ntic master
revolucionamos la comunicación

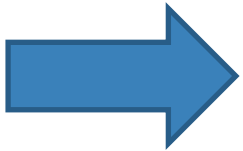
UNIVERSIDAD COMPLUTENSE MADRID

Bases de Datos

Isabel Riomoros

1 .- Introducción a las Bases de Datos

2.- Modelo Entidad-Relación

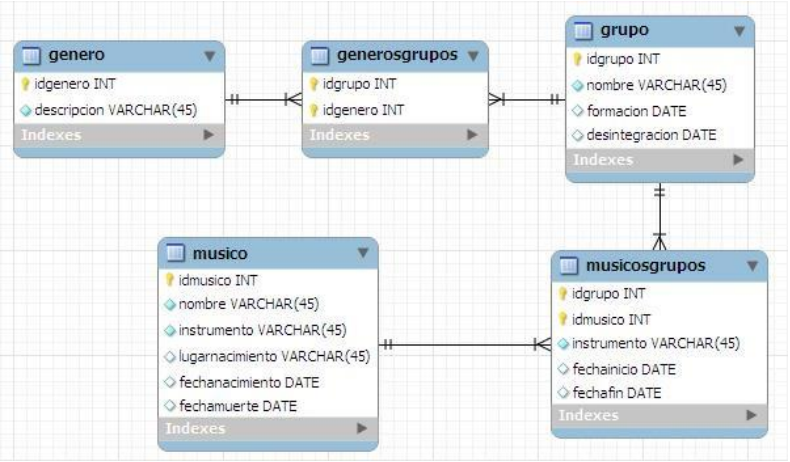


3.- Modelo Relacional

4.- SQL

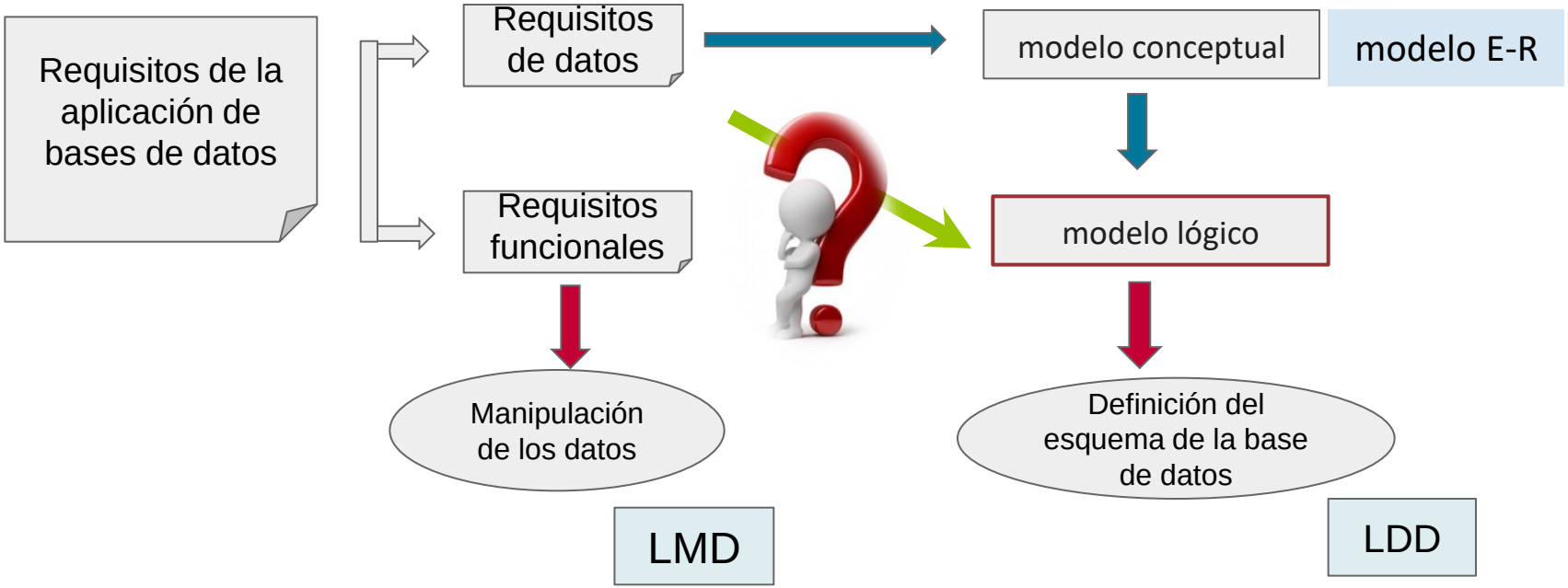
3

Modelo Relacional



El modelo de datos relacional organiza y representa la información en forma de tablas o relaciones, para representar tanto los datos como las relaciones entre ellos. Su simplicidad conceptual ha conducido a su adopción generalizada.

