T03. Modelo relacional. Normalización. Paso a tablas

3.1. Modelo Relacional

El modelo relacional busca una **solución** a los problemas derivados de la rigidez estructural de las bases de datos **jerárquicas**, presentando sólidos términos **matemáticos**.

El elemento básico del modelo relacional es la **relación** → estructura bidimensional que representa las **entidades**, las **filas** se llaman **tuplas**, que corresponden a las ocurrencias, cada **tupla** cuenta con una serie de **atributos** y cada uno de ellos tienen un **valor**.

Relación:

NIF	Nombre	**Apellidos**	Teléfono	Fech nacimiento	Fech alta
26012245Y	Luis	Pérez García	952121212	12/01/02	16/09/22
51244002B	Pedro	Lozano Ríos	956421242	25/05/00	05/07/22
78545125A	Ana	Gómez Saez	666182456	06/10/99	20/09/22
01251423Z	Patricia	Barrón López	636925522	23/08/85	27/08/22

Tupla:

01251423Z	Patricia	Barrón López	636925522	23/08/85	27/08/22

Atributo:

Apellidos

Valor:

Barrón López

Llamamos **dominio de un atributo** al **conjunto** de valores que pueden tomar un atributo para una **ocurrencia concreta**. Su idea típicamente se asocia con la de tipo de datos, en el caso del atributo el tipo de dato más apropiado es la cadena de caracteres.

Ej: Atributo (Nombre) → Seleccionamos cadena de caracteres porque el nombre está compuesto por caracteres.

Se identifican los siguientes **tipos** de **datos base**:

- **Números enteros.**
- **Números decimales.**
- **Cadenas de caracteres.**
- **Fechas y horas.**
- **Valores lógicos (verdadero o falso).**
- **Objetos (ficheros binarios).**

El valor **nulo** o **NULL** , representa la ausencia de valor, que no es lo mismo que vacío.

3.1.1 Condiciones que ha de cumplir una relación

1. Contener **un solo tipo de tupla**.
2. Dentro de una relación **no puede haber tuplas repetidas.**
3. Dentro de una relación **no puede haber atributos repetidos.**
4. **El orden de las tuplas** en una tabla **no está determinado.**
5. **El orden de los atributos** en una tabla **no está determinado.**

3.1.2 Integridad de las relaciones

Para **conectar** dos relaciones utilizamos la **Clave ajena**, se puede definir de la siguiente forma:

En **R1** y **R2** se define **Clave ajena** como **un atributo de R2**…

- Sus valores son del **mismo dominio** que los valores **de la clave primaria de R1.**
- La clave ajena puede ser **nula** o **coincidir** con un valor de la clave primaria **de R1.**
- La clave ajena **no tiene que ser un componente** de la clave primaria de R2.
- La clave ajena **puede aceptar Null.**

3.1.3. Integridad referencial.

Cualquier dato que **no** cumpla las reglas de integridad referencial **será incorrecto**, en una operación cabe la posibilidad de que el sistema **rechace la operación** o que la acepte y realice otras necesarias, todo ello para mantener la base de datos en un **estado correcto.**

El **diseñador** de la base de datos es el encargado de esto.

Para mantener la integridad referencial debemos plantearle a la Clave Ajena:

1. **¿Puede tener nulos?** Dependerá del problema que se esté resolviendo.

2. **¿Qué debería suceder si se intenta borrar el objetivo de una referencia de clave ajena?**

    - **Restringida**: Sólo se permite borrar **si no tiene dependencia de otra.**

    - Se propaga que genera **acciones** propagándose en cascada.

    - Se asigna valor Null a todas las claves ajenas que tienen como objetivo la clave primaria de la otra relación borrada.

3. **¿Qué debería suceder si se intenta modificar el objetivo de una referencia de clave ajena?**

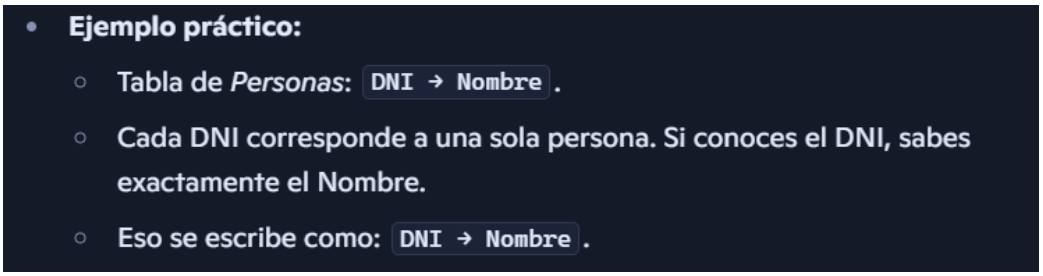
    - **Restringida:** Sólo se permite modificar **si no tiene dependencia de otra.**

    - Se propaga **(cascada)** La operación genera acciones de compensación propagándose en cascada.

