BASES DE DATOS

Carlos Gorria[1]

[1]Departamento de Matemáticas (UPV/EHU)

Leioa, curso 2020/21

1. Introducción. Modelos de bases de datos.
   - Concepto y aplicaciones de las bases de datos. Ejemplos.
   - Errores clásicos en el almacenamiento de datos.
   - Arquitecturas clásicas de bases de datos.
2. Diagramas entidad/interrelación.
   - Elementos de los diagramas entidad/interrelación.
   - Transformación del diagrama en tablas.
3. Normalización y depuración de esquemas relacionales.
   - Restricciones de integridad y de usuario.
   - Normalización y depuración de tablas.
4. Reunificación de la información.
   - Reunificación mediante álgebra relacional y de conjuntos.
5. El problema de la concurrencia.
   - Introducción a las transacciones.
   - Procesos multiusuario. Planificación de transacciones.
   - Concurrencia. Inconvenientes y soluciones. Bloqueos.

Concepto y aplicaciones de las bases de datos. Ejemplos.

Contexto:

- Actualmente los avances tecnológicos posibilitan la captación y almacenamiento masivo de datos asociados a procesos industriales, biológicos, económicos, sociales, etc.
- La utilización de datos en el diseño y validación de modelos facilita un conocimiento del medio y una eficaz herramienta de predicción en sistemas de cierta complejidad.
- Se han desarrollado numerosas técnicas estadísticas para analizar conjuntos de datos dependiendo de su estructura.
- Existe gran demanda de profesionales con conocimientos de matemáticas, estadística, programación y gestión de datos.
- El dominio de software de gestión de datos y del álgebra de conjuntos y relacional son necesarios para optimizar la extracción de información a partir de esos datos.

Concepto y aplicaciones de las bases de datos. Ejemplos.

Objetivos de las bases de datos:

- Describir los estados asociados a una realidad mediante el valor asignado a unos atributos distinguibles o cuantificables. Determinar los rangos de estos parámetros enteros, reales, lógicos o categóricos.
- Disponer de un procedimiento para almacenar valores de manera sencilla, accesible y flexible.
- Incorporar mecanismos para restringir el almacenamiento de valores que representen estados inadmisibles.
- Disponer de una estructura que relacione objetos de diferente tipo pero que tengan cierta asociación.
- Disponer de software eficiente para gestionar el conjunto de datos y que permita operaciones de actualización, búsqueda, ordenación, filtrado o concatenación.

Concepto y aplicaciones de las bases de datos. Ejemplos.

Representación de una bases de datos:

Los conjuntos de datos se identifican mediante arrays (listas) e intuitivamente se asocian con tablas bidimensionales.
Una tabla almacena información que caracteriza cada uno de los objetos de un conjunto que comparten patrón o estructura. Cada columna determina una propiedad o campo y cada fila a un objeto.
En una tabla $M \in \mathcal{A}_{n\times m}$ de n filas y m columnas, el componente a_{ij} almacena el valor del campo j para el objeto i.

Ejemplo de tabla de datos:

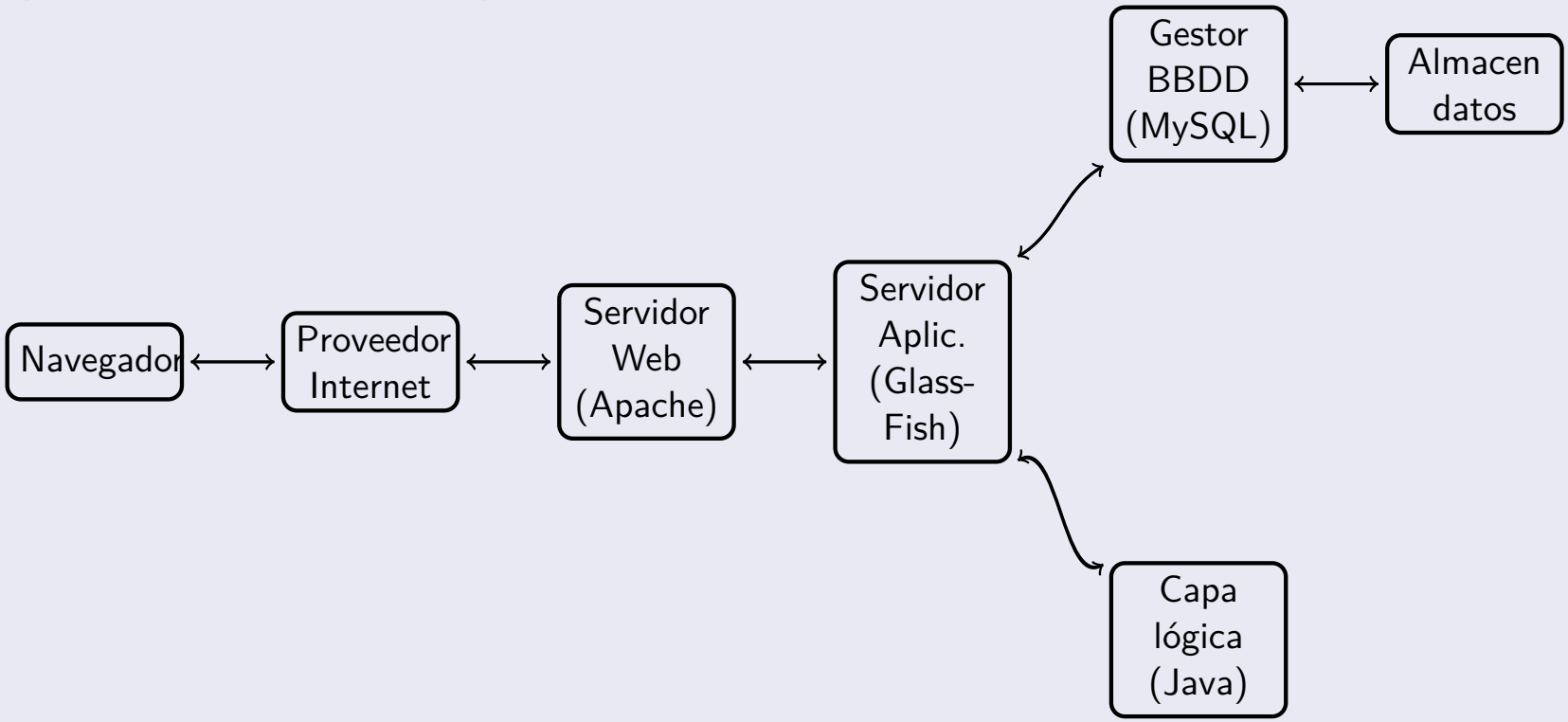
Tornilleria

Ref	Material	Longitud	Calibre
tt3003	titanio	30	3
fe6005	acero	60	5
al5004	aluminio	50	4

Concepto y aplicaciones de las bases de datos. Ejemplos.

Acceso a las bases de datos:

El acceso para modificaciones y consultas a la base de datos se hace a traves de un gestor como MySQL, Oracle, PostgreSQL, etc, que es invocado desde un navegador a través de un servidor de aplicaciones como Apache o GlassFixh.



Concepto y aplicaciones de las bases de datos. Ejemplos.

Etapas del diseño de una base de datos:

1. Enumeración de objetos de interés, características parametrizables, interrelaciones y tipos de consultas.
2. Identificación de entidades, atributos e interrelaciones.
3. Diseño gráfico del esquema entidad/interrelación.
4. Elaboración de tablas que preserven la semántica y las restricciones de integridad y de usuario.
5. Detección de claves candidatas y asignación de claves primarias y foráneas.
6. Depuración de tablas mediante las formas normales de Codd.
7. Diseño de consultas y restricciones en el gestor (MySQL).

Concepto y aplicaciones de las bases de datos. Ejemplos.

Tipo de datos que se almacenan:

- Números enteros o reales.
- Cadenas alfanuméricas.
- Variables lógicas o de tipo binario.
- Horas y fechas.
- Índices o claves identificativas.

Estructura de las tablas en una base de datos:

- Cada columna de una tabla bidimensional corresponde a un atributo o característica del tipo de objetos almacenados que comparten patrón.
- Cada fila de una tabla bidimensional corresponde a un registro u objeto único y distinguible.

Concepto y aplicaciones de las bases de datos. Ejemplos.

Requisitos de una base de datos:

- **Independencia:** Todos los registros de una tabla tienen la misma jerarquía y puede interaccionar de manera independiente con registros de otras tablas relacionadas.
- **Integridad:** La inserción y actualización de registros deben respetar las restricciones de la base de datos. La eliminación de un registro debe repercutir en todas las referencias que tuviera, dejando la estructura en un estado consistente.
- **Simplicidad:** Una distribución de los datos en tablas con pocas columnas, pero que refleje toda la información mediante interrelaciones es más manejable y fácil de mantener.
- **Unicidad:** Cada registro en una tabla está unívocamente determinado. Si es necesario se añade una "clave" a la tabla.

Concepto y aplicaciones de las bases de datos. Ejemplos.

Funcionalidad de una base de datos:

El gestor de la base de datos, mediante operaciones incluidas en las consultas basadas en **álgebra de conjuntos y operadores booleanos**, debe incorporar las siguientes funcionalidades para extraer la información que origina la estructura:

- Inserción, actualización y eliminación de registros.
- Concatenación de registros de diferentes tablas.
- Busquedas y filtrado de registros que cumplen ciertas condiciones y proyección de registros sobre un conjunto limitado de atributos.
- Ordenación de registros de manera ascendente o descendente según el campo o atributo elegido.

Concepto y aplicaciones de las bases de datos. Ejemplos.

Ejemplo 1 de estructuras de datos: Universidad

- **ELEMENTOS:**
  Centro: referencia, nombre, campus, localización,...
  Titulación: referencia, nombre, atribución, créditos,...
  Asignatura: referencia, titulación, curso, créditos, carácter,...
  Aula: referencia, localización, capacidad,...
  Estudiante: DNI, nombre, fecha_nac, contacto,...
  Docente: DNI, nombre, contacto, categoría, fecha_alta,...
- **INTERRELACIONES:**
  Centro/Titulación, Asign/Docente, Asign/Estudiante, Asign/Aula, Estudiante/Titulación
- **RESTRICCIONES:**
  Número máximo de convocatorias. Requisitos previos de acceso a titulación. Número de créditos aprobados para acceder a matriculación.

Concepto y aplicaciones de las bases de datos. Ejemplos.

Ejemplo 2 de estructuras de datos: Biblioteca

- **ELEMENTOS:**
  Libro: ISBN, título, edición, año, copia, idioma, tema, pags,...
  autor: referencia, apellido, nombre, nacionalidad,...
  Editorial: referencia, nombre, contacto, nacionalidad,...
  Distribuidora: referencia, nombre, contacto, nacionalidad,...
  Sala: referencia, localización, capacidad,...
  Socio: DNI, nombre, fecha_nac, contacto,...
  Personal: DNI, nombre, fecha_nac, contacto, fecha_alta,...
- **INTERRELACIONES:**
  Libro/Editorial, Editorial/Distrib, Libro/Autor, Socio/Libro.
- **RESTRICCIONES:**
  Máximo de libros de préstamo. Cuota de socio actualizada. Plazo de préstamo.

Errores clásicos en el almacenamiento de datos.

Observa la tabla que corresponde a todos los datos registrados en el sistema de una comercial de material fotográfico a partir de los pedidos recibidos. Indica las imperfecciones que observes:

cliente	**prod1**	**marca1**	**pais**	**prod2**	**marca2**	**pais**
Nikon	papel	Ruly	USA	null	null	null
Sony	film	Kodak	Japon	null	null	null
Minolta	papel	Ruly	USA	film	Agfa	Belg
Sony	film	Kodak	Japon	null	null	null
Nikon	film	Agfa	Belg	zoom	Tmr	Japon
Nikon	zoom	Sigma	USA	papel	Inacet	Fr

Errores clásicos en el almacenamiento de datos.

