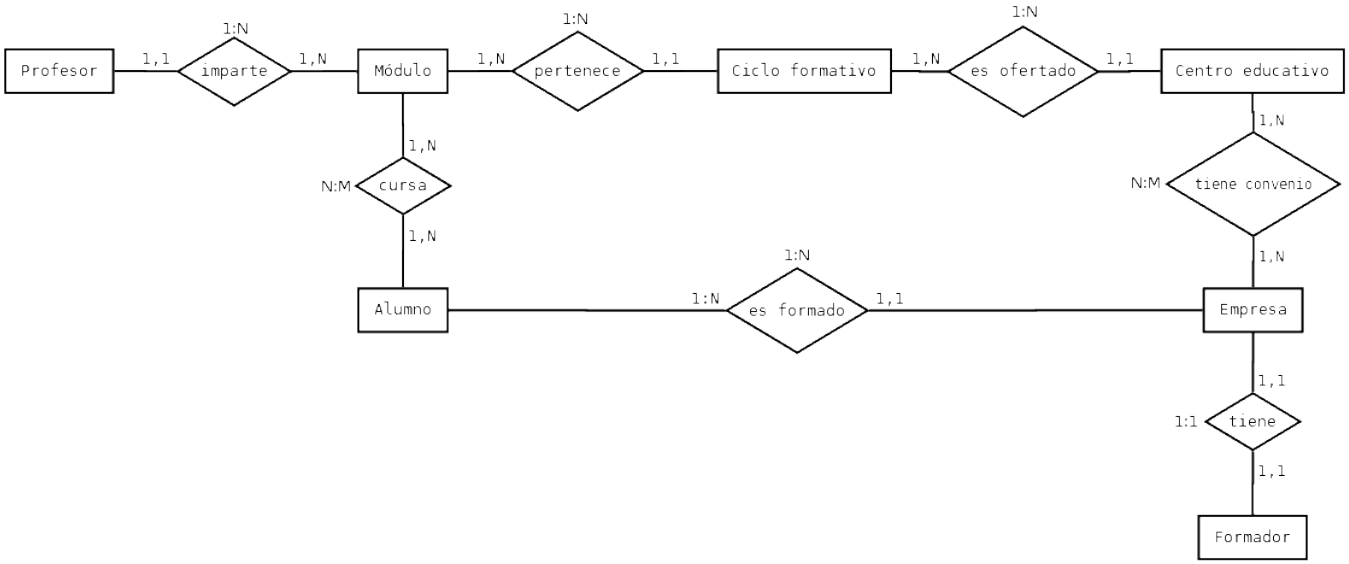
La **modalidad de Formación Profesional Dual** implica que puedo realizar un Ciclo Formativo combinando estancias en el centro educativo con estancias en empresas del sector y de la zona, lo que supone una adaptación máxima a lo que piden las empresas del entorno a los candidatos en sus ofertas de empleo.

INDICE

BASE DE DATOS PARA

PROYECTOS DE FORMACION

PROFESIONAL DUAL



El proyecto contiene un script de la base de datos de un proyecto de formación en modalidad dual con centro de trabajo. Los movimientos registrados deben de ser sobre los alumnos admitidos y sus correspondientes empresas asignadas.

Manuela Mena Gonzalez
Jose Muñoz Muñoz
Alvaro Limon Flor
Alejandro De la Rosa Cosano
Carlos Fernandez Gonzalez

CREACIÓN DE TABLAS

Una base de datos es un conjunto de datos relacionados entre si y almacenados sistemáticamente (agrupados o estructurados de cierta forma) para facilitar su posterior uso.
Nos permiten guardar grandes cantidades de información de forma organizada, para que luego podamos encontrarla y utilizarla más fácilmente.

Qué problema resuelven las bases de datos

Tenemos datos que necesitamos almacenar, que pueden ser de diferentes tipos (como datos de empresa, información sobre sobre asignaturas, alumnado, fechas de nacimiento, notas, etc). Pero el problema no pasa por tener datos en si, este no es un motivo suficiente para necesitar usar una base de datos. Los problemas que enfrentamos son los siguientes:

- Tamaño: la cantidad de datos que tenemos que manejar puede ir creciendo con el tiempo.