    - Se asigna valor **Null** a todas las claves ajenas que tienen como objetivo la clave primaria de la otra relación modificada.

3.2. Normalización.

Dependencia funcional: Decimos que *Y depende funcionalmente de X* si cada valor de X determina un único valor de Y.



Dependencia funcional plena: Depende funcionalmente de todo el conjunto X (X1, X2,…), pero no depende de ningún subconjunto.

- **Ejemplo práctico:**
  - Tabla de *Notas*: `(Alumno, Asignatura) → Nota`.
  - La nota depende de la combinación de Alumno y Asignatura.
  - Pero **no depende solo de Alumno** (porque un alumno tiene varias asignaturas) ni solo de Asignatura (porque una asignatura la cursan varios alumnos).
  - Entonces decimos que la dependencia es **plena.**

1FN. Primera forma normal

Una relación está en **(1FN)** si todos sus valores son **atómicos**, es decir, cada valor es **único**, aquí vemos la **transformación** de una tabla a una en 1FN.

NIF	Nombre	Apellidos	Teléfono
26012245Y	Luis	Pérez García	952121212 646252325
51244002B	Pedro	Lozano Ríos	956421242 854484848 666002001
78545125A	Ana	Gómez Saez	666182456
01251423Z	Patricia	Barrón López	636925522

👉

NIF	Nombre	Apellidos	**Teléfono**
26012245Y	Luis	Pérez García	952121212
26012245Y	Luis	Pérez García	646252325
51244002B	Pedro	Lozano Ríos	956421242
51244002B	Pedro	Lozano Ríos	854484848
51244002B	Pedro	Lozano Ríos	666002001
78545125A	Ana	Gómez Saez	666182456
01251423Z	Patricia	Barrón López	636925522

2FN. Segunda forma normal

Para que una relación este en **(2FN)** debe cumplir esta **reglas**:

- Está en **1FN**.
- Todos los atributos que no forman parte de la clave primaria **dependen** de ella **por completo**.

Veamos un proceso de **1FN** a **2FN**.

Cód libro	**Cód tienda**	Cantidad	Dirección
215	9	5	C/ Blas de Lezo
710	7	3	C/ Mármoles
215	5	4	C/ Jaboneros
152	9	1	C/ Blas de Lezo

Vemos como la **dirección** se repite **dos veces** en el caso “C/ Blas de Lezo” y **Cod Tienda** “9”.

1.

Cód libro	**Cód tienda**	Cantidad
215	9	5
710	7	3
215	5	4
152	9	1

2.

Cód tienda	Dirección
9	C/ Blas de Lezo
7	C/ Mármoles
5	C/ Jaboneros

En este caso nos obligamos a **dividir la relación** en dos:

1. **Clave primaria completa** (**Cod libro**, **Cod tienda**) y solo los atributos que dependen **totalmente de ella**, es decir, tenemos el **libro** "x" (Cod libro), que se encuentra en la **tienda número** "x" (cod tienda), y tenemos una **cantidad de** "x" aquí (Cantidad).

2. **Parte de la clave primaria** (**Cod tienda**) de la que **depende algún** o **algunos atributos**, es decir, tenemos que la **tienda** "x" (Cod tienda) se encuentra en la **dirección "**x" (Dirección).

Al dividirse la tabla en dos distintas, **no repetimos la misma información dos veces** en el caso de Cod tienda y Dirección.

3FN. Tercera forma normal

Para que una relación este en **(3FN)** debe cumplir esta **reglas**:

- Está en **2FN**.

