Base de datos: colección de datos y metadatos relacionados entre si con una mínima redundancia(representación del mundo real), son independientes de las apps que usan la base de datos y están regidas por reglas de negocio.

Modelo de datos: instrumento que se aplica a una parte del mundo real para obtener una estructura de datos denominada esquema.
-Conceptuales: como los usuarios perciben los datos.
-Lógicos: como lo entienden los usuarios finales pero lejos del lenguaje de máquina.
-Físicos: como almacenan los datos una máquina.

Conceptos:

-Entidad: Representación de un objeto o concepto del mundo real, con unos atributos propios.
-Entidad tipo: conjunto de entidades del mismo tipo.

-Atributo: indica una propiedad de la entidad o un dato de interés de este. Tipos:
+Simples +Compuestos(Más de un atributo)
+Univaluados +Multivaluados(Doble circulo)
+Almacenado(atributo normal) +Calculado (se obtiene a partir de otro atributo)
+Obligatorio (No puede ser Nulo) +Opcional (Puede ser Nulo)
-Dominio de un atributo: Son los valores que puede tomar este.

-Atributos clave: atributo o atributos que forman una combinación para cada entidad.
-Claves Candidatas: todas aquellas súper claves con un número mínimo de atributos.
-Clave Primaria: clave candidata escogida para identificar a la entidad.
-Clave Alternativa: aquellas claves que no han sido escogidas para ser la primaria.

-Rol: indica que papel tiene cada entidad en la relación.
+Grado de una relación: define el numero de entidades que participan en la relación. NO NECESARIO.

+Cardinalidad: define el n.º mínimo y el máximo de veces que la entidad participa en la relación.
    +Simplificada: Especifica solo el n.º máximo de entidades que participan en la relación.
    +Detellada: Refleja tanto el mínimo como el máximo.

+Atributo de una relación: son propias de la relación entre las entidades.
+Relación recursiva: una relación entre dos entidades del mismo tipo.

+Participación:
    +Total : la participación mínima es 1. Representación doble linea.
    +Parcial: la participación mínima es 0. Representación línea normal.

-Tipos de relaciones:
+Débiles: cuando hay una relación entre una entidad fuerte y una débil.
    +Débil por existencia: la entidad débil depende de la otra para existir. Si se borra la fuerte la débil también.

+Débil por identificación: implica que ya es débil por existencia, pero a parte necesita la clave de la entidad fuerte para poder ser identificada. En estos casos la clave principal de la débil es la clave de la fuerte más algún atributo de la débil, y siempre se propagan juntas.

Que hacer con las relaciones en el modelo relacional:

-En una relación 1:1 donde sus cardinalidades detalladas son ambas (1,1) pasas la clave a la entidad que quieras y si la relación tiene atributos estos siguen a la clave. En caso que una tenga una cardinalidad (0,1) se pasa a esta, para evitar nulos.
-Caso 1:1 en el que ambas cardinalidades sean (0,1), se crea una tabla nueva, solo una de las claves es principal y la otra alternativa.
-Caso N:M se crea una nueva tabla y ambas CP pasan a formar la CP de la nueva tabla, si la relación tiene atributos y son principales(necesarios para formar clave) la clave principal pasa a ser el atributo más las otras claves.
-Caso 1:N la clave del 1 pasa a N, y los atributos si tiene la siguen. En caso de que la clave que oasa sea (0,1) pasa como opcional (No se indica con asterisco, puede ser nula)
-Recursivas: igual que en caso 1:N.

Modelo Relacional:
LOS ATRIBUTOS MULTIVALUADOS NO SE PASAN

Para los atributos multivaluados se crea una nueva tabla, con el atributo multivaluado y la CP de la entidad a la que pertenece.
Restricciones:
-Restricción de integridad de entidad: los atributos de la clave primaria no pueden ser nulos.
-Restricción de integridad referencial: los valores de la clave foránea deben existir en la tabla de la que procede o ser nulos.
-Restricción de verificación: debe cumplir alguna condición (Check).