Errores estructurales

- Carencia de atributos diferenciadores o "claves" y aparición de registros idénticos, a pesar de que, quizás, correspondan a realidades diferentes.
- Información redundante que deriva en una sobreocupación de memoria e inconvenientes en la actualización y eliminación de datos.
- Atributos repetidos y aparición de campos nulos con un crecimiento horizontal indefinido de la estructura de la tabla.
- Carencia de registros en la BBDD por no haber pedidos que los referencien.

Errores clásicos en el almacenamiento de datos.

clientes

id_cl	cliente
c01	Olimpus
c02	Nikon
c03	Sony
c04	Pentax
c05	Minolta

proveedores

id_pr	prov	pais
p01	Ruly	USA
p02	Kodak	Japon
p03	Tmr	Japon
p04	Inacet	Fr
p05	Sigma	USA
p06	Agfa	Belg

Una distribución más eficiente de la información en tablas

pedidos

albaran	id_cl	id_pr	pedido
l5132	c02	p01	papel
l7301	c03	p02	film
l8122	c05	p01	papel
l0659	c05	p06	film
l3984	c03	p02	film
l2006	c02	p06	film
l2006	c02	p03	zoom
l0984	c02	p05	zoom
l0984	c02	p04	papel

Errores clásicos en el almacenamiento de datos.

Errores de formato

- Sintaxis con caracteres especiales que pueden diferir en función del editor, la codificación o el sistema operativo.
- Grabación de registros que correponden a la misma realidad, pero difieren en algún caracter o símbolo.
- Falta de coherencia en el tipo asignado a los atributos y en las restricciones de inserción, actualización y borrado.

Arquitecturas clásicas de bases de datos.

El modelo de bases de datos relacionales. Elementos:

Es el modelo teórico más flexible y universal utilizado para el almacenamiento de información y el diseño de aplicaciones prácticas y comerciales.

- Cada colección de datos que describen objetos o realidades similares se denomina **ENTIDAD** y se almacena en una tabla.
- Cada característica de una entidad, parametrizable en un cierto dominio y representada por un dato de un tipo determinado (números, fechas, cadenas alfanuméricas,...) se denomina **ATRIBUTO**.
- La asociación de objetos o registros de diferentes entidades se determina mediante **INTERRELACIONES**, que también se pueden almacenar en tablas de datos.

Arquitecturas clásicas de bases de datos.

El modelo de bases de datos relacionales. Identificadores o claves:

Los registros de cada tabla deben ser únicos, lo cual estará garantizado por un atributo "*clave*" o un conjunto de ellos.

- **Claves candidatas:** Es un atributo o conjunto de ellos que identifican univocamente a cada registro.
- **Claves primarias:** Es una clave candidata que se designa como prioritaria. Los gestores de BBDD automáticamente impiden que la clave primaria sea nula o coincida en dos registros diferentes.
- **Claves foráneas:** Son atributos de una tabla que, al mismo tiempo, son clave primaria en otra tabla. Son de gran utilidad para evitar inconsistencias al eliminar o modificar registros referenciados en otras tablas a través de un atributo común.

Arquitecturas clásicas de bases de datos.

El modelo de bases de datos relacionales. Propiedades:

Las entidades e interrelaciones de una base de datos, representadas por tablas, carecen de estructura jerárquica. Dos registros de tablas que con idéntico valor de un atributo común para ambas se pueden combinar directamente.

- **Independencia de filas:** Cada registro o fila de una tabla es independiente del resto y no le afecta el orden ni el instante en que ha sido grabado.
- **Independencia de columnas:** Las columnas de una tabla definen campos o atributos independientes y no les afecta el orden en que han sido declarados al generar la tabla.

Arquitecturas clásicas de bases de datos.

El modelo de bases de datos en red:

Las arquitecturas en red reflejan una perspectiva jerárquica de la dependencia entre registros. Esta tecnología está obsoleta por su rigidez y lentitud para el tratamiento de la información.

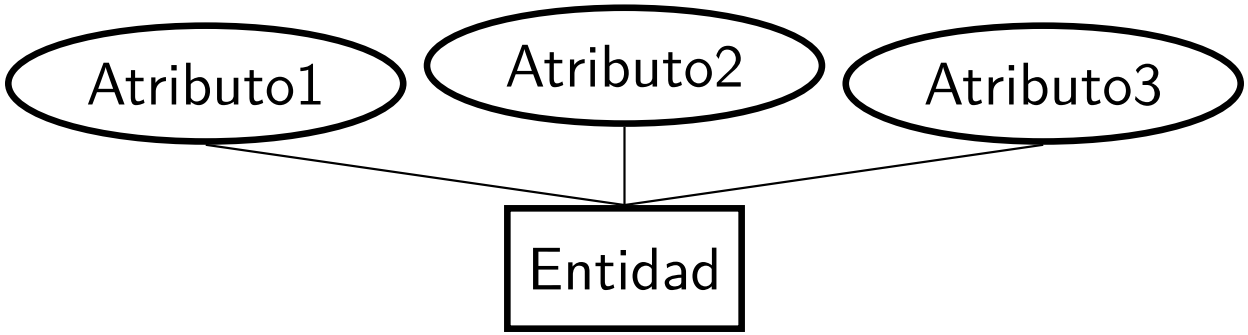
- **Registros:** Son las unidades de almacenamiento, con un título y una serie de atributos o campos.
- **Relaciones:** Se establecen entre dos registros, un *propietario* (dominante) y un *miembro* (dependiente). Un registro puede ser *miembro* en una relación y *propietario* en otra.
- **Acceso:** El acceso a la información se hace siguiendo la cadena de dependencias o apartir de un registro.

Por ejemplo, la relación `cl-alb` conecta el *propietario* `cliente` y el *miembro* `albarán` y, a su vez, la relación `alb-prod` conecta el *propietario* `albarán` y el *miembro* `producto`.

Elementos de los diagramas entidad/interrelación.

Entidades y atributos I:

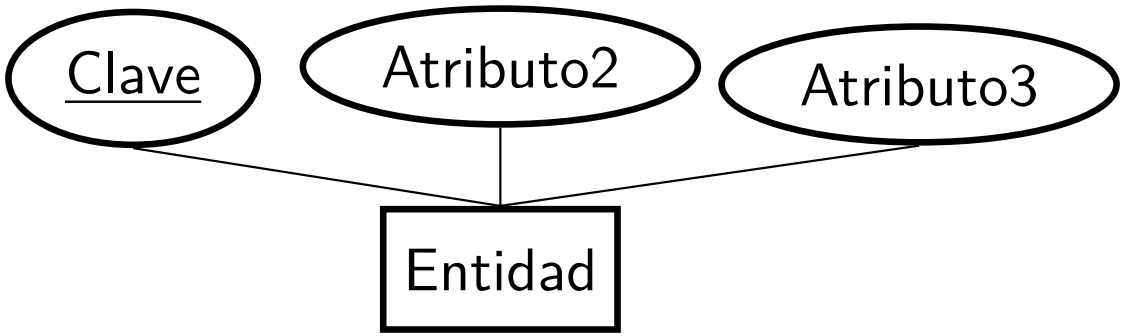
- Las colecciones de objetos de la BBDD que comparten estructura se agrupan en "**entidades**" y se almacenan en tablas. En el diagrama de flujo las entidades se representan por rectángulos.
- Cada característica o campo de una entidad se denomina "**atributo**" y ocupa una columna de la tabla. En el diagrama de flujo los atributos se representan por elipses.



Elementos de los diagramas entidad/interrelación.

Entidades y atributos II. Claves:

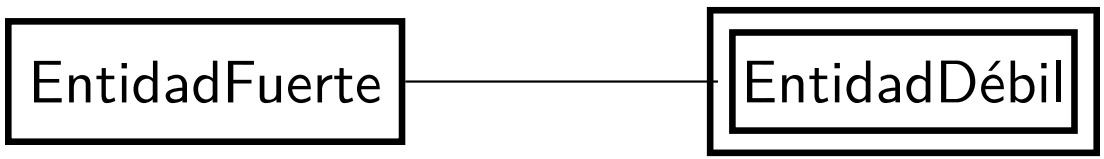
- Una "**clave candidata**" de una entidad es un atributo o un conjunto de ellos (sin subconjuntos propios que también lo sean) que determina univocamente los registros. A un valor concreto de una clave le corresponde un único registro.
- Una de las claves candidatas se selecciona como primaria y en el diagrama se representa como un atributo subrayado.
- Si una tabla carece de claves primarias conviene crear un campo identificativo que incluso refleje cierta información.



Elementos de los diagramas entidad/interrelación.

Entidades y atributos III. Entidades débiles:

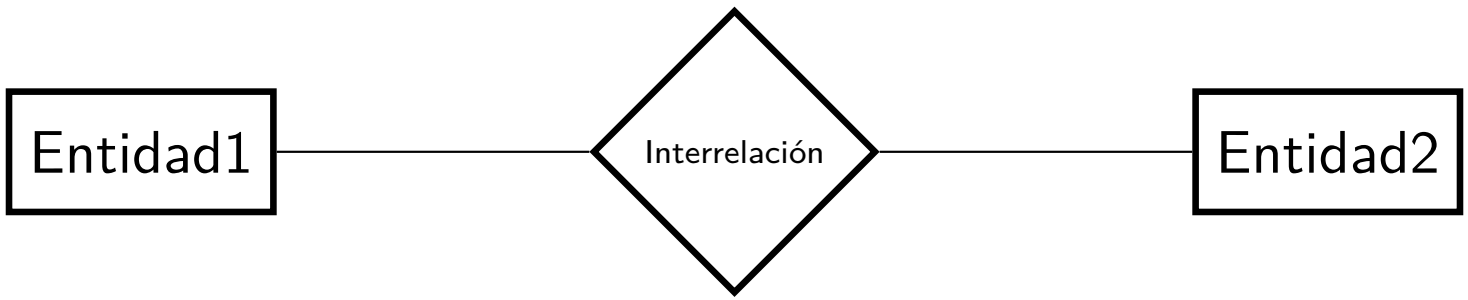
- Una "**entidad débil**" se representa por un doble rectángulo en el diagrama y es un tipo de objeto donde la existencia de cada registro está condicionados por la existencia de un registro en otra entidad regular de la BBDD, que se denomina "**entidad fuerte**". Es decir, si el registro de la entidad fuerte se elimina, entonces, los registros vinculados a éste en la entidad débil dejan de tener sentido en la BBDD.
- Cada registro de la entidad débil debe contener como **clave foránea** la **clave primaria** del registro asociado a él en la entidad fuerte. Esto facilita el mantenimiento de la BBDD.



Elementos de los diagramas entidad/interrelación.

Interrelaciones I:

- Una "**interrelación**" es una conexión entre dos o más entidades cuyos registros pueden estar asociados y se representa en el diagrama mediante un rombo.
- Una interrelación generalmente se transforma en una tabla que contiene las claves primarias de cada una de las entidades conectadas, aunque circunstancialmente también puede convertirse en una columna de una de esas dos entidades.
- Además de las claves primarias de las entidades asociadas a la interrelación, ésta también puede contener atribuos propios.



Elementos de los diagramas entidad/interrelación.

Interrelaciones II. Cardinalidad:

La cardinalidad de una interrelación puede ser de tres tipos:

i. *1:1*, si cada registro de una de las entidades conectadas por la interrelación aparece a lo sumo asociado a un registro de la otra entidad.
ii. *1:N*, si cada registro de la entidad izquierda puede estar asociado a numerosos registros de la entidad derecha, mientras que cada registro de la entidad de la derecha a lo sumo estará asociado a un registro de la entidad izquierda. Recíprocamente se define la cardinalidad *N:1*.
iii. *N:M*: si cada registro de cualquiera de las dos entidades conectadas por la interrelación puede estar asociado a numerosos registros de la otra entidad.

Elementos de los diagramas entidad/interrelación.