- Todos los atributos que no forman parte de la clave primaria son **independientes** entre sí, ninguno de ellos depende de otro, **excepto el campo clave.**

Cód Empleado	Nombre	Apellidos	Dirección	Cod Dpto del emplead.	Nombre Dpto	Fecha nac
25	Vicente	Carrera García	C/Lope Rueda, 23	4	Nóminas	20/07/98
18	Luisa	Vázquez Valero	C/Niquel 12	7	Contabilidad	10/04/01
75	Fernando	Casas Lobato	C/Plata 9	4	Nóminas	08/05/90
32	Elena	Troya Egea	C/Peso la harina, 5	3	Almacén	01/10/97

La clave primaria es **Cód Empleado**

Nombre, Apellidos, Dirección, Fecha nac → dependen directamente del **empleado**.
Cod Dpto del emplead. → también depende del **empleado**.
Nombre Dpto → depende de **Cod Dpto del emplead**. , no directamente del empleado

La solución es **separar** los **atributos** que dependen de otro atributo que no es clave primaria, en este caso, Nombre Dpto depende directamente de Cod Dpto del emplead, es decir, lo podríamos **omitir** en la tabla porque sabiendo el cod Dpto del empleado sabremos su nombre.

Cód Empleado	Nombre	Apellidos	Dirección	Cod Dpto empleado	Fecha nac
25	Vicente	Carrera García	C/Lope Rueda, 23	4	20/07/98
18	Luisa	Vázquez Valero	C/Niquel 12	7	10/04/01
75	Fernando	Casas Lobato	C/Plata 9	4	08/05/90
32	Elena	Troya Egea	C/Peso de la harina, 5	3	01/10/97

Ahora creamos **otra tabla** con el atributo que dependía de este:

Cod Dpto	Nombre Dpto
4	Nóminas
7	Contabilidad
3	Almacén

Así, cada atributo depende **únicamente de la clave primaria de su tabla**, y se rompe la **transitividad**.

3.3. Paso del DER al modelo Físico de Datos

El objetivo del Diagrama Entidad Relación y las técnicas de normalización es proporcionar el mejor diseño posible para una futura base de datos.

Diagrama Entidad-Relación	Entidad o Relación	Ocurrencia	Atributo
Modelo relacional	Relación	Tupla	Atributo
Modelo Físico de Datos	Tabla	Registro	campo

3.3.1. Nomenclatura

- Nombre de **tablas** → Se prefijan con un "T" (**T**Empleado, **T**Cliente).

- Nombre de **campos**:

  - **Números:** "n" (**n**Cantidad).
  - **Cadenas de caracteres:** "c" (**c**Nombre).
  - **Fechas:** "d" (**d**Nacimiento).
  - **Valores lógicos:** "l" (**l**sexo).
  - **Objetos:** "o" (**o**Foto, **o**DocumentoXML).

- Nombres (**genérico**) → Se evita el uso de "**ñ**" , "**Ç**" y los vocales **acentuadas** o con **diéresis.** Solo se acepta:

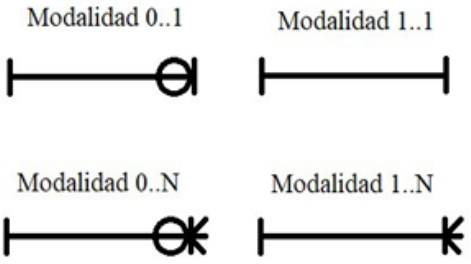
  - Letras del alfabeto inglés.
  - Números.
  - Guión bajo (_).

- **Identificadores** → Todo campo creado para identificar de forma unívoca los registros de una tabla llevará como sufijo las letras " ID" (nCliente**ID**, nLibro**ID**, nPais**ID**).

3.3.2. Reglas de transformación

- **Las tablas** se representan como un rectángulo con el nombre de la tabla en en la parte superior, los campos que conformen la clave primaria irán subrayados y en orden.

- **La modalidad**.



En algunos casos, las relaciones generan tablas, para esto seguiremos estos criterios:

- Utilizar un **sustantivo** que represente la acción implícita en la relación ("**usuario compra producto**" generaría **TCompra**).

- **Concatenar** los **nombres** de las entidades relacionadas ("**profesor imparte asignatura**" generaría **TProfesorAsignatura**).