Proceso de diseño de una Base de Datos



Modelo de datos

Modelo de datos es un conjunto de herramientas **conceptuales** para la descripción de los **datos**, las **relaciones** entre ellos, su semántica y las restricciones de consistencia.

Los modelos de datos tienen que ver con tres aspectos de los datos:

- Estructura de los datos
- Integridad de los datos
- Manejo de los datos

Objetivo: aislar al usuario de las estructuras físicas de los datos

independencia de las aplicaciones respecto de los datos

Descripción del modelo

- Fue introducido por Codd (1979)
- Modelo lógico de datos de no muy alto nivel, orientado al registro. **Gran simplicidad**.
- **Sólida base teórica:** la teoría de conjuntos y la lógica de predicados de primer orden
- Implementado en muchos SGBD
- Cada base de datos es un conjunto de relaciones, cada una de las cuales es una tabla con filas y columnas
- El concepto principal es **la relación o tabla**

Descripción del modelo

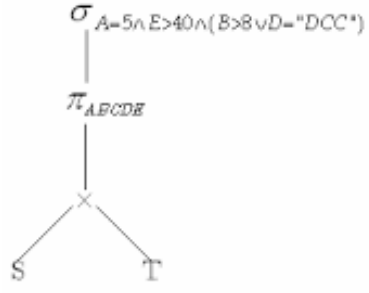
codArticulo	denom	precio	unidades
0001	Ord. Sobremesa	600.00	12
0002	Ord. Portátil	1000.00	6
0003	Tarjeta Red	20.00	25
0004	Impresora Láser	200.00	4
0005	Ratón USB	7.00	50
0006	Monitor TFT	250.00	10
0007	Router inalámbrico	100.00	30

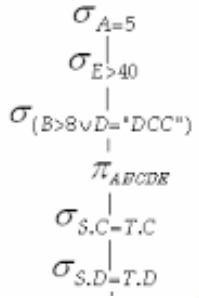
Sencillez y uniformidad: por estar basado en tablas es una representación uniforme y sencilla de manejar.

Sólida fundamentación teórica: el modelo está definido con rigor matemático, el diseño y la evaluación del mismo pueden hacerse por métodos sistemáticos basados en abstracciones.

de la teoría de conjuntos		propios relacionales	
∪	Unión	σ DONDE	Selección
∩	Intersección	Π [ ]	Proyección
~ -	Diferencia	⋈ ∞	Concatenación (join)
×	Producto cartesiano	÷	División

Independencia del interfaz del usuario: los lenguajes relacionales, al manipular conjuntos de registros, proporcionan una gran independencia respecto a la forma en la que los datos están almacenados.





Descripción del modelo

- Una **base de datos relacional** es un conjunto de **tablas**, a cada una de las cuales se le asigna un nombre exclusivo.
- Cada **fila** de la **tabla** representa una colección de valores de datos relacionados entre sí, se le denomina **tupla.** Esos valores se pueden interpretar como hechos que describen una entidad o un vínculo entre entidades del mundo real.
- Todos los valores de una columna tienen el mismo tipo de datos

Atributos

- Las cabeceras de las tablas son los atributos
- Cada atributo tiene un nombre
- Un atributo está asociado siempre a una relación y representa una propiedad de esta
- Varios atributos distintos pueden tomar sus valores dentro del mismo dominio
- ***Se requiere que los atributos sean atómicos*** (no son atómicos los compuestos y los multivalorados)

Dominio

- *Conjunto finito y homogéneo de valores atómicos*.
- Conjunto de valores permitidos para un atributo
- Un dominio debe tener un **nombre**, un número **finito** de valores **homogéneos** y tiene que tener un **valor atómico**, no puede ser un conjunto de valores.

Descripción del modelo

- **Esquema de relación**: nombre de relación **R**, más un conjunto de **n** atributos $\mathbf{A_i}$ definidos en sendos dominios $\mathbf{D_i}$ (no necesariamente distintos).