Interrelaciones III. Participación de las entidades:

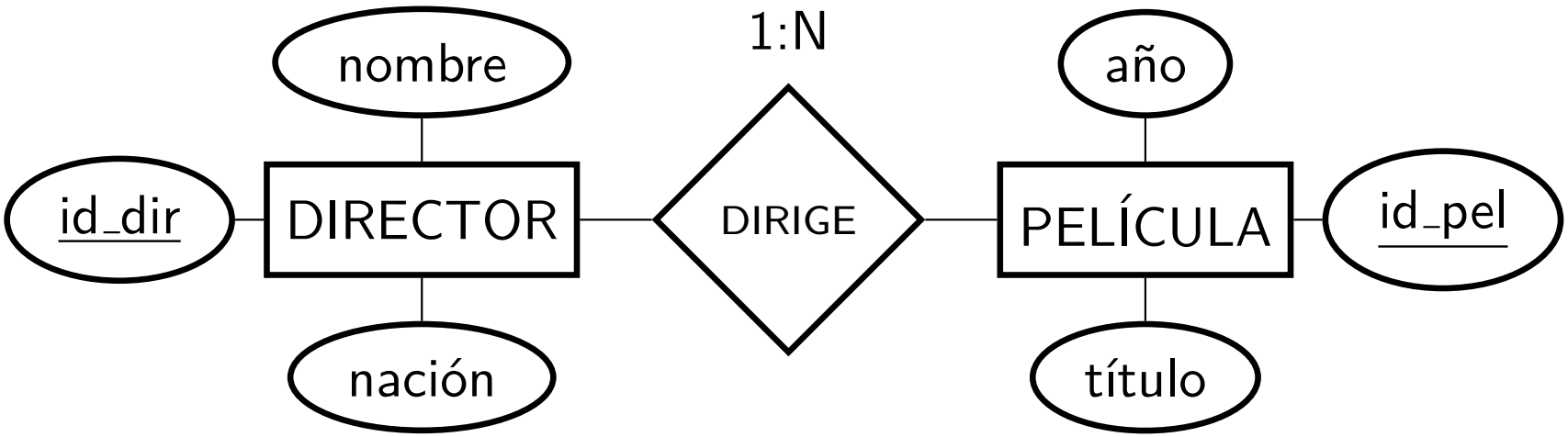
La participación de las entidades conectadas por una interrelación puede ser de dos tipos:

i. *Obligatoria*: si todos los registros de esa entidad al menos aparecen una vez relacionados con algún registro de la otra entidad conectada por la interrelación.
ii. *Optativa*: si pueden existir registros de esa entidad que no estén relacionados con ningún registro de la otra entidad conectada por la interrelación.

Elementos de los diagramas entidad/interrelación.

Ejemplo 1 de diagramas de flujo: Entidades e interrelaciones

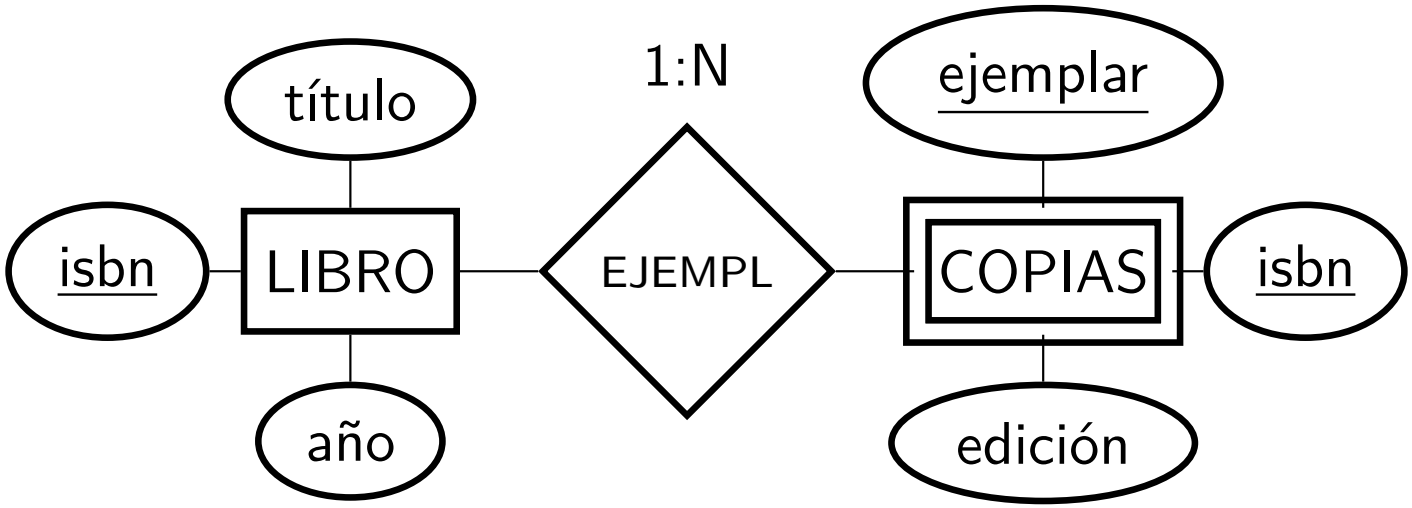
- **Entidades:**
  DIRECTOR: id_dir, nombre, nación,...
  PELÍCULA: id_pel, título, año,...
- **Interrelación:** DIRIGE: id_dir, id_pel
- **Cardinalidad:** 1:N



Elementos de los diagramas entidad/interrelación.

Ejemplo 2 de diagramas de flujo: Entidades fuertes y débiles

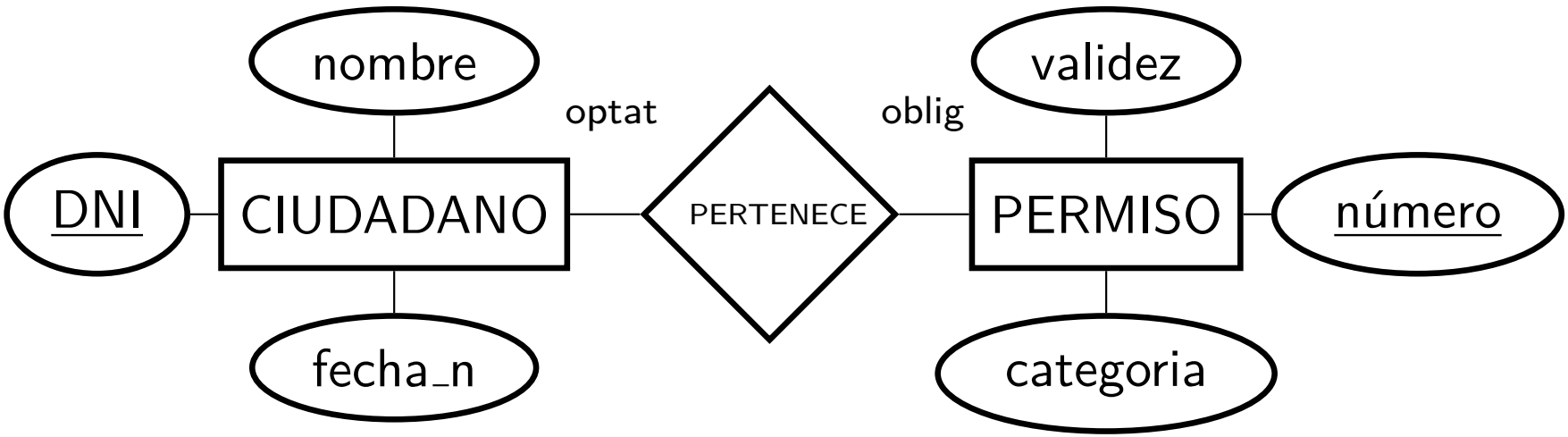
En la base de datos de una biblioteca se define la entidad "*libro*". Se dispone de varios ejemplares de los títulos más demandados que se registran en la entidad débil "*copias*". Si un libro se retira del catálogo de la biblioteca por contenido inadecuado u obsoleto y su registro se elimina de la tabla "*libro*", los registros vinculados a éste en en la tabla *copias*" también pueden eliminarse.



Elementos de los diagramas entidad/interrelación.

Ejemplo 3 de diagramas de flujo: Participación de las entidades

En la base de datos de la DGT se registran todos los ciudadanos a efectos de responsabilidades en posibles infracciones, y el conjunto de permisos de conducir en vigor. La primera entidad es de participación optativa, puesto que no todos los ciudadanos poseen permiso de conducir, mientras que la segunda es de participación obligatoria, puesto que cada permiso pertenece a algún conductor.



Transformación del diagrama en tablas.

Criterios para la propuesta de tablas I.

- Cada entidad se transforma en una tabla y cada interrelación en la mayoría de casos prácticos también se transforma en una tabla. Circunstancialmente, en algunos casos las interrelaciones pueden abreviarse, convirtiéndose en una simple columna de una de las entidades relacionadas.
- Los criterios para la forma en que se almacena en la BBDD una interrelación dependen de la cardinalidad de ésta y de la forma de participación, voluntaria u obligatoria, de las entidades conectadas.

Transformación del diagrama en tablas.

Criterios para la propuesta de tablas II.

Se pueden dar los siguientes casos en las interrelaciones

(i) 1:1 con participación obligatoria de ambas partes.

(ii) 1:1 con participación obligatoria de una parte y optativa de la otra.

(iii) 1:1 con participación optativa de ambas partes.

(iv) 1:N con participación obligatoria de la parte de N.

(v) 1:N con participación optativa de la parte de N.

(vi) N:M independientemente del tipo de participación.

(vii) Generalización.

(viii) Especialización.

Transformación del diagrama en tablas.

(i) 1:1 con participación obligatoria de ambas partes

La opción más eficiente es que ambas entidades y la interrelación se transformen en una sola tabla cuyos registros tienen los campos de cada par de elementos emparejados.

Ejemplo: La base de datos de un club hípico que registra a cada jinete y el caballo que monta, que es único y de su propiedad.

id_jinete	**Nombre**	**id_caballo**	**Nombre**
j005	Rodrigo	c0021	Babieca
j016	LuckyLuke	c009	JollyJumper
...	...	...	...

Transformación del diagrama en tablas.

(ii) 1:1 con participación obligatoria de un lado y optativa de otro

La opción más eficiente es que la entidad de partición optativa se transforme en una tabla y la de participación obligatoria en otra tabla con una columna de ésta última reservada para la clave de la primera tabla, que representa la interrelación.

Ejemplo: La base de datos de un hospital con el registro de empleados (entidad de participación optativa) y el listado de departamentos (entidad de participación obligatoria) que requieren de supervisor (interrelación).

id_empl	**Nombre**
t094	Lara
t189	Pedro
...	...

Ref_dpt	**depto.**	**id_sup**
s11	Cardio	t094
s05	Trauma	t057
...	...	...

Transformación del diagrama en tablas.

(iii) 1:1 con participación optativa de ambas partes

La opción más eficiente para almacenar toda la información sin redundancia es que cada entidad se transforme en una tabla y la interrelación que las conecta en una tercera tabla que almacene las claves de los registros emparejados de cada entidad.

Ejemplo: La base de datos de un club de patinaje artístico con las entidades vinculadas a las deportistas femeninas y a los deportistas masculinos y la interrelación de las parejas para competición.

patinadoras

id_pF	**Nombre**
F045	Diana
F106	Lisa
...	...

patinadores

id_pM	**Nombre**
M008	Mario
M099	Alex
...	...

parejas

id_pF	**id_pM**
F045	M099
F032	M127
...	...

Transformación del diagrama en tablas.

(iv) 1:N con participación obligatoria del lado de N

La opción más eficiente es que la entidad del lado de 1 se transforme en una tabla y la del lado N, que tiene participación obligatoria, en otra tabla con una columna reservada para la clave de la primera tabla y que equivale a la interrelación.

Ejemplo: La base de datos de una escuela, con entidades para estudiantes y docentes y una interrelación para indicar la asignación de algunos docentes como tutores de ciertos estudiantes.

docentes

id_pr	**Nombre**
t094	Lara
t189	Pedro
...	...

estudiantes

id_est	**Nombre**	**id_tut**
s595	Sonia	t094
s274	Emilio	t057
...	...	...

