La funcionalidad del WHERE es la de condicionar y filtrar las consultas SELECT que se realizan a una base de datos.

SQL

```
SELECT nombre_columna_1, nombre_columna_2, ...  
FROM nombre_tabla  
WHERE condicion;
```

Teniendo una tabla clientes, podría consultar primer nombre y apellido, filtrando con un WHERE solamente los usuarios que su país es igual a Argentina de la siguiente manera:

SQL

```
SELECT primer_nombre, apellido  
FROM clientes  
WHERE pais = 'Argentina';
```

Operadores

Operadores

=>	Igual a	IS NULL>	Es nulo
>>	Mayor que	BETWEEN>	Entre dos valores
>=>	Mayor o igual que	IN>	Lista de valores
<>	Menor que	LIKE>	Se ajusta a...
<=>	Menor o igual que			
<>>	Diferente a			
!=>	Diferente a			

ORDER BY

ORDER BY se utiliza para ordenar los resultados de una consulta según el valor de la columna especificada. Por defecto, se ordena de forma ascendente (ASC) según los valores de la columna. También se puede ordenar de manera descendente (DESC) aclarándolo en la consulta.

SQL

```
SELECT nombre_columna1, nombre_columna2
FROM tabla
WHERE condicion
ORDER BY nombre_columna1;
```

Teniendo una tabla usuarios, podría consultar los nombres, filtrar con un WHERE solamente los usuarios mayores de 21 años y ordenarlos de forma descendente tomando como referencia la columna nombre.

SQL

```
SELECT nombre, rating
FROM artistas
WHERE rating > 1.0
ORDER BY nombre DESC;
```

Clase 9: Módulo III - Cierre de semana

Clase 10: Módulo III - CRUD y Checkpoint I

¿Qué es un CRUD?

CRUD es el acrónimo de "Crear, Leer, Actualizar y Borrar" (Claramente que en inglés, Create, Read, Update and Delete).

Clase 11: Módulo III - Uso DML y Queries ML

[Between y Like](#)

Las directrices Between y Like son fundamentales para poder hacer este tipo de consultas y más.

- **BETWEEN:** Cuando necesitamos obtener valores dentro de un rango, usamos el operador BETWEEN.
 - BETWEEN incluye los extremos.
 - BETWEEN funciona con números, textos y fechas.
 - Se usa como un filtro de un WHERE.

Por ejemplo, coloquialmente:

Dados los números: 4, 7, 2, 9, 1

Si hiciéramos un BETWEEN entre 2 y 7 devolvería 4, 7, 2 (excluye el 9 y el 1, e incluye el 2).

Query de ejemplo

Con la siguiente consulta estaríamos seleccionando nombre y edad de la tabla alumnos solo cuando las edades estén entre 6 y 12.

```
SQL SELECT nombre, edad
FROM alumnos
WHERE edad BETWEEN 6 AND 12;
```

- **LIKE:** Cuando hacemos un filtro con un WHERE, podemos especificar un patrón de búsqueda que nos permita especificar algo concreto que queremos encontrar en los registros. Eso lo logramos utilizando comodines (wildcards).

Por ejemplo, podríamos querer buscar:

- Los nombres que tengan la letra 'a' como segundo carácter.
- Las direcciones postales que incluyan la calle 'Monroe'.
- Los clientes que empiecen con 'Los' y terminen con 's'.

Comodines

- **COMODÍN (%):** Es un sustituto que representa cero, uno, o varios caracteres.
- **COMODÍN (_):** Es un sustituto para un solo carácter.

Queries de ejemplo

SQL

```
SELECT nombre  
FROM usuarios  
WHERE nombre LIKE '_a%';
```

Devuelve aquellos nombres que tengan la letra 'a' como segundo carácter.

SQL

```
SELECT nombre  
FROM usuarios  
WHERE direccion LIKE '%Monroe%';
```

Devuelve las direcciones de los usuarios que incluyan la calle 'Monroe'.

SQL

```
SELECT nombre  
FROM clientes  
WHERE nombre LIKE 'Los%s';
```

Devuelve los clientes que empiecen con 'Los' y terminen con 's'.