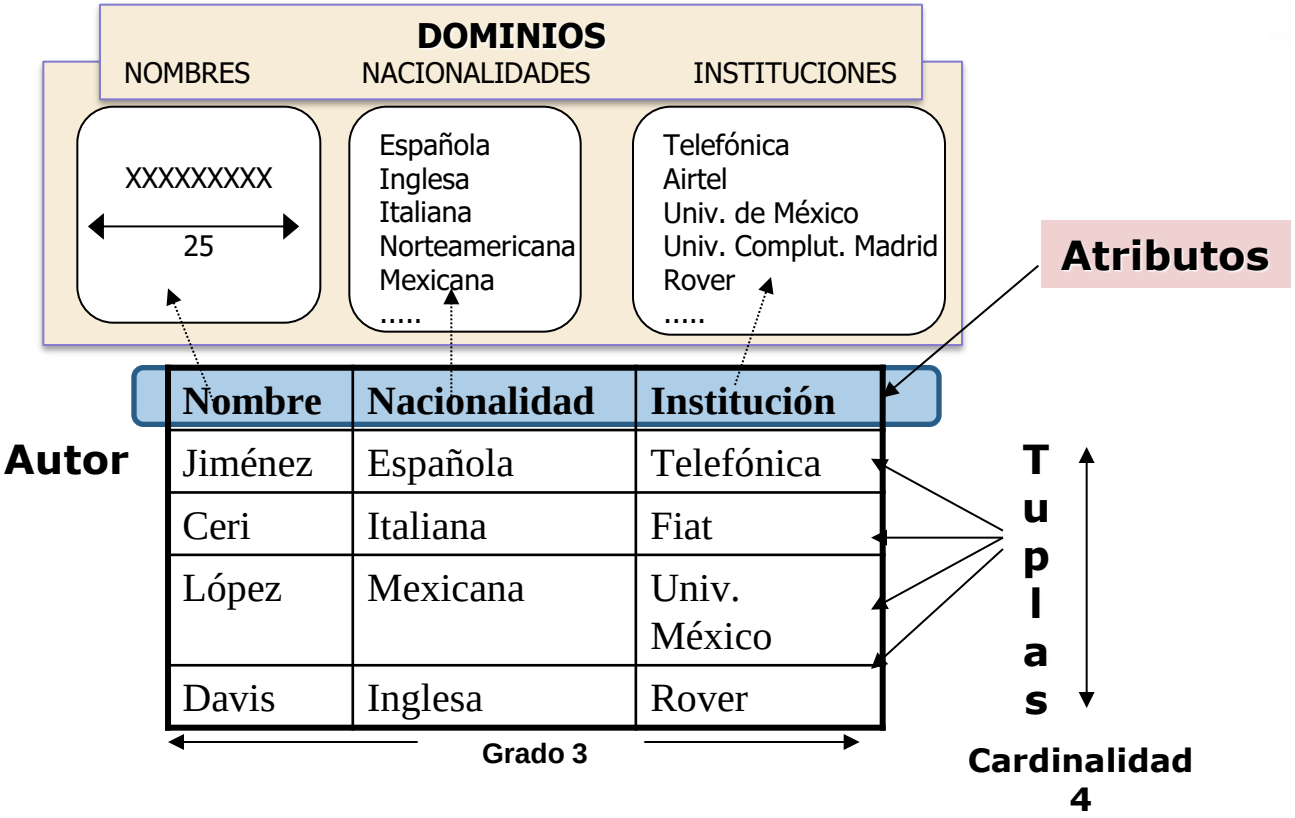
- **Relación o estado de la relación**: conjunto de m elementos denominados **tuplas** t_j. Cada tupla es un conjunto de n pares (A_i, v_{ij}) donde A_i es un atributo y v_{ij} es el valor de la tupla t_j para ese atributo. La relación o estado de la relación habitualmente se denomina r(R).

- El número de tuplas de una relación en un momento concreto se denomina **cardinalidad** de la relación.

- **Instancia de una relación** Conjunto de tuplas (filas) que, en un instante determinado, satisfacen el esquema de relación y se encuentran almacenadas en la base de datos.

- **Esquema de una base de datos** = conjunto de esquemas de relación
- **Estado de la base de datos** = conjunto de los estados de las relaciones