Transformación del diagrama en tablas.

(v) 1:N con participación optativa del lado de N

La opción más eficiente para almacenar toda la información sin redundancia es que cada entidad se transforme en una tabla y la interrelación que las conecta en una tercera tabla que almacene las claves de los registros emparejados de cada entidad.

Ejemplo: La base de datos de una federación de natación, con el registro de clubs y el de nadadores federados, cada uno por un club.

club

id_club	**Club**
C08	C.Splas
C35	C.Ola
...	...

nadador

id_nad	**Nombre**
N036	Dune
N377	Sara
...	...

competidores

id_club	**id_nad**
C08	N377
C08	N205
...	...

Transformación del diagrama en tablas.

(vi) N:M con participación indiferente por ambas partes

La opción más eficiente para almacenar toda la información sin redundancia es que cada entidad se transforme en una tabla y la interrelación que las conecta en una tercera tabla que almacene las claves de los registros asociados.

Ejemplo: La base de datos de nadadores federados, competiciones y como interrelación la participación de nadadores en éstas últimas.

nadador

id_nad	**Nombre**
N036	Dune
N377	Sara
...	...

competición

id_cp	**Camp**
R08	CP2019
R17	CE2020
...	...

participación

id_club	**id_cp**
N377	CP2019
N377	CL2018
N510	CP2019

Transformación del diagrama en tablas.

(vii) Generalización

Entidades que comparten varios atributos son susceptibles de ser agrupadas en una sola "superentidad" o tabla con dichos atributos y unas tablas por cada entidad con los atributos o matices diferenciadores además de la columna de identificación o clave.

Ejemplo: La base de datos de una empresa de alquiler de vehículos que cuenta con turismos y furgonetas.

Turismos

matric	**CV**	**km**	**gama**
KVH34	90	3562	eco
JDK76	140	6347	berl
FME10	200	1995	lujo
...	...	...	...

Industriales

matric	**CV**	**km**	**uso**
HSB34	110	8643	pas
LWF76	150	3791	trn
JRG00	180	4368	cmp
...	...	...	...

Transformación del diagrama en tablas.

La información se almacena en 3 tablas: "vehículos" con las columnas comunes y "turismos" e "industriales" con las distintivas:

Vehículos

matric	**CV**	**km**
KVH34	90	3562
JDK76	140	6347
FME10	200	1995
HSB34	110	8643
LWF76	150	3791
JRG00	180	4368
...	...	...

Turismos

matric	**gama**
KVH34	eco
JDK76	berl
FME10	lujo
...	...

Industriales

matric	**uso**
HSB34	pasaj
LWF76	indust
JRG00	camper
...	...

Transformación del diagrama en tablas.

(viii) Especialización

Para evitar redundancia en tablas donde aparecen registros prácticamente idénticos, con a penas, algún matiz diferenciador, se considera una tabla con los atributos donde puede haber registros idénticos y otra tabla para los atributos diferenciadores.

Ejemplo: La base de datos de deportistas en unos juegos olímpicos.

licencia	**nombre**	**pais**	**disciplina**
USA095	Anthony Philips	USA	100m
USA095	Anthony Philips	USA	longitud
ESP029	Juan Tobías	ESP	5000m
ESP029	Juan Tobías	ESP	10000m
RUS134	Vasile Ivanovich	RUS	pértiga
...	...	...	...

Transformación del diagrama en tablas.

La información se almacena en la tabla "atletas" con las columnas potencialmente redundantes y la tabla "especialidades" para indicar las especialidades deportivas en las que competirá cada atleta:

Deportistas

licencia	**nombre**	**pais**
USA095	AnthonyPhil	USA
ESP029	JuanTobías	ESP
RUS134	VasileIvan	RUS
...	...	...

Especialidades

licencia	**disciplina**
USA095	100m
USA095	longitud
ESP029	5000m
ESP029	10000m
RUS134	pértiga
...	...

Transformación del diagrama en tablas.

Ejercicio 1: Diseñar un diagrama de la base de datos de un banco que permita obtener las siguientes vistas o consultas:

Perfil cliente					
Id_cl	**TipoDoc**	**nº**	**Apellido1**	**Apellido2**	**Nombre**
R80086	DNI	64092815B	Ser	Lana	Miriam

Dirección	**CP**	**Ciudad**	**F.Nac.**	**email**	**Tef.**
Alta,21-4B	23440	Baeza	3-9-1985	mser@tail.org	6923764

Cuentas vinculadas al cliente R80086					
RefC	**Id_cl**	**Tipo**	**oficina**	**FechaAp**	**Saldo**
CC845729	R80086	CCorriente	291	2015-01-19	3725
CV543709	R80086	CValores	291	2015-01-19	0
CP310866	R80086	CPrestamo	291	2017-08-04	-73200

Transformación del diagrama en tablas.

Cuenta CC845729. Movimientos periodo 03/02-03/03/2029				
ref	**concepto**	**efectivo**	**saldo**	**fecha**
I290377	Ingreso. Nomina	1450.00	3181.30	2029-03-02
R290326	Reintegro. Cajero	-45.00	3136.30	2029-03-03
T290366	Transferencia. Gimnasio	-36.00	3100.30	2029-03-07
V290395	Compra. Acc. PhElectric	-753.25	2347.05	2029-03-08
I290367	Ingreso. Divid. TmBank	18.43	2365.48	2029-03-10
R290340	Reintegro. Cajero	-60.00	2305.48	2029-03-10
T290343	Transferencia. Librería	-28.60	2276.88	2029-03-12
R290347	Reintegro. Cajero	-20.00	2256.88	2029-03-13
V290315	Venta. Acc. PtUnion	630.82	2887.70	2029-03-14
D290354	Domiciliación. Elect.	-31.55	2856.15	2029-03-14

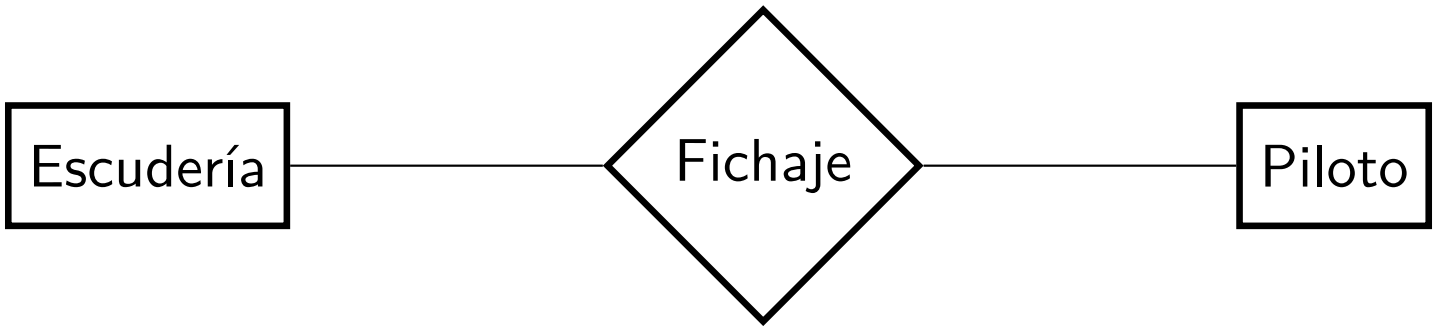
Transformación del diagrama en tablas.

Cuenta C845729. Transferencias periodo 15/05-31/05/2026					
ref	**concepto**	**efectivo**	**c. cargo**	**c. abono**	**fecha**
T260501	c.natación	34.00	CC845729	CC6589272	2026-05-16
T260502	dentista	65.00	CC845729	CC9835478	2026-05-19
T260503	dietas	121.00	CC5002826	CC845729	2026-05-27

Cuenta CV543709. Oper. bolsa periodo 15/05-31/05/2026						
ref	**valor**	**op**	**acc**	**efectivo**	**c. cargo**	**fecha**
V260501	C	CitricosSA	40	-550.	543709	2026-05-18
V260502	V	TurbinasSA	25	890.	543709	2026-05-29
V260503	C	WireconSA	60	-720.	543709	2026-05-30

Transformación del diagrama en tablas.

Ejercicio 2: Considerar la BBDD de una web de motor y proponer dos situaciones. En una la cardinalidad de la interrelación "*Fichaje*" debe ser formulada como 1:N y otro en el que debe serlo como N:M. Discute también las circunstancias bajo las cuales la participación de ambas entidades puede ser formulada como obligatoria u optativa:



Transformación del diagrama en tablas.

Ejercicio 3: Considerar la BBDD de una tienda online de música donde, entre otras, se habrán definido las entidades "artista", "grupo" y "canción". Proponer un esquema de interrelaciones que consideres lógico y determinar la cardinalidad y el tipo de participación de las entidades vinculadas.

Ejercicio 4: En la BBDD de la federación de waterpolo se considera la entidad "equipo" para para almacenar la información propia de cada conjunto que compite en una categoría oficial. Determinar que tipo de elemento se necesitaría en la BBDD para para representar los enfrentamientos entre dos equipos.

Restricciones de integridad y de usuario.

Integridad inherente:

- La clave primaria es un atributo de la tabla que no puede ser nulo ni estar duplicado.
- Los valores factibles para un atriuto están condicionados por el tipo asignado (número entero o real, cadena alfanumérica, fecha, etc) y por el dominio acotado que se puede imponer.

Ejemplo: En la base de datos de la Agencia Tributaria se asigna el NIF o CIF del contribuyente como clave primaria. Pero hay opciones como "`unique`", "`not null`" o "`auto_increment`"para dotar a otros atributos de funcionalidad similar.

Ejemplo: Para un atributo categórico que requiere una respuesta si/no, se puede definir un dominio acotado "`enum('si','no')`".

Restricciones de integridad y de usuario.

Integridad referencial:

Dadas dos tablas relacionadas mediante un atributo que es clave primaria en una de ellas y clave foránea en la otra, al eliminar o modificar registros de la primera se debe elegir entre una de las siguientes alternativas para preservar la coherencia referencial:

- *Restringida:* Impide borrar en una tabla registros que estén referenciados en otra tabla mediante una clave foránea.
- *En cascada:* Las modificaciones en los registros de una tabla, se trasladan a las tablas asociadas mediante una clave foránea.
- *Indeterminada:* La eliminación de registros en una tabla, transforma en nulos los valores de claves foráneas de otras tablas conectadas con la primera mediante dichas claves.
- *Sin efecto:* La eliminación de registros en una tabla no afecta a registros de las tablas conectadas mediante claves foráneas.

Restricciones de integridad y de usuario.

Ejemplo: En la BBDD de una tienda online se declaran las tres entidades

- `fabricante`: id_fab, marca, ...
- `distribuidor`: id_dis, nombre, ...
- `artículo`: id_art, artículo, precio, id_fab, id_prv, ...

"id_fab" e "id_dis‘" son claves foráneas de la entidad `artículo`.

Si desaparece un fabricante es factible actuar en *cascada* con los artículos referenciados mediante "id_fab" por desabastecimiento.

Si desaparece un distribuidor es más lógico que la repercusión en los artículos referenciados mediante "id_dis‘" se defina como *indeterminada* ya que otro distribuidor puede proveerlos.

Restricciones de integridad y de usuario.

Restricciones de usuario:

Para evitar estados infactibles del sistema, se puede:

i. acotar el dominio de definición de los atributos,

ii. imponer restricciones a la inserción y modificación de registros mediante unos subprogramas (*triggers*) que se activan automáticamente "*antes*" o "*después*" de alguna de las siguientes acciones:

   - *insertar* (insert),
   - *actualizar* (update),
   - *borrar* (delete).

Ejemplo: Antes de grabar una *transferencia* en la tabla *operaciones* de una entidad financiera, se comprueba que la cuenta de abono es solvente, es decir, su saldo es mayor que el importe a transferir.