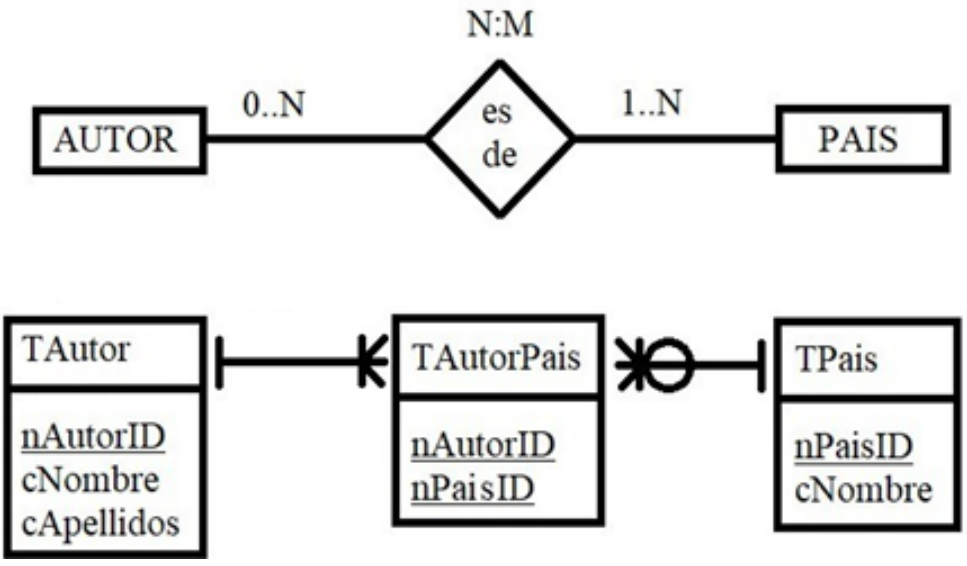
La transformación de **componentes** del diagrama Entidad-Relación en **elementos** del **modelo físico** de datos sigue las siguientes reglas:

1. Toda **entidad** se convierte en una **tabla**.

2. Todo **atributo** pasa a ser un **campo**, los atributos clave primaria se convierten en campos de la clave primaria de la nueva tabla.

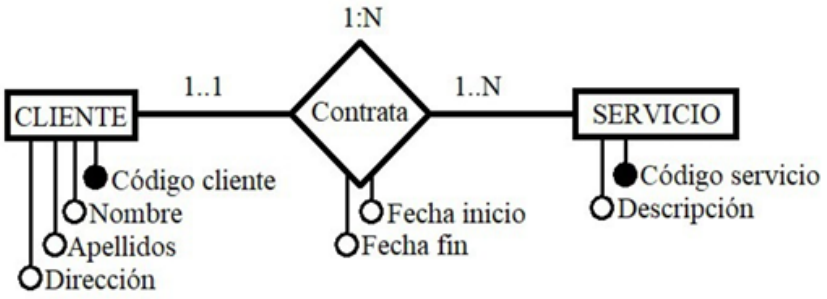
3. Las **relaciones** se representan basadas en su **cardinalidad**:

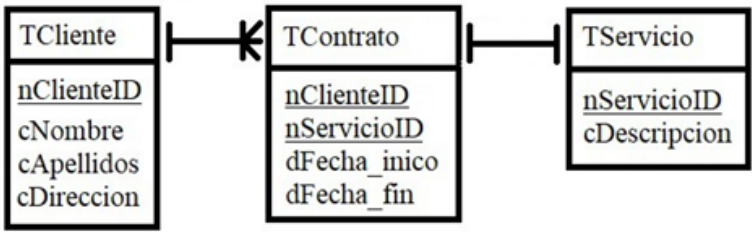
    - **Relaciones M:N.** Se convierten en tabla. Su clave primaria será la concatenación de las claves primarias de las entidades que relacionan.



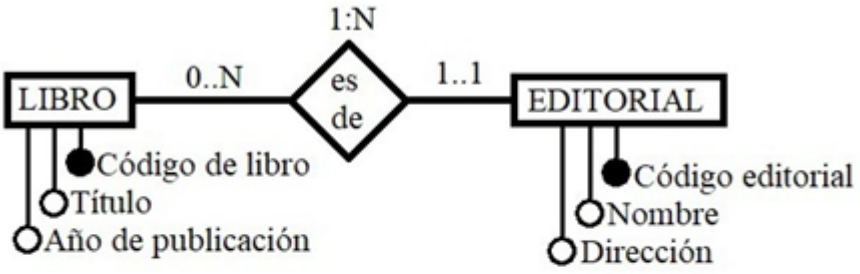
- Relación **1:N.** Se presentan 3 posibilidades:

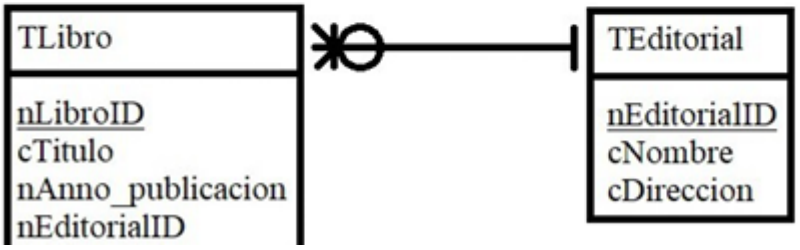
    1. Si la relación tiene **algún atributo**, se convierte en tabla:



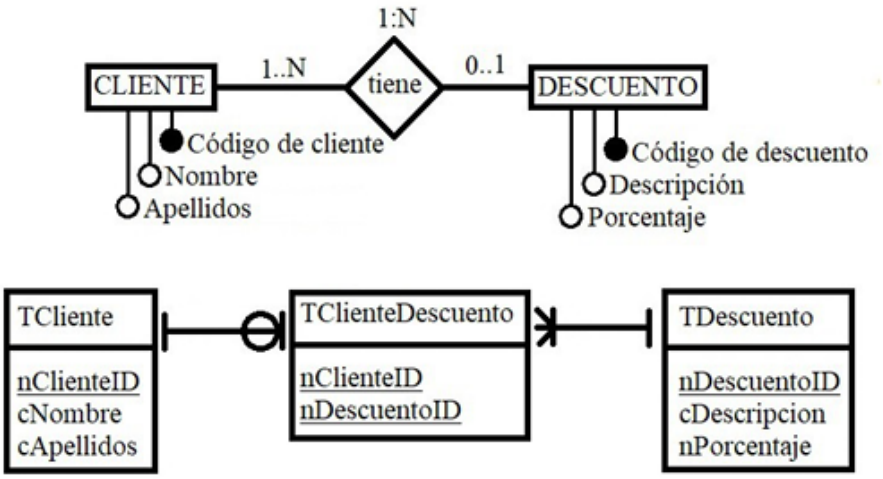


    2. Si la modalidad mínima en el lado de cardinalidad **1 es 1** y no hay atributos, la relación desaparece.

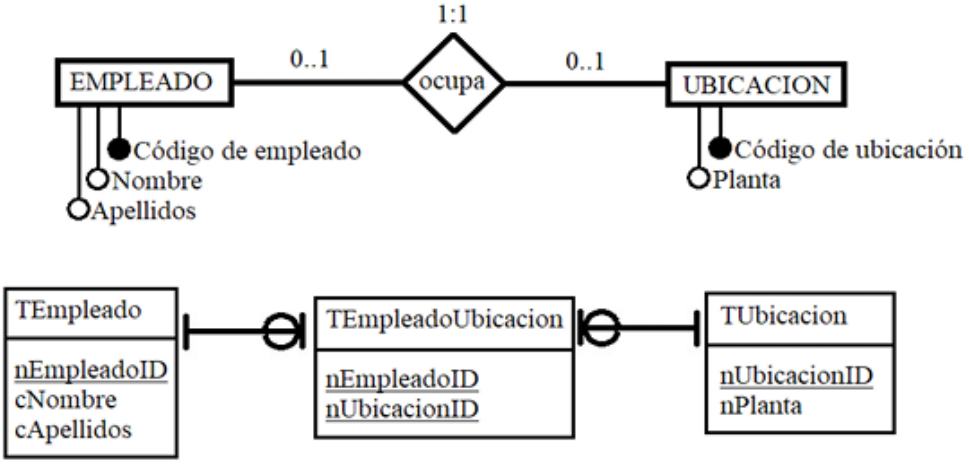




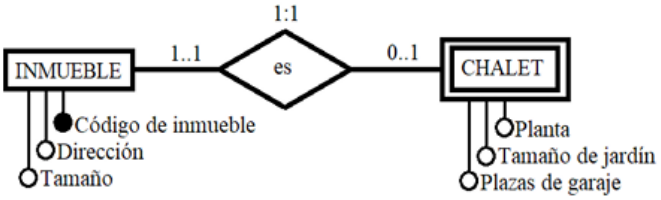
3, Si la modalidad mínima en el lado de cardinalidad **1** es **0..1,** la relación se convierte en tabla.