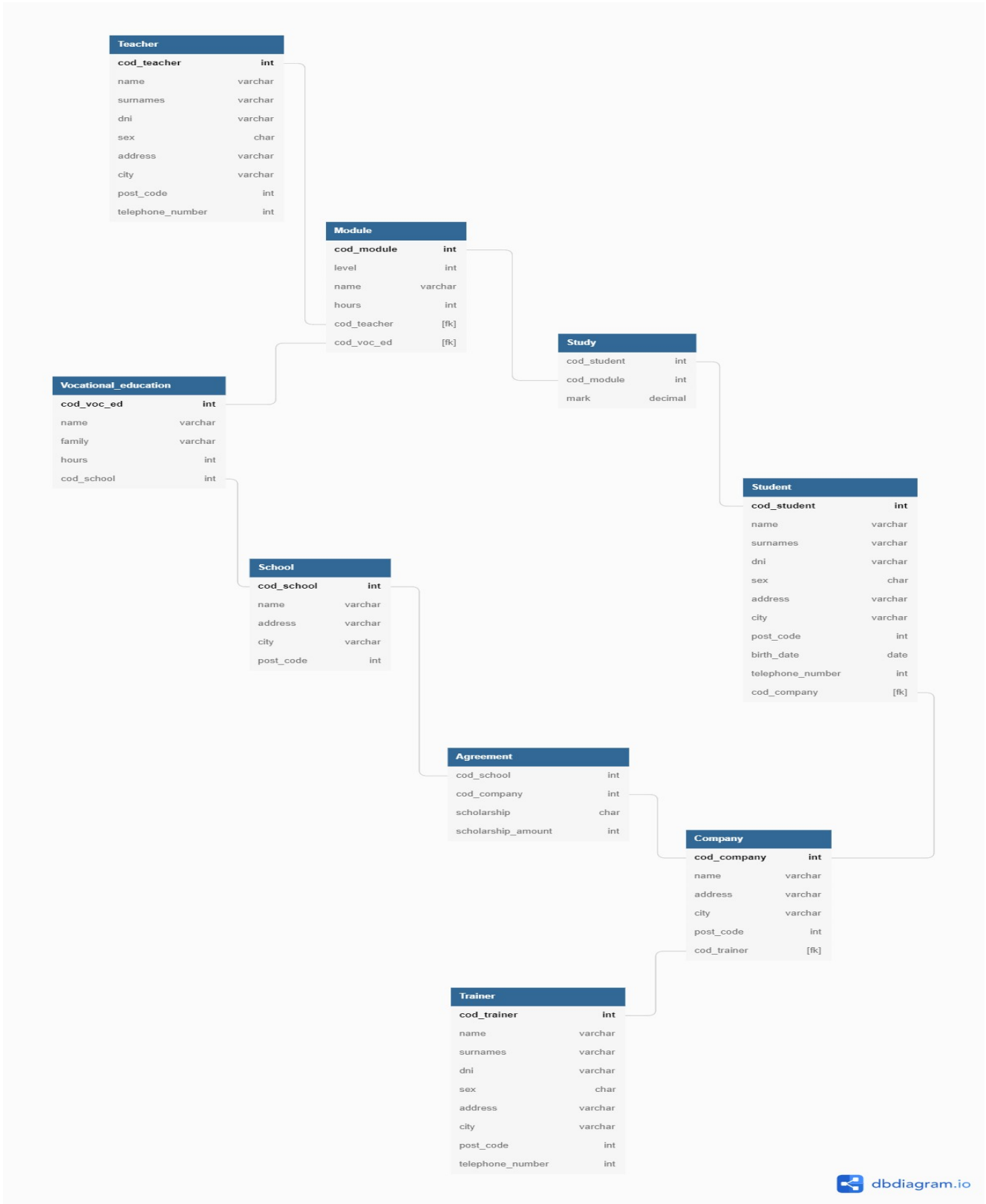
- Facilidad de actualización de los datos: qué pasaría si, por ejemplo, 2 personas quieren editar datos a la vez?

- Precisión: hay algo que prevenga que ingresemos datos con el formato incorrecto?

- Seguridad: quién puede acceder a estos datos? cómo controlamos el acceso y diferentes permisos o privilegios?

- Redundancia: la duplicación de datos genera conflictos, cómo sabemos cuál es el correcto?

- Importancia: no queremos perder datos ni el trabajo realizado frente a imprevistos .