Normalización y depuración de tablas.

Dependencias funcionales I. Notación.

Para representar los elementos de los esquemas relacionales se utiliza la siguiente simbología

- **Tablas**: se denotan por mayúsculas en negrita, **R**, **S**, etc.
- **Atributos**: se denotan por mayúsculas en itálica, "A", "B", etc. Una concatenación de atributos se escribiría AB.
- **Dominios**: los rangos de los atributos se denotan por letras griegas, "α", "β", etc. Por ejemplo, $A \in \alpha$.
- **Claves**: se determinan subrayando los atributos elegidos para este fin, "$\underline{A}$", "$\underline{AB}$", etc.
- **Registros**: cada registro se denota por una letra minúscula, "x", "y", etc.
- **Valores**: x.A denota el valor del atributo A en el registro x.

Normalización y depuración de tablas.

Dependencias funcionales II. Definición.

Un atributo o un conjunto de ellos "A" determina univocamente otro "B" cuando para cada registro "x", conocido el valor x.A se deduce el valor x.B. Se dice que "B *depende funcionalmente* de A" y se escribe $A \rightarrow B$.

Lógicamente, todos los atributos de una tabla dependen funcionalmente de cualquiera de las claves candidatas.

Se dice que un atributo C depende *totalmente* de una clave $\underline{AB}$, cuando no existe ningún subconjunto de dicha clave que determine funcionalmente C. Si existiera un subconjunto A de la clave, tal que $A \rightarrow C$, entonces se diría que la dependencia es *parcial*.

Normalización y depuración de tablas.

Dependencias funcionales III. Axiomas de Armstrong.

- Reflexividad: $B \subset A \Longrightarrow A \rightarrow B$.
- Aumento: $A \rightarrow B \Longrightarrow AC \rightarrow BC$.
- Transitividad: $A \rightarrow B \ \wedge \ B \rightarrow C \Longrightarrow A \rightarrow C$.
- Unión: $A \rightarrow B \ \wedge \ A \rightarrow C \Longrightarrow A \rightarrow BC$.
- Pseudotransitividad: $A \rightarrow B \ \wedge \ BD \rightarrow C \Longrightarrow AD \rightarrow C$.
- Descomposición: $A \rightarrow B \ \wedge \ C \subset B \Longrightarrow A \rightarrow C$.

Normalización y depuración de tablas.

Dependencias funcionales IV. Atributos multivaluados.

Un atributo B es multivaluado respecto a A, $A \rightarrow\!\!\!\rightarrow B$, cuando cada valor de A tiene asociados una serie de valores de B.

Si $A \rightarrow\!\!\!\rightarrow B$ y el conjunto de valores de B es independiente del valor asociado de A entonces el producto cartesiano $A \times B$ representa todas las posibles tuplas.

Normalización y depuración de tablas.

Ejemplo: La base de datos de una tienda almacena las referencias de los artículos (atributo A), la categoría que le corresponde (atributo B) y tasa del IVA aplicado (atributo C), las dependencias funcionales detectadas serían $A \rightarrow B$ y, a su vez, $B \rightarrow C$.

Ejemplo: Son frecuentes las dependencias multivaluadas, $A \rightarrow\rightarrow B$, en catálogos de productos con diversas variantes. En ocasiones los valores de B son independientes de A (tabla de la izquierda) y en otras existe dependencia (tabla de la derecha).

zapatillas

Modelo	**color**
Sport	blanco
Sport	negro
Clasic	blanco
Clasic	negro

vehículos

Modelo	**plazas**
deportivo	2
deportivo	4
monovolumen	5
monovolumen	8

Normalización y depuración de tablas.

1FN (primera forma normal de Codd).

Un esquema verifica la primera forma normal **1FN** si cumple las siguientes premisas:

- No pueden existir dos registros idénticos en una tabla. Si es necesario, debe definirse una clave primaria para distinguir objetos diferentes que tengan registros coincidentes.
- La clave primaria no puede coincidir en dos registros.
- El orden de las filas (tuplas o registros) y de las columnas (atributos) de una tabla es irrelevante.
- En una tabla los atributos son atómicos, es decir, no se pueden definir atributos que representen la misma realidad y con idéntico dominio. Lo contrario supondría la proliferación de valores nulos y redundancia en la tabla y su crecimiento horizontal y alteración de su estructura.

Normalización y depuración de tablas.

Ejemplo: Listado de miembros de un club y datos de contacto.

Socios y telefonos

Ref	Nombre	Tel1	Tel2
S09	Julia	91374	68873
S12	Yolanda	60276	
S28	Félix	94938	61192

$\Longrightarrow$

Socios

Ref	Nombre
S09	Julia
S12	Yolanda
S28	Félix

Teléfonos

Ref	Tel
S09	91374
S09	68873
S12	60276
S28	94938
S28	61192

Normalización y depuración de tablas.

2FN (segunda forma normal de Codd).

Un esquema verifica la segunda forma normal **2FN** si cumple **1FN** y, además,

no presenta dependencias parciales respecto a una clave candidata. Es decir, un atributo que no pertenezca a ninguna clave candidata no puede depender funcionalmente solo de parte de una clave.

Un esquema $\{\underline{AB}, C\}$ con $B \rightarrow C$ debe descomponerse en dos tablas $\{\underline{AB}\}$ y $\{\underline{B}, C\}$

Normalización y depuración de tablas.

Ejemplo: Listado de nadadores en una competición por clubs:
A=Id_n, B=dorsal, C=club y D=ciudad
Tanto $\underline{A}$ como $\underline{BC}$ son claves candidatas y $C \rightarrow D$.

Datos competición

Id_n	**dorsal**	**club**	**ciudad**
N304	05	Olas	Leon
N312	13	Olas	Leon
N426	02	Buceo	Huesca
N341	17	Buceo	Huesca
...	...	...	...

$\Longrightarrow$

Datos nadador

Id_n	**dorsal**	**club**
N304	05	Holas
N312	13	Holas
N426	02	Buceo
N341	17	Buceo

Datos club

club	**ciudad**
Holas	Leon
Buceo	Huesca

Normalización y depuración de tablas.

3FN (segunda forma normal de Codd).

Un esquema verifica la tercera forma normal **3FN** si cumple **2FN** y, además,

no presenta dependencias transitivas respecto a la clave principal. Es decir, no puede haber una cadena de dos o más dependencias funcionales sucesivas entre atributos con origen en una clave.

Un esquema $\{\underline{A}, B, C\}$ con $A \rightarrow B$ y $B \rightarrow C$ debe descomponerse en dos tablas $\{\underline{A}, B\}$ y $\{\underline{B}, C\}$

Normalización y depuración de tablas.

Ejemplo: Listado de ciudades de más de un millón de habitantes:
A=ciudad, B=país, C=continente y D=pob.
$\underline{A}$ es clave primaria y se da que $A \to B$ y $B \to C$.

Ciudades y población

ciudad	**país**	**cont**	**pob**
Madrid	Esp	Europa	3.2m
Barcelona	Esp	Europa	1.6m
El Cairo	Egi	Africa	8.2m
Bogota	Col	AméricaS	7.7m
Seul	CorS	Asia	10.5m
...	...	...	...

$\Rightarrow$

Ciudades y población

ciudad	**país**	**pob**
Madrid	Esp	3.2m
Barcelona	Esp	1.6m
El Cairo	Egi	8.2m
Bogota	Col	7.7m

Países

país	**cont**
Esp	Europa
Egi	Africa

Normalización y depuración de tablas.

FNBC (forma normal de Boyce-Codd).

Un esquema verifica la forma normal de Boyce-Codd **FNBC** si cumple **3FN** y, además,

no ocurre que un atributo no incluido en una clave candidata determine funcionalmente parte de dicha clave. Es decir, los únicos atributos que determinan funcionalmente a otros pertenecen a claves candidatas.

Un esquema $[\underline{AB}, C]$ con $C \rightarrow B$ debe descomponerse en dos tablas $[\underline{AC}]$ y $[\underline{C}, B]$

Normalización y depuración de tablas.

Ejemplo: Horarios de clase. Cada docente solo esta involucrado en una asignatura o parte de ella, y la imparte ciertos días:
A=día, B=hora, C=asignatura y D=docente, con $\underline{ABC}$ y $D \to C$.

Horario y docentes

día	**hora**	**asign**	**docente**
lunes	9:00	cálculo	Ana
lunes	10:00	inform	Juanjo
lunes	11:00	física	Ester
martes	9:00	física	Ester
martes	10:00	cálculo	Luis
martes	11:00	inform	Pilar
miercoles	9:00	cálculo	Ana
miercoles	10:00	física	Ester

$\Longrightarrow$

Horario

día	**hora**	**docente**
lunes	9:00	Ana
lunes	10:00	Juanjo
lunes	11:00	Ester
martes	9:00	Ester

Docencia

docente	**asign**
Ana	cálculo
Juanjo	inform

Normalización y depuración de tablas.

4FN (cuarta forma normal de Codd).

Un esquema verifica la cuarta forma normal de Codd **4FN** si cumple **FNBC** y, además,

en el esquema no puede haber dependencias multivaluadas no triviales, es decir solo se permiten las que consisten entre dos partes de la clave o en una clave y un atributo no primario. Tampoco hay cadenas transitivas de dependencias multivaluadas. De lo contrario se genera redundancia en la información.

Un esquema $\{\underline{ABC}\}$ con $A \rightarrow\rightarrow B$ y $B \rightarrow\rightarrow C$ debe descomponerse en dos tablas $\{\underline{AB}\}$ y $\{\underline{BC}\}$

Un esquema $\{\underline{ABC}\}$ con $A \rightarrow\rightarrow B$ y $A \rightarrow\rightarrow C$ debe descomponerse en dos tablas $\{\underline{AB}\}$ y $\{\underline{AC}\}$

Normalización y depuración de tablas.

Ejemplo: Listado de tipos de tren, tipos de pasaje y servicios:
A=tren, B=pasaje y C=servicio.
$\underline{ABC}$ es clave candidata y se dan $A \rightarrow\rightarrow B$ y $B \rightarrow\rightarrow C$.

trenes y servicios

tren	**pasaje**	**servicio**
AVE	turista	cafe
AVE	vip	cafe
AVE	vip	wifi
Talgo	turista	cafe
Talgo	vip	cafe
Talgo	vip	wifi
Regional	turista	cafe
...	...	...

$\Longrightarrow$

tren	**pasaje**
AVE	turista
AVE	vip
Talgo	turista
Talgo	vip
Regional	turista

pasaje	**servicio**
turista	cafe
vip	cafe
vip	wifi

Normalización y depuración de tablas.

5FN (quinta forma normal de Codd).

Esta restricción se formuló para evitar la redundancia inducida por atributos parcialmente multivaluados, es decir, cuando la tabla no se compone de todas las combinaciones del producto cartesiano de los atributos multivaluados, sino que faltan algunas combinaciones. Por ello se exige que para alcanzar la **5FN** además de cumplir la **4FN** se eviten la presencia de múltiples atributos parcialmente multivaluados. No siempre se puede encontrar una solución completamente carente de redundancia. Dependiendo del caso, se debe analizar qué solución puede evitar en parte dicha redundancia.

Reunificación mediante álgebra relacional y de conjuntos.

Unificación o concatenación:

El objetivo de la descomposición de tablas es el de distribuir la información en una estructura que sea eficiente para insertar, actualizar o borrar registros, que acelere ordenaciones y consultas y que evite redundancias y ambigüedades.

Es fundamental establecer una red de interrelaciones y de claves foráneas, a partir de las cuales, se pueda recuperar la información original de forma completa y precisa mediante la concatenación o unión natural (join). Esta operación permite asociar registros de dos tablas para los que coincide el valor de una clave que es atributo común a ambas tablas.

Gracias a las concatenaciones y otras operaciones lógicas y algebraicas se extrae, combina y filtra la información contenida en las tablas para obtener las descripciones y estados demandados.