Descripción del modelo

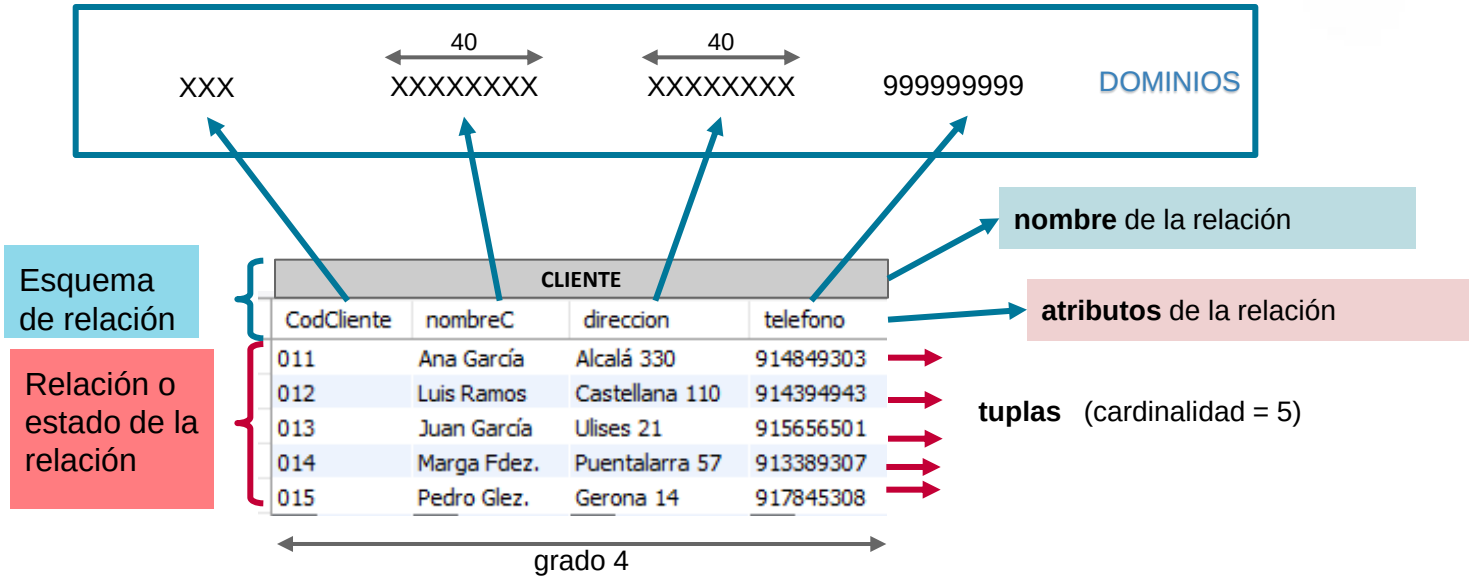


Ejemplo: Relación cuenta

Atributo	Dominio	
Nombre-sucursal	El conjunto de los nombres de las sucursales	D_1
Número-cuenta	El conjunto de números de cuenta	D_2
Saldo	El conjunto de los saldos	D_3

- Cuenta sólo contendrá un **subconjunto** del conjunto de todas las filas posibles
- Cuenta es un subconjunto de D_1 x D_2 x D_3
- Los matemáticos definen las relaciones como subconjuntos del producto cartesiano de la lista de dominios, se corresponde con la definición de tabla

Descripción del modelo, ejemplo



Descripción del modelo

No confundir los conceptos de tabla y relación

- Una tabla **es una forma de** representar una relación
- Una relación tiene unas propiedades intrínsecas que no tiene una tabla, propiedades que se derivan de la misma definición matemática de relación (se trata de un conjunto)

En una relación:

- **No** puede haber **dos tuplas iguales**
- El orden de la tuplas no es significativo
- El orden de los atributos no es significativo
- Cada **atributo sólo puede tomar un único** valor del dominio; no se admiten grupos repetitivos

Descripción del modelo

- Una base de datos consta de **múltiples relaciones**
- La información sobre una empresa se divide en partes, en la que cada relación almacena una parte de la información

cuenta : Almacena la información de cuentas
impositor: Almacena información de los clientes con sus cuentas
cliente: Almacena información sobre las cuentas

- Si almacenamos toda la **información como una única relación**

***banco*(*número_cuenta, saldo, nombre_cliente*, …)** tenemos

Repetición de información (ej., dos clientes comparten una cuenta)

Necesidad de valores null (ej., representar un cliente sin ninguna cuenta)

La **teoría de normalización** trata de cómo diseñar esquemas de relación

Claves

Clave candidata *es un conjunto no vacío de atributos que identifican* ***unívoca*** *y* ***mínimamente*** *cada tupla de una relación.*

En toda relación hay al menos una clave candidata, los atributos de la relación tendrán la **propiedad de unicidad** (por la definición de relación no puede haber dos tuplas iguales)

Clave primaria es *la clave candidata que el usuario elegirá, por consideraciones ajenas al modelo relacional, para identificar las tuplas de la relación.* Los atributos que forman parte de la clave primaria no pueden ser nulos.

Las **Clave alternativas** son aquellas claves candidatas que no han sido elegidas como claves primarias en la relación.

Clave ajena (externa) es una clave ajena de una relación R_2 es un conjunto no vacío de atributos cuyos valores han de coincidir con los valores de la clave primaria de una relación R_1.

Elegir la clave primaria

- **Unicidad:** debe garantizar que cada fila en la tabla tenga un valor único que la identifique de manera inequívoca. Evita duplicados y asegura que no haya ambigüedad en los datos.
- **Inmutabilidad**: debería ser inmutable, es decir, no debería cambiar con el tiempo. Ayuda a mantener la integridad de los datos y evita problemas de actualización.
- **Simplicidad:** debe ser lo más simple posible. Esto facilita la lectura, escritura y comprensión de los datos. A veces, se utilizan enteros autoincrementales como claves primarias simples.
- **Significado**: En algunos casos, una clave primaria que tenga un significado semántico puede ser útil para comprender mejor los datos. Sin embargo, debe tenerse cuidado para asegurarse de que siga siendo única y que no cambie.