Teacher
cod_teacher int
name varchar
surnames varchar
dni varchar
sex char
address varchar
city varchar
post_code int
telephone_number int
Module
cod_module int
level int
name varchar
hours int
cod_teacher [fk]
cod_voc_ed [fk]
Study
cod_student int
cod_module int
mark decimal
Vocational_education
cod_voc_ed int
name varchar
family varchar
hours int
cod_school int
Student
cod_student int
name varchar
surnames varchar
dni varchar
sex char
address varchar
city varchar
post_code int
birth_date date
telephone_number int
cod_company [fk]
School
cod_school int
name varchar
address varchar
city varchar
post_code int
Agreement
cod_school int
cod_company int
scholarship char
scholarship_amount int
Company
cod_company int
name varchar
address varchar
city varchar
post_code int
cod_trainer [fk]
Trainer
cod_trainer int
name varchar
surnames varchar
dni varchar
sex char
address varchar
city varchar
post_code int
telephone_number int
dbdiagram.io

INSERCIÓN DE DATOS

Instrucción INSERT INTO junto con su sintaxis, ejemplos y casos de uso.

- Creación de tabla SQL inmediatamente mientras inserta registros con los tipos de datos apropiados.
- Uso de INSERT INTO para insertar registros en un grupo de archivos particular.

Una vez que insertemos los datos en la tabla, podremos usar la siguiente sintaxis para nuestra instrucción INSERT INTO.

```
--Inserciones para la tabla Formador (Trainer)
INSERT INTO Trainer (name,surnames,dni,sex,address,city,post_code,telephone_number)
VALUES ('Javier', 'Diaz Fernandez', '54678654F', 'H', 'C. Aguilas C-1 5 P-2', 'Sevilla', 41004, 423546186);
INSERT INTO Trainer (name,surnames,dni,sex,address,city,post_code,telephone_number)
VALUES ('Sara','Lopez Garcia','54676945S','M', 'C.Pirineos F-7 3 P-3', 'Sevilla', 41005, 453256764);
INSERT INTO Trainer (name,surnames,dni,sex,address,city,post_code,telephone_number)
VALUES ('Miguel', 'Reyes Ortiz', '45327694F', 'H', 'C. Rosalia G-2', 'Malaga', 29010, 654532836);
INSERT INTO Trainer (name,surnames,dni,sex,address,city,post_code,telephone_number)
VALUES ('Luis', 'Rivera Martinez', '37465498Y','H', 'C.Rafaela R-3 6 P-3', 'Malaga', 29009, 546768523);
INSERT INTO Trainer (name,surnames,dni,sex,address,city,post_code,telephone_number)
VALUES ('Paula', 'Perez Barroso', '34768793R', 'M', 'C. Asturias E-3 2 P-1', 'Sevilla', 41010, 455672129);
```

Si hemos especificado todos los valores de la referida columna según las órdenes de la misma columna de la tabla, entonces no necesitamos especificar nombres de columna. En consecuencia, Podemos insertar registros directamente en la tabla.

No podremos insertar datos sin especificar los nombres de las columnas si en el orden de las columnas se hallan diferencias y hay una falta de coincidencia entre la inserción de datos y el orden de los valores de las columnas es diferente. Entonces Podemos obtener el siguiente mensaje de error.

```
--Inserciones para la tabla Empresa (Company)
INSERT INTO Company (name,address,city,post_code,cod_trainer)
VALUES ('NTT DATA', 'C. Americo Vespucio, 5 - 2C', 'Sevilla', 41092, 1);
INSERT INTO Company (name,address,city,post_code,cod_trainer)
VALUES ('Ricoh', 'C. Esclusa 11', 'Sevilla', 41011, 2);
INSERT INTO Company (name,address,city,post_code,cod_trainer)
VALUES ('NTT DATA', 'C. Larios', 'Malaga', 29005, 3);
INSERT INTO Company (name,address,city,post_code,cod_trainer)
VALUES ('Accenture', 'C. Severo Ochoa 6', 'Malaga',29590, 4);
INSERT INTO Company (name,address,city,post_code,cod_trainer)
VALUES ('Tier1', 'C. Boabdil Vega 7', 'Sevilla', 41900, 5);
```

CONSULTAS

En una instrucción SQL, la cláusula **WHERE** especifica criterios que tienen que cumplir los valores de campo para que los registros que contienen los valores se incluyan en los resultados de la consulta.

Like es un tipo de operador lógico que se usa para poder determinar si una cadena de caracteres específica coincide con un patrón específico. Se utiliza normalmente en una sentencia Where para buscar un patrón específico de una columna.