Reunificación mediante álgebra relacional y de conjuntos.

Ejemplo: Establecimientos hosteleros en el catálogo de una agencia de viajes y servicios ofertados:

Establecimientos

id_h	**nombre**	**ciudad**
H005	Tobazo	Huesca
A381	Alcón	Ávila

servicios

id_h	**servicio**
H005	aloj.
H005	menú
A381	cafet.
A381	pizz.

$\Longrightarrow$

Establecimientos y servicios

id_h	**nombre**	**ciudad**	**servicio**
H005	Tobazo	Huesca	aloj.
H005	Tobazo	Huesca	menú
A381	Alcón	Ávila	cafet.
A381	Alcón	Ávila	pizz.

Reunificación mediante álgebra relacional y de conjuntos.

Operaciones de conjuntos I. Unión, intersección y diferencia:

Las operaciones de unión e intersección se pueden aplicar a tablas de idéntica estructura, es decir, mismos atributos y dominios.

Consideramos las tablas
$\mathbf{R}[A_1 \in \alpha_1, \ldots, A_n \in \alpha_n]$ y $\mathbf{S}[A_1 \in \alpha_1, \ldots, A_n \in \alpha_n]$,

$$
\begin{aligned}
\mathbf{T} = \mathbf{R} \cup \mathbf{S} &\Rightarrow \mathbf{T} = \{\mathrm{t} : (\exists \mathrm{r} \in \mathbf{R},\ \mathrm{t} = \mathrm{r}) \vee (\exists \mathrm{s} \in \mathbf{S},\ \mathrm{t} = \mathrm{s})\},\\
\mathbf{T} = \mathbf{R} \cap \mathbf{S} &\Rightarrow \mathbf{T} = \{\mathrm{t} : (\exists \mathrm{r} \in \mathbf{R},\ \mathrm{t} = \mathrm{r}) \wedge (\exists \mathrm{s} \in \mathbf{S},\ \mathrm{t} = \mathrm{s})\},\\
\mathbf{T} = \mathbf{R} - \mathbf{S} &\Rightarrow \mathbf{T} = \{\mathrm{t} : (\exists \mathrm{r} \in \mathbf{R},\ \mathrm{t} = \mathrm{r}) \wedge (\nexists \mathrm{s} \in \mathbf{S},\ \mathrm{t} = \mathrm{s})\}.
\end{aligned}
$$

Reunificación mediante álgebra relacional y de conjuntos.

Ejemplo: Listados de empresas de reparaciones concertadas por dos compañías de seguros y servicios ofertados:

CompSeguros1

id_e	**empr.**	**servicio**
E004	GanzuaSA	carpint.
E004	GanzuaSA	cerraj.
E096	ChispasSA	electr.
E135	GrifosSA	fontan.

CompSeguros2

id_e	**empr.**	**servicio**
E041	ZocalosSA	carpint.
E004	GanzuaSA	cerraj.
E187	ParabolSA	antenas

$\Longrightarrow$

CompSeguros1 $\cup$ CompSeguros2

id_e	**empr.**	**servicio**
E004	GanzuaSA	carpint.
E004	GanzuaSA	cerraj.
E041	ZocalosSA	carpint.
E096	ChispasSA	electr.
E135	GrifosSA	fontan.
E187	ParabolSA	antenas

CompSeguros1 $\cap$ CompSeguros2

id_e	**empr.**	**servicio**
E004	GanzuaSA	cerraj.

Reunificación mediante álgebra relacional y de conjuntos.

Operaciones de conjuntos II. Producto cartesiano:

Dadas dos tablas que no tienen que compartir necesariamente ningún atributo, su producto cartesiano es el listado de todos los registros resultantes de la concatenación de un registro cualquiera de la primera tabla con un registro cualquiera de la segunda tabla.

El número de nuevos registros es el producto de los números de registros de cada tabla, dado que emergen todas las combinaciones posibles entre los elementos de ambas estructuras.

Consideramos las tablas $\mathbf{P}$ y $\mathbf{Q}$,

$$\mathbf{T} = \mathbf{P} \times \mathbf{Q} \quad \Rightarrow \quad \mathbf{T} = \{t = pq : (\exists p \in \mathbf{P}) \vee (\exists q \in \mathbf{Q})\}.$$

Reunificación mediante álgebra relacional y de conjuntos.

Ejemplo: Catálogo de pizzas de un restaurante:

Bases

grosor	**textura**
fina	crujiente
fina	elástica
gruesa	esponjosa

Ingredientes

relleno	**queso**
bacon	mozarella
gambas	emmental

$\Rightarrow$

Bases $\times$ Ingredientes

grosor	**textura**	**relleno**	**queso**
fina	crujiente	bacon	mozarella
fina	crujiente	gambas	emmental
fina	elástica	bacon	mozarella
fina	elástica	gambas	emmental
gruesa	esponjosa	bacon	mozarella
gruesa	esponjosa	gambas	emmental

Reunificación mediante álgebra relacional y de conjuntos.

Operaciones relacionales I. Proyección:

Esta operación supone la proyección de una tabla sobre un subespacio o subconjunto de atributos. El resultado es el listado de los registros o tuplas (filas) diferentes que se obtienen al elegir solo una parte de los atributos.

Consideramos la tabla $\mathbf{R}[A_1 \in \alpha_1, \ldots, A_n \in \alpha_n]$ y el subconjunto de atributos $\{A_1, \ldots, A_k\} \subset \{A_1, \ldots, A_n\}$, entonces

$$\mathbf{T} = \Pi_{\{A_1,\ldots,A_k\}} R \quad \Rightarrow \quad \mathbf{T} = \{\mathrm{t} \in \alpha_1 \times \cdots \times \alpha_k : (\exists \mathrm{r} \in \mathbf{R} : \mathrm{t}.A_i = \mathrm{r}.A_i,\ \forall i = 1, \ldots, k)\}.$$

Reunificación mediante álgebra relacional y de conjuntos.

Ejemplo: Inventario de prendas de vestir en una tienda:

Catálogo productos

tipo	**material**	**talla**	**color**
jersey	algodón	L	azul
jersey	poliester	S	blanco
camiseta	algodón	M	rojo
camiseta	algodón	L	negro
camiseta	poliester	XL	blanco
camiseta	poliester	S	azul
pantalon	algodón	L	blanco
pantalon	algodón	M	rojo

$\Longrightarrow$

Catálogo prendas/material

tipo	**material**
jersey	algodón
jersey	poliester
camiseta	algodón
camiseta	poliester
pantalon	algodón

Reunificación mediante álgebra relacional y de conjuntos.

Operaciones relacionales II. Restricción:

Consideramos la aplicación $f : \mathbf{S} \to \{\text{“}verdadero\text{”}, \text{“}falso\text{”}\}$ formulada mediante operadores relacionales sobre la tabla $\mathbf{S}$. Una restricción $\sigma_f(\mathbf{R})$ significa la selección de los registros que cumplen la condición $f(s) = \text{“}verdadero\text{”}$ y se representa:

$$\sigma_f(\mathbf{R}) = \{\mathrm{t} : (\exists \mathrm{r} \in \mathbf{R}, \mathrm{t} = \mathrm{r}) \wedge f(t) = \text{“}verdadero\text{”}\}$$

Ejemplo: Selección de atletas sub23 de un club, $f(\Pi_{edad}\mathbf{R}) < 23$:

atletas

Id_A	**nombre**	**edad**
A208	Marta	26
A196	Juan	30
A053	Pablo	21
A014	Irene	22

$\Longrightarrow$

atletas_sub23

Id_A	**nombre**	**edad**
A053	Pablo	21
A014	Irene	22

Reunificación mediante álgebra relacional y de conjuntos.

Operaciones relacionales III. Concatenación o "natural join":

Esta operación consiste en la fusión de registros de dos tablas que coinciden en el valor o conjunto de valores comunes a ambas tablas. Técnicamente la operación se denomina "*natural join*".

Dadas las tablas
$\mathbf{R}[A_1 \in \alpha_1, \ldots, A_k \in \alpha_k, A_{k+1} \in \alpha_{k+1}, \ldots, A_n \in \alpha_n]$ y
$\mathbf{S}[A_1 \in \alpha_1, \ldots, A_k \in \alpha_k, B_{k+1} \in \beta_{k+1}, \ldots, B_m \in \beta_m]$, el conjunto de registros de la concatenación $\mathbf{T} = \mathbf{R} \bowtie \mathbf{S}$ se representa

$$\mathbf{T} = \mathbf{R} \bowtie \mathbf{S} \Rightarrow \quad \mathbf{T} = \{\mathrm{rs} \in \alpha_1 \times \cdots \times \alpha_n \times \beta_{k+1} \times \cdots \times \beta_m : (\exists \mathrm{r} \in \mathbf{R} \wedge \exists \mathrm{s} \in \mathbf{S} : \mathrm{r}.A_i = \mathrm{s}.A_i,\ \forall i = 1, \ldots, k)\}.$$

Reunificación mediante álgebra relacional y de conjuntos.

Ejemplo: Concatenación de la información de parcelas de secano y el cultivo de cereal elegido anualmente para cada una de ellas:

parcelas para cereal

Id_finca	**m²**
F03	63000
F12	98000

$\bowtie$

cultivos y precio

Id_finca	**año**	**cultivo**	**€/kg**
F03	2027	trigo	0.47
F03	2028	avena	0.83
F12	2027	cebada	0.55
F12	2028	trigo	0.49

$\Downarrow$

Id_finca	**m²**	**año**	**cultivo**	**€/kg**
F03	63000	2027	trigo	0.47
F03	63000	2028	avena	0.83
F12	98000	2027	cebada	0.55
F12	98000	2028	trigo	0.49

Reunificación mediante álgebra relacional y de conjuntos.

División:

Consideramos las tablas $\mathbf{R}[A, B]$ y $\mathbf{S}[B]$, los conjuntos de atributos $A = \{A_1, \ldots, A_n\}$ y $B = \{B_1, \ldots, B_m\}$ y el subconjunto $\hat{A} \subset A$.

La división $\Pi_{\hat{A}}\mathbf{R} \div \Pi_B\mathbf{S}$ de la proyección $\Pi_{\hat{A}}\mathbf{R}$ entre la proyección $\Pi_B\mathbf{S}$ es el listado de registros de $\Pi_{\hat{A}}\mathbf{R}$ que en la tabla $\mathbf{R}$ aparecen asociados a todos y cada uno de los valores de $\Pi_B\mathbf{S}$.

Su formulación en términos de operadores de conjuntos es

$$\Pi_{\hat{A}}\mathbf{R} \div \Pi_B\mathbf{S} = \Pi_{\hat{A}-B}\mathbf{R} - (\Pi_{\hat{A}-B}((\Pi_{\hat{A}-B}\mathbf{R} \times \Pi_B\mathbf{S}) - \Pi_{\hat{A}}\mathbf{R}))$$

Observación: Esta operación es muy útil para identificar registros que verifican propiedades o patrones determinados, como por ejemplo, clientes que coinciden en la selección de productos, o pacientes con parámetros de salud semejantes.

Reunificación mediante álgebra relacional y de conjuntos.

Ejemplo: Oferta de un operador multimedia y selección (división) de tarifas que ofrecen cierto conjunto de contenidos. Los conjuntos de atributos para procesar la división son A={tarifa,calidad}, B={contenido} y $\hat{A}$={tarifa}:

R: oferta mutimedia

tarifa	**contenido**	**calidad**
premium	TV	cable
enjoy	TV	cable
basic	TV	cable
premium	series	HD
enjoy	series	SD
premium	cine	HD

$\div$

S: contenido

contenido
TV
series

$=$

$\Pi_{\hat{A}}\mathbf{R} \div \Pi_{B}\mathbf{S}$:

tarifa
premium
enjoy

Introducción a las transacciones.