Borrados o Modificados:
-Cascada, en casi de ser modificados los valores de la CP , se modifican de la misma forma los hijos.
-Restricted: en casi de ser modificados , no deja llevar ningún cambio en los hijos.
-Set Null, en este caso sustituye los valores por un NULL, siempre que sea permitido.
-Set Default, en este caso los valores son sustituidos por un default.

Modelo Relacional Extendido:

-Especialización: Se definen subclases de una superclase. Es necesario que las subclases tengan atributos o relaciones propias.
    +Disjuntas: la superclase solo se puede ver reflejada en una de las subclases a la vez.
    +Overlay: la superclase puede ser al mismo tiempo más de una subclase.
    +Total: la superclase se tiene que ver reflejada en alguna de las subclases.
    +Parcial: la superclase no tiene pq ser miembro de una de las subclases.

-Generalización: Es el proceso contrario.

Opciones a la hora de pasar al Relacional, A,B,C y D

-Caso A Tabla principal y las subtablas

-Caso B , participación total y disjunta. Si es overlay habria redundancia. Si tiene relaciones o hereda de otras tablas no.
-Caso C una sola tabla con todo. Disjunta, total.
-Cado D overlay, tipo (S/N)

Sentencias DDL: aquellas que se usan para la creación de una base de datos y sus componentes.

-Sentencia Create:

CREATE DATABASE nombrebase; -Asi se crea la Base de datos
use nombre base; -Asi se usa

comando combinando ambas:
CREATE DATABASE nombrebase;
go
use nombre base;
El go se emplea para realizar primero la creación y después usar la tabla.

Ejemplo de crear tabla:

```
CREATE TABLE nombretabla(
        atributo            tipo de atributo,
        atributo            tipo de atributo,
        CONSTRAINS PK,
        CONSTRAINS FK,
);
```

Indicar que es una PK se puede indicar en la definición del propio atributo o a pie de tabla.
Al crear el atributo es necesario especificar al lado si puede tomar valores nulos o no, en el caso de ser PK no es necesario pq ya es implícito. El saber si es nulo o no nos lo indica el relacional.

```
CREATE TABLE nombretabla(
        atributo(CP)        tipo de atributo,
        atributo            tipo de atributo NOT NULL,

);
```

UNIQUE: la columna solo permitirá valores únicos.
IDENTITY: crea una columna autoincrementable. El tipo de dato tiene que ser numérico.
DEFAULT: un valor en caso de que la columna este vacía o sea borrada.

IMPORTANTE: A la hora de crear las tablas crear primero las que no hereden ninguna fk ya que puede haber errores al hacer referencia a tablas que aun no existen.

-**CONCAT**, se usa al querer juntar dos campos en uno:

```
CREATE TABLE NombreTabla (
        NombreCompleto AS CONCAT(Nome, Apellidos),
);
```

-**DEFAULT,** no es necesario definirlo como constraint pero es más cómodo ponerle nombre para poder referenciarlo.

```
CREATE TABLE NombreTabla (
        CIF     CHAR(9)     NOT NULL  CONSTRAINT NombreConstraint default 'valorDefault' ,
);
```

-Campo calculado

```
CREATE TABLE NombreTabla (
        NombreCampoFinal AS (Soldo - (Soldo*Desconto/100)),
        );
```

Declaras el nombre que deseas y especificas que campos se usan para obtener el nuevo campo.

-**CHECK**: restricción que usar para pedir datos específicos o que cumplan lo establecido.

```
CREATE TABLE NombreTabla (
        Desconto    FLOAT NOT NULL CONSTRAINT NombreConstraint CHECK(Desconto <= 25),
        );
```

Tipos de datos:

-Numéricos:

+Int(-2^31 a 2^31) +Smallint(-32768 a 32.767) +Tinyint(0 a 255)

Según como de grande sea el número que quieres guardar.

Decimal(Precisión número total de digitos tanto decimales como no, Escala número de digitos decimales)

Numeric es lo mismo.

+Money +Smallmoney

Se suele usar smallmoney salvo cantidades muy grandes.