Elegir la clave primaria

- Tamaño y rendimiento: El tamaño puede afectar el rendimiento de las consultas y el almacenamiento.
- Relaciones: puede depender de las relaciones con otras tablas. Debe ser fácilmente utilizable como clave externa en otras tablas para establecer relaciones.
- Facilitar las consultas: La clave primaria debe facilitar la realización de consultas y la recuperación de datos. Debe ser fácil de usar en las consultas habituales.
- Escalabilidad: para acomodar el crecimiento futuro de la base de datos sin problemas.
- Cumplir con estándares: En algunos casos, es importante cumplir con estándares de la industria o regulaciones que pueden dictar el tipo de clave primaria que se debe utilizar

Ejemplos clave primaria

- Tabla de empleados. número de identificación único como clave primaria, es fácil de entender, única para cada empleado y generalmente inmutable a lo largo del tiempo.

- En una tabla que registra las relaciones entre estudiantes y cursos, podrías tener una clave primaria compuesta que combina el ID del estudiante y el ID del curso. En este caso, la clave primaria se compone de dos columnas para asegurar que no haya duplicados en la combinación estudiante-curso.

- Información sobre libros de una biblioteca, podrías utilizar el número ISBN del libro como clave primaria. El ISBN tiene un significado semántico y es único para cada libro. Sin embargo, debe asegurarse de que no cambie, ya que es una identificación externa al sistema.

Ejemplos de clave primaria

- Pedidos de clientes, puedes usar un ID autonumérico como clave primaria. Esta clave se genera automáticamente y es simple de administrar, ya que no requiere intervención del usuario.

- Tenemos una tabla de clientes y una tabla de pedidos. Puedes utilizar el ID del cliente como clave primaria en la tabla de clientes y luego utilizarlo como clave externa en la tabla de pedidos para establecer una relación entre las dos tablas. Esto permite relacionar cada pedido con un cliente específico.

Malas elecciones de clave primaria

- Utilizar el nombre del empleado como clave primaria. Esto no sería inmutable, ya que los empleados pueden cambiar sus nombres debido a matrimonio u otros motivos. Además, los nombres no son necesariamente únicos.

- Usar el nombre del estudiante y el nombre del curso como clave primaria en una tabla de inscripciones. Esto sería poco eficiente y propenso a errores, ya que los nombres no son valores únicos ni inmutables.

- Utilizar el número de teléfono de un cliente como clave primaria. Los números de teléfono pueden cambiar con el tiempo o no ser únicos, ya que varias personas pueden compartir un número de teléfono.

Artículo

idArticulo	nomArticulo	precio	unidades	descuento	precioFinal
0001	Ordenador Sobremesa	600.00	12	20.00	480.00
	Ordenador Portátil	1000.00	6	0.00	0.00
0003	Tarjeta Red	20.00	50	0.00	0.00
0004	Impresora Láser	200.00	4	0.00	0.00
0005	Ratón USB	7.00	50	0.00	0.00
0006	Monitor TFT	250.00	10	0.00	0.00
0007	Router inalámbrico	100.00	30	0.00	0.00
0008	altavoz	100.00	30	0.00	0.00
0009	Teclado Mecánico	50.00	40	0.00	0.00
0010	Disco Duro Externo	80.00	25	0.00	0.00
0011	Memoria USB 64GB	15.00	100	0.00	0.00
012	[illegible]	45.00	20	0.00	0.00

Clave ajena de la relación articulo

Compra

idCliente	idArticulo	fecCompra	numUnidades
011	0001	2016-10-06	1
011	0005	2017-10-06	2
012	0001	2018-11-06	3
012	0002	2019-11-06	1
012	0003	2018-11-06	3
012	0004	2018-11-06	3
012	0005	2022-11-06	3
012	0006	2018-12-06	3
012	0007	2018-11-06	3
012	0008	2018-12-06	3

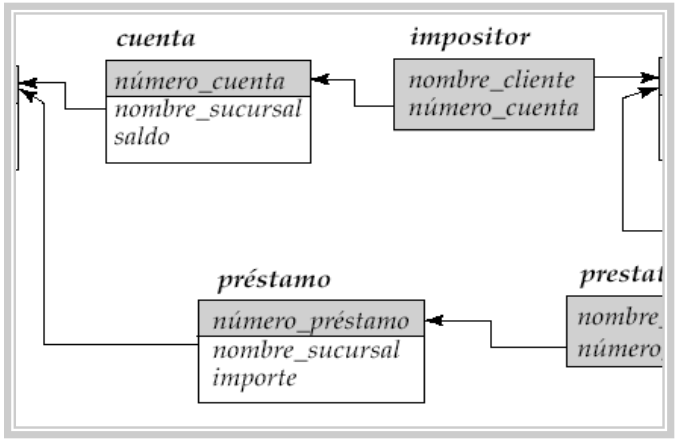
CLIENTE (idCliente, NombreC, Direccion, teléfono,fecNac,email)
ARTICULO(idArticulo, nomArticulo, Precio, unidades)
COMPRA(IdCliente, IdArticulo, FecCompra, NumUnidades)