Definición y propiedades de las transacciones:

Una transacción es una secuencia de operaciones sobre la BBDD que equivale a una unidad de trabajo simple. Sus propiedades son:

- **Atomicidad:** equivalencia con operación única.
- **Aislamiento:** durante la ejecución de la transacción todos los datos leídos desde la BBDD se utilizarán sin influencias de otras transacciones ejecutadas paralelamente.
- **Consistencia:** un estado admisible de la BBDD se transforma en otro estado admisible.
- **Durabilidad:** el resultado de una transacción exitosa se graba en la BBDD y el de una fallida se resuelve con la anulación de las instrucciones y el retorno al anterior estado admisible.

Un ejemplo de transacción es la lectura de datos numéricos, cierto procesamiento aritmético y la escritura de resultados.

Introducción a las transacciones.

Encapsulamiento de las transacciones:

En la sintaxis de la transacción se deben indicar ciertos elementos:

- **start transaction:** comienzo a partir del cual la lectura y procesamiento de datos se mantendrá independiente de otras operaciones externas en la BBDD.
- **savepoint:** eventuales puntos intermedios de grabación para almacenar resultados que deben consolidarse a pesar de que la transacción sea fallida.
- **rollback:** retorno al anterior estado admisible de la BBDD cuando la transacción no pueda continuar exitosamente.
- **commit:** etiqueta que marca la finalización y la liberación de los recursos bloqueados durante la transacción.

Introducción a las transacciones.

Ejemplo de transacción:

```
START TRANSACTION reserva_billete;
  USE traficoaereo;
  SET @ia = (SELECT id_nv FROM nave NATURAL JOIN
  rutas WHERE rutas.date = 24-12-2025);
  SET @np = (SELECT n_pl FROM nave WHERE id_nv = @ia);
  IF @np > 0 THEN
    INSERT INTO viaj(name,pass) VALUES(@name,@pass);
    UPDATE nave SET n_pl = n_pl-1 WHERE id_nv = @ia;
  ELSE
    ROLLBACK TRANSACTION reserva_billete;
  END IF
COMMIT;
```

La transacción graba los datos de un viajero y actualiza plazas de un vuelo si hay disponibilidad y, si no hay, se aborta.

Introducción a las transacciones.

Estados de las transacciones:

Una transacción a lo largo de su proceso se puede encontrar en alguno de los siguientes estados:

- **Activo:** estado que sucede a la etiqueta inicial.
- **Parcialmente cometido:** estado que sucede a la ejecución de la última instrucción.
- **Cometido:** estado admisible posterior a la grabación de datos resultantes de la ejecución.
- **Fallado:** instante en el que no se puede continuar el desarrollo lógico de la transacción por diferentes causas (condiciones incumplidas, incompatibilidad de datos, inconsistencia de la información, división por 0, salto de bloqueos, etc).
- **Abortado:** estado admisible, anterior al comienzo de la transacción, y que, tras un fallo, se ha restaurado.

Introducción a las transacciones.

Modelos de transacciones:

El modelo "*ANSI/ISO*" incorpora solo las instrucciones "`commit`" para volcar los resultados en la BBDD tras una ejecución exitosa y "`rollback`" para el retorno a un estado admisiblede la BBDD y previo al comienzo de la transacción tras un fallo.

El modelo "*transact-SQL*" incorpora además la instrucción "`rollback transaction`" para retornar a un punto intermedio etiquetado mediante "`savepoint`". En cualquier caso, en este modelo no se permiten alteraciones estructurales de la BBDD como la creación y modificación de tablas.

Observación: Las tablas en MySQL se generan, por defecto, bajo la declaración "*InnoDB*", que equivale al modelo "*transact-SQL*", con posibilidad de puntos intermedios de grabación. Para forzar esta funcionalidad se puede declarar "`ENGINE = InnoDB`".

Procesos multiusuario. Planificación de transacciones.

Procesos multiusuario:

Para que las transformaciones realizadas sobre la BBDD desemboquen en un estado consistente, desde el gestor de BBDD se debe planificar la secuencia de *procesos concurrentes*, es decir, los que podrían solaparse en el tiempo debido al instante cuya ejecución ha sido ordenada. Para evitar que procesos ejecutados simultaneamente interfieran entre ellos, se puede aislar o bloquear temporalmente los datos.

Procesos multiusuario. Planificación de transacciones.

Planificación de transacciones:

Las transacciones *concurrentes* multiusuario se pueden planificar:

- **En serie:** no se ejecutan simultaneamente transacciones diferentes, sino que el comienzo de una sucede al final de la anterior. La ejecución se ralentiza, pero se garantiza la consistencia de la BBDD.
- **Concurrentemente:** se paraleliza la ejecución de diferentes transacciones. Pueden darse problemas de inconsistencia y perdida de lógica en sentencias de lectura y escritura. Para evitarlo se deben emplear estrategias de serialización, que consisten en agrupar bloques de instrucciones en cada transacción que sean independientes de otros bloques de otras transacciones y lanzar simultaneamente su ejecución.

Procesos multiusuario. Planificación de transacciones.

Ejemplo de transacción concurrida

T_1	T_2
LEER A	LEER A
A = A - 50000	temp = 0.1*A
ESCRIBIR A	A = A - temp
LEER B	ESCRIBIR A
B = B - 50000	LEER B
ESCRIBIR B	B = B + temp
	ESCRIBIR B

Plan 1 en serie

T_1	T_2
LEER A	
A = A - 50000	
ESCRIBIR A	
LEER B	
B = B - 50000	
ESCRIBIR B	
	LEER A
	temp = 0.1*A
	A = A - temp
	ESCRIBIR A
	LEER B
	B = B + temp
	ESCRIBIR B

Plan 2 en serie

T_1	T_2
	LEER A
	temp = 0.1*A
	A = A - temp
	ESCRIBIR A
	LEER B
	B = B + temp
	ESCRIBIR B
LEER A	
A = A - 50000	
ESCRIBIR A	
LEER B	
B = B - 50000	
ESCRIBIR B	

Plan 3 concurrente

T_1	T_2
LEER A	
A = A - 50000	
ESCRIBIR A	
	LEER A
	temp = 0.1*A
	A = A - temp
	ESCRIBIR A
LEER B	
B = B - 50000	
ESCRIBIR B	
	LEER B
	B = B + temp
	ESCRIBIR B

Concurrencia. Inconvenientes y soluciones. Bloqueos.

Se pueden dar diversos escenarios en los que una incorrecta planificación del procesamiento en paralelo de transacciones puede desembocar en resultados indeseados.

Problemas de concurrencia I. Lectura sucia.

T_1	T_2
BEGIN TRANSACTION UPDATE cuenta SET saldo = saldo - 10 WHERE n_cta=25	
	BEGIN TRANSACTION SELECT SUM(saldo) FROM cuenta WHERE n_cta<50 COMMIT TRANSACTION
UPDATE cuenta SET saldo = saldo + 10 WHERE n_cta = 45 COMMIT TRANSACTION	

T2 accede al servidor antes de que T1 grabe las modificaciones realizadas. El resultado de la operación no refleja la realidad.

Concurrencia. Inconvenientes y soluciones. Bloqueos.

Problemas de concurrencia II. Lectura sucia.

T_1	T_2
BEGIN TRANSACTION UPDATE cuenta SET saldo = saldo - 10 WHERE n_cta=25	
	BEGIN TRANSACTION SELECT SUM(saldo) FROM cuenta WHERE n_cta<50 COMMIT TRANSACTION
IF saldo < 0 THEN ROLLBACK TRANSACTION END IF COMMIT TRANSACTION	

T2 accede al servidor y realiza unas operaciones con unos datos a los que T1 ha aplicado temporalmente unas modificaciones que posteriormente han sido anuladas. El resultado de la operación no refleja la realidad.

Concurrencia. Inconvenientes y soluciones. Bloqueos.

Problemas de concurrencia III. Lectura no repetible.

T_1	T_2
BEGIN TRANSACTION SELECT saldo FROM cuenta WHERE n_cta = 25	
	BEGIN TRANSACTION UPDATE cuenta SET saldo = saldo -10 FROM cuenta WHERE n_cta = 25 COMMIT TRANSACTION
SELECT saldo FROM cuenta WHERE n_cta = 25 COMMIT TRANSACTION	

T1 accede dos veces a la cuenta para leer el mismo dato, que ha sido modificado por T2 en el intervalo de tiempo entre ambas lecturas, con lo que hay discrepancia en el valor de un dato.

Concurrencia. Inconvenientes y soluciones. Bloqueos.

Problemas de concurrencia IV. Lectura fantasma.

T_1	T_2
BEGIN TRANSACTION SELECT * FROM cuenta WHERE n_cta $<$ 25	
	BEGIN TRANSACTION UPDATE cuenta DELETE * FROM cuenta WHERE n_cta BETWEEN 10 AND 15 COMMIT TRANSACTION
SELECT * FROM cuenta WHERE n_cta $<$ 25 COMMIT TRANSACTION	

T1 realiza dos veces cierta búsqueda sobre un conjunto de filas, pero entre ambas lecturas T2 ha modificado el conjunto de filas, con lo que la búsqueda difiere.

Concurrencia. Inconvenientes y soluciones. Bloqueos.

Problemas de concurrencia V. Desaparción de operaciones.

T_1	T_2
BEGIN TRANSACTION SET @a1 = (SELECT saldo FROM cuenta WHERE n_cta = 25)	
	BEGIN TRANSACTION SET @a2 = (SELECT saldo FROM cuenta WHERE n_cta = 25) SET @a2 = 2*@a2 UPDATE cuenta SET saldo = @a2 WHERE n_cta = 25 COMMIT TRANSACTION
SET @a1 = 1 + @a1 UPDATE cuenta SET saldo = @a1 WHERE n_cta = 25 COMMIT TRANSACTION	

T1 accede a un registro para ser procesado, pero antes de esto último, T2 acede al mismo registro, lo modifica en su memoria temporal y lo graba. Luego la transacción T1 finaliza sus operaciones y graba su versión de este registro quedando sin validez la salida de T2.

Concurrencia. Inconvenientes y soluciones. Bloqueos.

Problemas de concurrencia VI. Escritura inconsistente.

T_1	T_2
BEGIN TRANSACTION SET @a1 = (SELECT saldo FROM cuenta WHERE n_cta = 25) SET @a1 = 1 + @a1 SELECT @a1	
	BEGIN TRANSACTION SET @a2 = (SELECT saldo FROM cuenta WHERE n_cta = 25) SELECT @a2 SET @b2 = (SELECT dispon FROM tarjeta WHERE n_cta = 25) SELECT @b2 COMMIT TRANSACTION
SET @b1 = (SELECT dispon FROM tarjeta WHERE n_cta = 25) SET @b1 = 1 + @b1 SELECT @b1 COMMIT TRANSACTION	

Dos datos coincidentes y almacenandos en dos tablas son leidos por T1 para idénticamente modificados, pero tras modificarse el primero, T2 también lee ambos datos encontrando discrepancia.

Concurrencia. Inconvenientes y soluciones. Bloqueos.

Garantizar la consistencia I. Recuperabilidad.

T_1	T_2
BEGIN TRANSACTION	
LEER a	
	BEGIN TRANSACTION
	LEER a
MODIFICAR a	
GRABAR a	
	IF condición ROLLBACK TRANSACTION
	COMMIT TRANSACTION
LEER b	
COMMIT TRANSACTION	

Se debe evitar que dos transacciones, T1 y T2, lean el mismo dato "a", que T2 lo lea antes de que T1 lo haya modificado y grabado y que, además, T2 finalice con posibilidad de *ROLLBACK* antes que T1. En el caso de que se aborte T2, se retorna al anterior estado admisible, perdiéndose el resultado de T1.

Concurrencia. Inconvenientes y soluciones. Bloqueos.

Garantizar la consistencia II. Abortos en cascada.