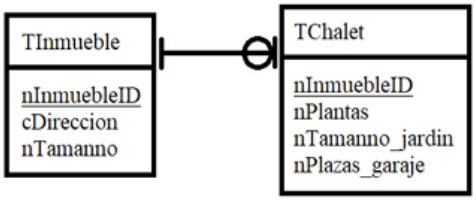


- **Relaciones 1:1.** El objetivo es **evitar** a toda costa que queden **campos sin valor.**
    1. Si **ambas** modalidades son **0..1,** la relación genera una tabla.

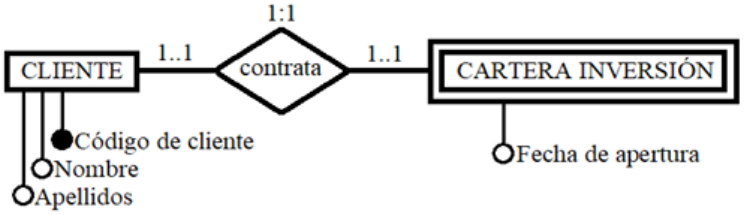


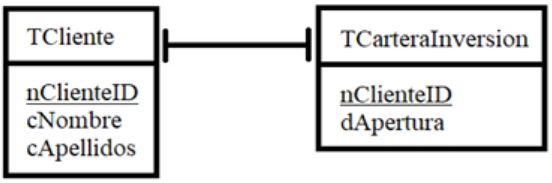
2. Si una modalidad es **0..1** y la otra **1..1,** la clave de la entidad con **modalidad mínima 1 se propaga a la entidad con modalidad mínima 0.**





3. Si **ambas** modalidades son **1..1**, pero una de las entidades es débil, la clave de la entidad **fuerte se propaga a la débil.**





3.4. Otras consideraciones

3.4.1. Índices

Como **mínimo** habrá un fichero de **índice** que relaciones cada registro con su valor correspondiente de clave primaria, la tarea del diseñador es definir los índices que vea **necesarios**.

Es necesario señalar la **importancia** del **orden** de campos en la definición de un índice (no es lo mismo ordenar por cApellidos + cNombre que por cNombre + cApellidos). Además este ordenamiento de información **ayuda** a buscar valores con **más rapidez.**

A pesar de todas sus ventajas, la existencia de índices **crea redundancia** y **ralentiza** los procesos de **inserción**, **actualización** y **borrado**. Por ese motivo es importante **no abusar** de su número ni de su tamaño

3.4.2. Vistas

Una **vista** es una **consulta** sobre una o varias tablas almacenadas en la base de datos. El uso de vistas **incrementa el nivel de seguridad** en el acceso a los datos y son **más eficientes** que las consultas efectuadas puntualmente, al encontrarse ya definidas en la base de datos.

Una vista puede obtener una lista de clientes, o una relación de facturas cuyo importe esté **dentro de un rango**, pero **no puede filtrar sus valores** por un dato variable.

3.4.3. Restricciones sobre campos

- **UNIQUE**. Todos los valores de ese campo deben ser **únicos**, y **no** se permiten valores **repetidos**. Cuando una clave primaria está compuesta por un solo campo, dicho campo está obligado a cumplir la restricción UNIQUE.

- **NOT NULL. Impide** que un campo pueda tener **valores nulos.** Obligado en los campos que forman la clave primaria.

- **DEFAULT.** Si no se especifica un valor **se asignará el valor por defecto**, que **no será NULL.**

3.4.5. Acceso concurrente

En la operativa de la mayoría de los sistemas de información ocurre que **más de un empleado** trabaja con los **mismos datos.** Esta situación ha de estar controlada para evitar problemas. ¿Como? con las **políticas de bloqueo.**

3.4.6. Políticas de bloqueo

Un bloqueo no es más que cuando un usuario accede a unos datos se **bloquea el acceso** de otros hasta que el primero no **termine** y **libere la información.** Puede generar problemas de lentitud del sistema incluso bloqueos. Por lo que es muy **importante** hacer un **buen diseño.**

- **Compartido.** Se permite que otros usuarios **puedan acceder** a los datos bloqueados, **pero no modificarlos.**

- **Exclusivo. Solo** el usuario que ha bloqueado la información puede **consultarla** y **modificarla.**