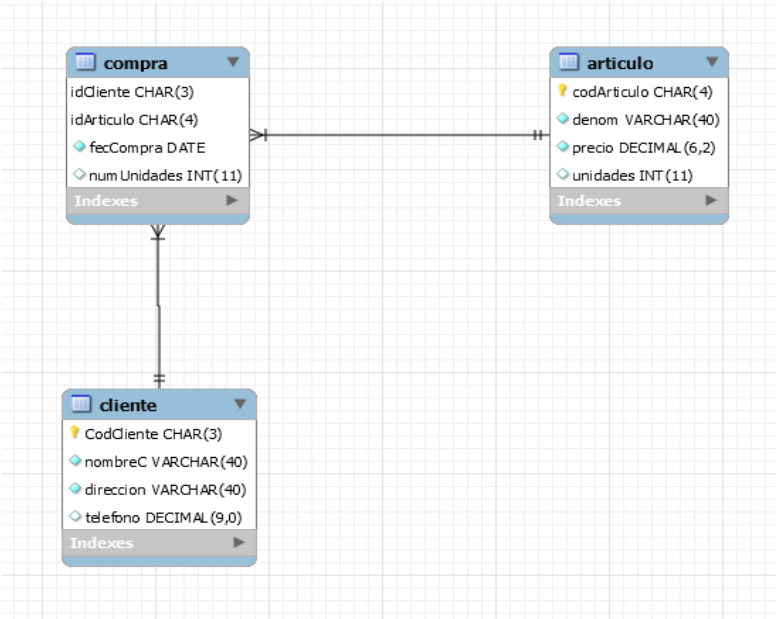
Cliente

idCliente	nombreC	direccion	telefono	fecNac	email
008	Torcuato Montero	Rio Duero 14	937846308	1980-05-14	torcuato.montero@example.com
009	Asuncion Rodríguez	Pez 14	914565308	1975-08-22	asuncion.rodriguez@example.com
011	Angela Callejo	Pedro Villar 330	914849303	1988-11-30	angela.callejo@example.com
012	Maribel Riocal	Luna 11	914394943	2000-03-15	maribel.riocal@example.com
013	Juan Antonio Sanz	Clavel 21	915656501	1982-07-19	juan.sanz@example.com

Diagramas de esquema

El esquema de relación junto con las dependencias de la clave primaria y externa, se muestra en los **diagramas de esquema**





Representación de un modelo relacional

- Un esquema relacional se representará mediante un **grafo dirigido**
- Los nodos son las relaciones de la BBDD
- Los arcos representan restricciones de la clave ajena
- El nombre de las tablas es representado en mayúsculas, los atributos en minúscula
- Primero aparece el nombre de la relación y luego entre paréntesis los atributos
- Las claves **primarias** aparecen **subrayadas**
- Las claves **alternativas** aparecen en **negrita**
- Las claves **ajenas o externas** están representadas en letra **cursiva** y referencian a la relación en la que son clave primaria mediante una flecha
- Los atributos que pueden tomar valores **nulos** aparecen con un **asterisco**

Ejemplo, clave ajena

Editorial

NombreE	**Dirección**	**País**	**Ciudad**
Universal Books	Brown Sq.23	EEUU	Los Angeles
Rama	Canillas, 144	España	Madrid
McGraw-Hill	Basauri, 117	España	Madrid
Paraninfo	Virtudes, 7	España	Madrid

NomEditorial es una clave ajena a la relación libro, referencia a nombreE que es clave primaria en la relación Editorial.
El atributo NomEditorial **no** va a poder tomar valores que no estén en NombreE

Libro

ISBN	**Título**	**...**	**NomEditorial**
0034523452	Int. Artificial		Paraninfo
1224344441	Concep. y Dis		Rama
2298108485	Turbo C++		McGraw-Hill
9876543255	Virus Informáticos		
0987654678	Sistemas Operativos		Rama

Ejemplo, grafo del modelo relacional

PELICULA(CodPelicula,....)

CodPelicula, clave primaria de pelicula

CodPelicula, clave ajena en interviene
CodActor clave ajena en interviene
CodPelicula y CodActor forman la clave primaria de interviene

INTERVIENE(*CodPelicula, CodActor*)

ACTOR(CodActor,....)

CodActor, clave primaria de actor

Líneas del grafo que unen la clave ajena con la clave primaria con la que se relaciona

Restricciones e integridad

- Existen muchas restricciones en los valores de un estado de la base de datos.
- Estas restricciones se derivan del minimundo que representa la base de datos.
- **Cuando se cumplen las restricciones se dice que el estado de la base de datos es íntegro.**
- Por ello, son sinónimos “restricción”/”integridad”/“regla de integridad”
- Hay ciertas restricciones o reglas de integridad que pueden ser mantenidas automáticamente por el SGBD. Se expresan con el LDD.
- Mantener la integridad de manera automática por el SGBD es más seguro que dejar que cada programa haga cumplir las restricciones por su cuenta.