T_1	T_2	T_3
BEGIN TRANSACTION LEER a MODIFICAR a GRABAR a		
	BEGIN TRANSACTION LEER a MODIFICAR a GRABAR a	
		BEGIN TRANSACTION LEER a MODIFICAR a GRABAR a
IF cond1 ROLLBACK COMMIT TRANSACTION		
	IF cond2 ROLLBACK COMMIT TRANSACTION	
		IF cond3 ROLLBACK COMMIT TRANSACTION

Hay que evitar los procesos solapados "*en cascada*" ya que la recuperación de datos tras un *ROLLBACK* anularía las operaciones realizadas en sucesivas transacciones posteriores.

Concurrencia. Inconvenientes y soluciones. Bloqueos.

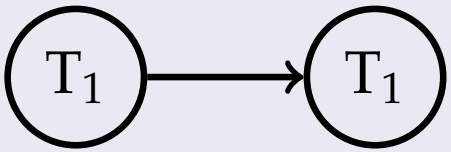
Serialización de procesos I. Definiciones.

- **Gránulo:** Unidad simple de información.
- **Acción:** Manipulación básica del gránulo (inserción, borrado, modificación, actualización), f_i. Dos acciones, f_1, f_2, son **permutables** si al ejecutarlas el resultado es indiferente del orden en que actúen, por ejemplo dos lecturas de un gránulo.
- **Transacción:** Secuencia de acciones, T=$\{f_1, f_2, \ldots, f_n\}$, ejecutadas sobre la BBDD que preservan su consistencia.
- **Plan de ejecución:** Secuencia $\Lambda = \{f_1, f_2, g_1, \ldots\}$ de acciones ordenadas con respecto a cada transacción del conjunto $\mathbf{C} = [T_1, T_2]$, $\mathrm{T}_1 = \{f_1, f_2, \ldots\}$, $\mathrm{T}_2 = \{g_1, g_2, \ldots\}$.
- **Plan serial:** Plan de ejecución Λ que coincide con una permutación de $\mathbf{C}$. Si el resultado de un plan de ejecución $\hat{\Lambda}$ coincide con el de uno serial Λ, entonces $\hat{\Lambda}$ es **serializable**.

Concurrencia. Inconvenientes y soluciones. Bloqueos.

Serialización de procesos II. Representación mediante grafos

Dadas dos transacciones, $T_i = \{f_1, f_2, \ldots\}$ y $T_j = \{g_1, g_2, \ldots\}$ en un plan de ejecución Λ donde, cada acción de lectura o escritura g_m viene precedida de una acción f_n, siendo alguna de las dos de escritura, entonces se dibuja un grafo dirigido de T_i a T_j.



T_1	T_2
BEGIN TRANSACTION	
	BEGIN TRANSACTION
LEER a	
ESCRIBIR a	
LEER b	
ESCRIBIR a	
	LEER a
	ESCRIBIR a
	LEER b
	ESCRIBIR b
COMMIT TRANSACTION	
	COMMIT TRANSACTION

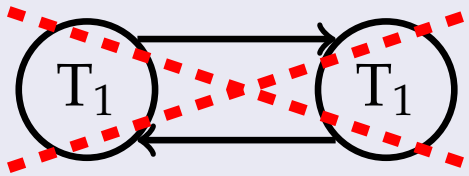
Concurrencia. Inconvenientes y soluciones. Bloqueos.

Serialización de procesos III. Relaciones de precedencia.

Dadas dos transacciones, $\mathrm{T}_i = \{f_1, f_2, \ldots\}$ y $\mathrm{T}_j = \{g_1, g_2, \ldots\}$ en un plan de ejecución Λ la transacción T_i precede a T_j y se escribe $\mathrm{T}_i < \mathrm{T}_i$ cuando existen acciones no permutables f_i y g_j, donde la ejecución de f_i es necesariamente anterior a g_j.

Consecuentemente se establece un grafo con un arco de T_i a T_j.

Para que Λ sea serializable es suficiente que no haya ciclos en el grafo del conjunto de transacciones $\mathbf{C} = [\mathrm{T}_i, \mathrm{T}_j, \ldots]$.



Concurrencia. Inconvenientes y soluciones. Bloqueos.

Bloqueos I. Definición.

Los gestores de bases de datos incorporan la posibilidad de realizar bloqueos a tablas o a filas. Un bloqueo es un cierre temporal del aceso a una tabla o fila cuyos datos están siendo consultados y procesados por una transacción. Así se evita que otras transacciones utilicen y modifiquen datos antes de que la primera los haya grabado y traslade la BBDD a un estado consistente.

T_1	T_2 (propuesta)	T_2 (real)
LEER a		
	LEER b	LEER b
LEER c		
BLOQUEAR c		
	ESCRIBIR b	ESCRIBIR b
	LEER c	
ESCRIBIR a		
ESCRIBIR c		
DESBLOQUEAR c		
	ESCRIBIR c	LEER c
		ESCRIBIR c

Concurrencia. Inconvenientes y soluciones. Bloqueos.

Bloqueos II. Bloqueos recurrentes.

Los bloqueos recurrentes o "*abrazos de la muerte*" entre dos o más transacciones suponen un obstáculo. Se trata del intento por parte de una transacción de acceder a unos datos bloqueados por otra transacción que, a su vez, necesita acceder a datos bloqueados por la primera. Ambas transacciones quedarían en espera indefinida. Se determinan criterios de ordenación "*timestamp*" para desbloquear el flujo de ejecución priorizando que una transacción continue sin interferencias o sea abortada.

Ejemplo de bloqueo recurente o abrazo de la muerte

T_1	T_2
LEER a	
BLOQUEAR a	
	LEER b
	BLOQUEAR b
LEER b	
	LEER a
⋮	⋮

Concurrencia. Inconvenientes y soluciones. Bloqueos.

Bloqueos III. Formulación del flujo de ejecución.

Consideremos el gránulo g al que se accede en la transacción T_r. Ante el propósito de acceso a g por otra transacción T_s se utiliza la formulación afín al Álgebra de Boole para ordenar el flujo de transacciones concurrentes.

- Al gránulo g se le asigna un vector $A(g)$ de estados, con componentes $a_i \in \{1 \text{ (bloqueado)}, 0 \text{ (desbloqueado)}\}$, $\forall i = 1, \ldots, k$, asociados a cada operación $f_i(g) \in \{$*lectura*, *lectura protegida*, *escritura*, *actualización*, *eliminación*, ...$\}$.
- El vector $B = \{b_1, \ldots, b_k\}$ determina las solicitudes de bloqueo/desbloqueo sobre el gránulo g en cierto instante.
- Matriz de compatibilidad entre estados $\begin{pmatrix} c_{11} & \cdots & c_{1k} \\ \vdots & \ddots & \vdots \\ c_{k1} & \cdots & c_{kk} \end{pmatrix}$

Concurrencia. Inconvenientes y soluciones. Bloqueos.

Bloqueos V. Algebra de Boole.

Las componentes de la matriz de compatibilidades C valen $c_{ij} = 1$ si los estados a_i y a_j son compatibles en dos transacciones diferentes, es decir, si una transacción T_r puede ejecutar $f_i(g)$ y simultáneamente T_s puede ejecutar $f_j(g)$. De la misma manera, la matriz $\neg C$, complementaria de C según la notación de Morgan, tiene por componentes $\hat{c}_{ij} = \mod(c_{ij} + 1, 2)$.

Considerense, por ejemplo, las operaciones *lectura* y *escritura*, solo compatible la primera consigo misma. Entonces las respectivas matrices de compatibilidades y su complementaria son

$$C = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \quad \text{y} \quad \neg C = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$$

Concurrencia. Inconvenientes y soluciones. Bloqueos.

Bloqueos VI. Algebra de Boole, notación.

Consideramos un retículo distributivo $(\mathcal{B}, \odot, \otimes)$ compuesto por el conjunto $\mathcal{B}$ que contiene, al menos el conjunto vacio $\emptyset$ y el conjunto completo $\mathcal{U}$, en este caso, se toman $\emptyset = 0$ y $\mathcal{U} = 1$.

- Se definen las operaciones internas $\otimes, \oplus : \mathcal{B} \times \mathcal{B} \to \mathcal{B}$. El producto booleano $\otimes$ equivale a la intersección $(\wedge)$ y la suma booleana a la $\oplus$ unión $(\vee)$.

$\otimes$	0	1
0	0	0
1	0	1

$\oplus$	0	1
0	0	1
1	1	1

- La inclusión lógica se representa por $\preceq$, cumpliéndose, $0 \preceq 1$.
- El complementario o negación de un estado a se escribe $\neg a$.

Concurrencia. Inconvenientes y soluciones. Bloqueos.

Bloqueos VII. Multiplicación matricial booleana.

Dadas dos matrices numéricas $P \in \mathcal{M}_{m\times k}$ y $Q \in \mathcal{M}_{k\times n}$, las componentes h_{ij} de la multiplicación booleana $H = P \otimes Q \in \mathcal{M}_{m\times n}$ son

$$hij = p_{i1} \cdot q_{1j} \oplus \cdots \oplus p_{ik} \cdot q_{kj} = \max\{p_{i1} \cdot q_{1j}, \ldots, p_{ik} \cdot q_{kj}\}.$$

En particular, para una matriz cuadrada de compatibilidades $C \in \mathcal{M}_{k\times k}$ y un vector de estados $A \in \mathbb{R}^k$, las componentes h_i de $H = C \otimes A \in \mathbb{R}^k$ son

$$h_i = c_{i1} \cdot a_1 \oplus \cdots \oplus c_{ik} \cdot a_k = \max\{c_{i1} \cdot a_1, \ldots, c_{ik} \cdot a_k\}.$$

Concurrencia. Inconvenientes y soluciones. Bloqueos.

Para determinar los bloqueos se analiza el producto booleano

$$\neg C \otimes A(g) \in \mathbb{R}^k.$$

Cada componente $\hat{c}_{ij}$ asociada a estados incompatibles f_i y f_j vale 1 en la matriz complementaria $\neg C$ y el resto son 0. La acción f_i no se podrá acometer si algún f_j incompatible está bloqueado, $a_j = 1$. Esto equivale a que la componente i del producto booleano $\neg C$ por $A(g)$ vale $\hat{c}_{i1} \cdot a_1 \otimes \cdots \otimes \hat{c}_{ik} \cdot a_k = 1$.

Las acciones acometibles serán las f_i con un $\hat{h}_i = 0$ asociado o equivalentemente con componente 1 en el complementario booleano $\neg\big(\neg C \otimes A(g)\big)$.

Concurrencia. Inconvenientes y soluciones. Bloqueos.

Bloqueos VIII. Formulación del flujo de ejecución.

Al arrancar la secuencia de transacciones, $\mathrm{T}_1, \mathrm{T}_2, \ldots$, se inicializa el vector de estados, $A_0(g) = \{a_1, \ldots, a_k\}$, sobre cada gránulo g utilizado por T_1. Posteriormente cada transacción T_r ejecuta el respectivo bloqueo de tablas o filas, *LOCK TABLES(*T_r*,B,g)*, y al finalizar la transacción el desbloqueo, *UNLOCK TABLES(*T_r*,B,g)*.

A lo largo de la secuencia de transacciones las solicitudes de bloqueo sobre un gránulo se ejecutan si son compatibles con el estado y la matriz de compatibilidades asociadas al gránulo. La condición para permitir que una transacción acometa una operación y realizar los bloqueos necesarios es la siguiente:

$$B \preceq \Big(\neg\big(\neg \mathrm{C} \otimes A(g)\big)\Big)$$

Concurrencia. Inconvenientes y soluciones. Bloqueos.

Ejemplo: Considerar cuatro acciones o modos {leer, escribir, insertar, eliminar} y las siguientes matriz de compatibilidad C, vector inicial de estados $A(g)$ y solicitud de bloqueos B. Deducir si B se puede acometer inmediatamente.

$$C = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \quad A(g) = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \quad B = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

La propuesta de bloqueos es factible, ya que se cumple la inclusión

$$\neg C \otimes A(g) = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \end{pmatrix} \Rightarrow B = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \preceq \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} = \neg\left(\neg C \otimes A(g)\right).$$