+Float +Real

Cadenas de hora y fecha

+Time(hora) +Date(año/mes/día) +Smalldatetime(date y time)

+Datetime(date y time pero con centésimas) +Datetime2 (Más completo que datetime)

Cadenas de caracteres:

+Char(un número fijo de caracteres) +Varchar(No es un número fijo pero se sabe el máximo)

+Bit(campos S/N) tmb se puede usar boolean

-INSERT:

A la hora de introducir los datos si no se especifican todos los campos es necesario introducir valores por cada uno de los campos.

```
INSERT INTO nombretabla (campo, campo)
VALUES (datos, datos);
```

Posibles escenarios, si tienes un campo con un default y al introducir los datos no lo pones se rellena solo, si no tiene default se rellenaria con un NULL.
Si es un campo en que esta especificado NOT NULL o es PK es obligatorio introducir un dato.
Al introducir un dato si es una FK tiene que existir en la otra tabla Integridad Referencial.
En caso de querer modificar una PK que existe en otra tabla como FK es necesario que el update sea en cascada.

DML: utilizadas para la gestión de datos.

-DELETE: para eliminar uno o mas campos de una tabla

```
DELETE valor
FROM nombretabla
WHERE condiciones;
```

En el delete podemos especificar TOP(numero) que eliminara las x filas superiores.
TOP(Numero) PERCENT : eliminaria el x % de las filas superiores.
Sin el WHERE elimina toda la tabla.

```
DELETE TOP(25) PERCENT
FROM PRODUCTOS
WHERE PRECIO = 25 AND DESCRIPCION ='ROJO';
```

EXTRA, en caso de querer eliminar todas las filas se usa TRUNCATE TABLE. Más rápido y menos recursos.

-UPDATE: se usa para actualizar los campos de una tabla.

```
UPDATE nombretabla
        set columna1 = “nuevoDato”
            columna2 = “nuevoDato”
        ….
WHERE condiciones
```

```
UPDATE PRODUCTOS
        SET nombre = 'Camisola',
        PRECIO  = 60
 WHERE NOMBRE = 'Camisa';
```

-SELECT:Consultar tablas o datos existentes

```
SELECT *
FROM nombretabla
```

Da todos los datos de la tabla

```
SELECT Campo,
        Campo,
FROM nombretabla;
```

-WHERE:se usa para filtrar los valores, especificando condiciones.

```
WHERE columna = datoExistente;
```

IN Comprobar que los datos existen

```
        WHERE Localidad IN ('Madrid', 'Barcelona');
        WHERE Localidad NOT IN ('Valencia', 'Alicante', 'Sevilla' );
```

BETWEEN: comprobar que un valor está en un rango

```
        WHERE PrecioActual NOT BETWEEN 200 AND 400;
```

OPERADORES:

=
Igual

<>
Distinto

!=
Distinto

>:
Mayor que

>=:
Mayor o igual que

<:
Menor que

<=:
Menor o igual que

IN ( ):
Coincide con un valor en una lista dada

NOT:
Niega la condición

BETWEEN:
Dentro de un rango

IS NULL:
es NULL

IS NOT NULL:

No es NULL

LIKE:
Coincide con un patrón (Se pueden usar comodines % _)

EXISTS:
La condición se cumple si la subconsulta devuelve al menos una fila

AND:
TRUE si ambas expresiones son TRUE

OR:
TRUE si cualquiera de las dos expresiones booleanas es TRUE

SOME:
TEUE si alguna de las comparaciones de un conjunto es TRUE

ALL:
TRUE si el conjunto completo de de comparaciones es TRUE.

-LIKE:se usa para buscar una cadena específica.

% número cualquiera de caracteres
_ un único carácter
[] uno del intervalo o los valores
[^]no pertenece a ese conjunto.

EXTRAS, NO NECESARIO:
Si quieres cambiar el nombre de una columna sin tener que borrar toda la tabla:

```
EXEC sp_rename 'nombreTabla.columna', nuevaColumna', 'COLUMN';
```