Restricciones

1.- Restricción de clave primaria, (PRIMARY KEY)

Permite declarar un atributo o conjunto de atributos como clave primaria de una relación

2.- Restricción de unicidad (UNIQUE)

Nos permite definir claves alternativas. Los valores de uno o mas atributos no pueden repetirse en diferentes tuplas de una relación

3.- Restricción de obligatoriedad (NOT NULL)

Permite declarar si uno o más atributos de una relación deben tomar siempre un valor

4.- Restricción de clave ajena (FOREIGN KEY)

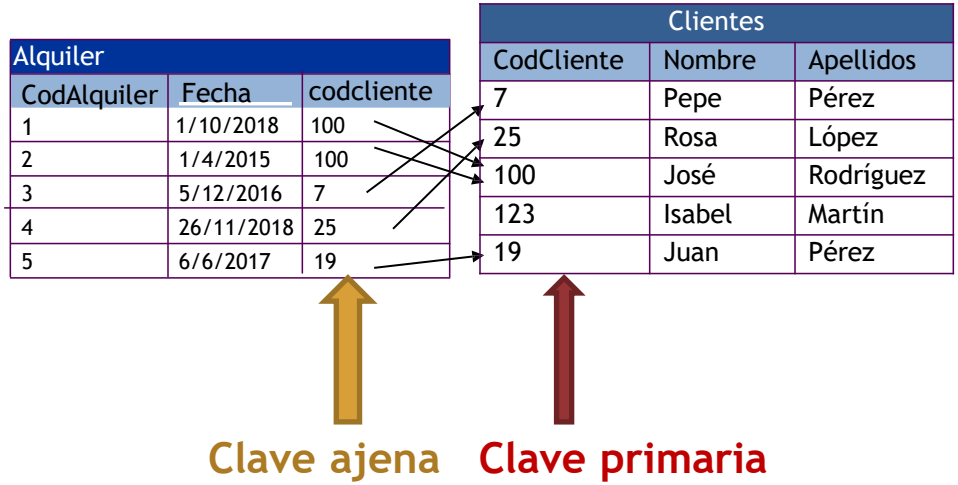
También denominada **integridad referencial,** se utiliza para enlazar relaciones de una base de datos mediante claves ajenas.

Sirve para mantener la consistencia entre las tuplas

Los atributos que son clave ajena en una relación no necesitan tener los mismos nombres que los atributos de la clave primaria con la cual ellos se corresponden.

La integridad nos permite enlazar relaciones entre sí dando lugar a la estructura de la BD, el modelo relacional permite definir ***las opciones de borrado y modificación en las claves ajenas.***

Claves ajenas - Foreign key



Cuando rellenamos codCliente de la tabla alquiler y tenemos la restricción de clave ajena, el código de cliente tiene que existir en cliente. De esta forma nos aseguramos que los clientes a los que alquilo algún inmueble los tengo en mi base de datos.

Restricciones

Las posibilidades para una **operación de actualización**:

Borrado/modificación en cascada (CASCADE)

El B/M de una tupla en la relación padre ocasiona un B/M de todas las tuplas relacionadas en la relación hija.

Borrado/modificación restringido (RESTRICT)

En este caso si existen tuplas en la relación hija relacionadas con la tupla de la relación padre sobre la que se realiza la operación, entonces no se permitir llevar a cabo esa operación.

Borrado/modificación con puesta a nulos (SET NULL)

Nos permite poner el valor de la clave ajena referenciada a NULL, cuando se produce el B/M de una tupla en la relación padre.

Borrado/modificación con puesta a un valor por defecto (SET DEFAULT)

Es similar al anterior, con la excepción de que el valor al que se ponen las claves ajenas referenciadas es un valor por defecto que se habrá especificado en la definición de la tabla correspondiente.

Las opciones de borrado y modificación pueden ser distintas para una determinada clave ajena de una relación; p.e., es posible definir el borrado en cascada y la modificación restringida.

Ejemplo

Actualización en cascada
Edificio(NombreEdif,…
Despacho(numD, *NomEdif,….*

Si borramos un edificio se borrarán todos los despachos del edificio
Si actualizamos a un valor, por ejemplo cambiamos el nombre, se cambiará también el nombre del edificio en la relación despachos

Restricciones

5.- Restricciones de Verificación (CHECK)

Puede ser necesario especificar una condición que deben cumplir los valores de determinados atributos de una relación de la BD aparte de las restricciones ya vistas de clave primaria, unicidad, obligatoriedad y clave ajena.