Se puede especificar el orden en que se desean recuperar los registros de las tablas mediante la claúsula **ORDER BY** Lista de Campos. En donde Lista de campos representa los campos a ordenar.

Con **GROUP BY** determinamos la agrupación por una o varias columnas con un mismo valor en la tabla consultada.

```
--Consultas Simples:

select name, surnames from student where sex like 'H' and city like 'Sevilla';
select address from student where surnames like 'Ruiz Garcia';
select count(name) from trainer;
select round(avg(hours),2) from module where cod_teacher = 10;
select name, surnames from teacher where cod_teacher in (1,4,10);
select name, surnames from teacher where cod_teacher not in (1,4,10);
select name, surnames from student where cod_student = 2 or cod_student = 3;


--Consultas Medias:

select name,surnames from student where sex like 'H' order by name;
select cod_school,name from school where cod_school in (select cod_school from agreement where scholarship = 'N') group by cod_school;
select scholarship_amount, cod_school from agreement where scholarship_amount >= 100 and scholarship_amount <= 300 order by cod_school;
```

INNER JOIN nos permite extraer las intersecciones en registros de distintas tablas de forma eficiente. Si deseamos obtener resultados idénticos de dos tablas o tres según sea el caso. Esta cláusula busca coincidencias entre 2 tablas, en función a una columna que tienen en común. De tal modo que sólo la intersección se mostrará en los resultados.

Con `LEFT JOIN` damos prioridad a la tabla de la izquierda, y buscamos en la tabla derecha.

En el caso de `RIGHT JOIN` la situación es muy similar, pero aquí se da prioridad a la tabla de la derecha.

Las palabras clave DESC y ASC se usan juntas junto con la instrucción SELECT y la cláusula ORDER BY.

```sql
--Consulta con INNER JOIN entre tres tablas
SELECT S.NAME AS 'Nombre del alumno',S.SURNAMES AS 'Apellidos',M.NAME AS 'Asignatura',ST.MARK AS 'Nota' FROM STUDENT S INNER JOIN STUDY ST ON
S.COD_STUDENT=ST.COD_STUDENT INNER JOIN MODULE M ON M.COD_MODULE=ST.COD_MODULE WHERE S.NAME LIKE 'a%o' ORDER BY ST.MARK DESC;

--Consulta con LEFT JOIN entre dos tablas
SELECT T.NAME AS 'Nombre del formador',T.SURNAMES AS 'Apellidos',C.NAME AS 'Empresa' FROM TRAINER T LEFT JOIN COMPANY C ON T.COD_TRAINER=C.COD_TRAINER
WHERE C.NAME LIKE 'NTT DATA' ORDER BY C.POST_CODE ASC;

--Consulta con INNER JOIN entre cuatro tablas
SELECT SC.NAME AS 'Centro',SC.POST_CODE AS 'Codigo postal',V.NAME AS 'Ciclo',V.HOURS AS 'Horas',M.NAME AS 'Modulo',M.HOURS
AS 'Horas',CONCAT(T.NAME,' ',T.SURNAMES) AS 'Profesor'
,T.DNI AS 'Dni',T.TELEPHONE_NUMBER AS 'Telefono' FROM SCHOOL SC INNER JOIN VOCATIONAL_EDUCATION V
ON SC.COD_SCHOOL=V.COD_SCHOOL INNER JOIN MODULE M ON
V.COD_VOC_ED=M.COD_VOC_ED INNER JOIN TEACHER T ON M.COD_TEACHER=T.COD_TEACHER WHERE T.SURNAMES LIKE 'M%' ORDER BY M.HOURS DESC;

--Consulta con RIGHT JOIN E INNER JOIN entre tres tablas
SELECT CONCAT(S.NAME,' ',S.SURNAMES) AS 'Alumno',S.DNI AS 'Dni',S.TELEPHONE_NUMBER AS 'Telefono',C.NAME AS 'Empresa',S.CITY
AS 'Ciudad',C.ADDRESS AS 'Direccion',A.SCHOLARSHIP AS 'Beca',
A.SCHOLARSHIP_AMOUNT AS 'Cantidad' FROM STUDENT S INNER JOIN COMPANY C
ON S.COD_COMPANY=C.COD_COMPANY RIGHT JOIN
AGREEMENT A ON C.COD_COMPANY=A.COD_COMPANY WHERE C.NAME='NTT DATA' OR A.SCHOLARSHIP='N'
ORDER BY A.SCHOLARSHIP_AMOUNT ASC;
```