Para ello se definen las restricciones de verificación, que siempre llevan implícitas un rechazo en caso de que no se cumpla la condición especificada y que también se comprueban ante una inserción, borrado o modificación.

6.- Aserciones (ASSERTION)

Generaliza al anterior, lo forman las aserciones en las que la condición se establece sobre elementos de distintas relaciones.

Tienen un nombre que lo especifique. Funcionan igual que las restricciones de verificación.

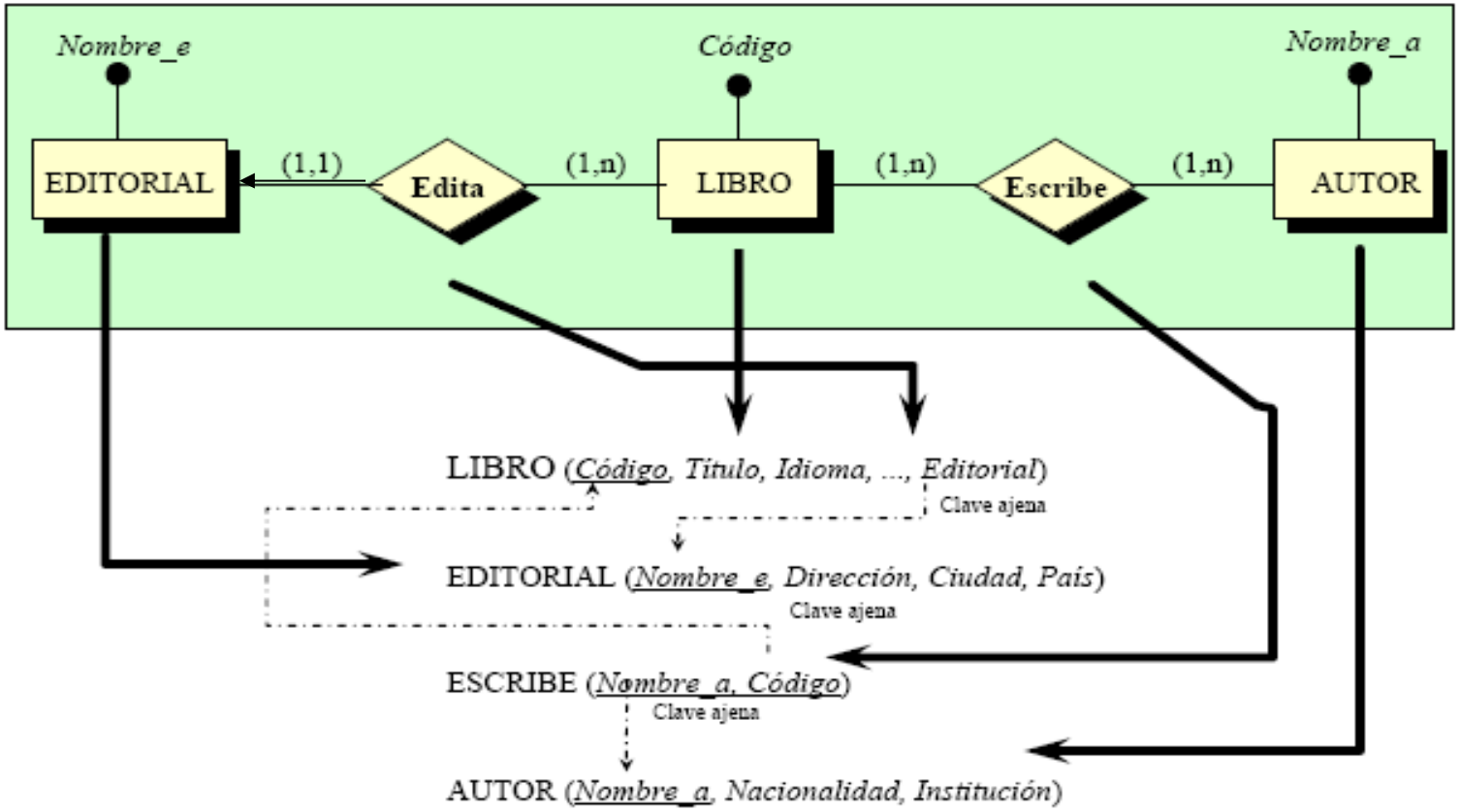
La diferencia consiste en que las restricciones de verificación tienen como ámbito una única relación.

7.- Disparadores (TRIGGER)

Permiten además de indicar una condición, especificar la acción que queremos llevar a cabo si la condición se hace verdadera.

Se pueden interpretar como reglas de tipo evento-condición-acción (ECA) que cuando se produce un evento, si se cumple una condición entonces se realiza una determinada acción.

Ejemplo, modelo entidad-relación y grafo relacional



Problemas que pueden surgir en el modelo relacional

- Incapacidad para almacenar ciertos hechos
- Redundancias y por tanto, posibilidad de incoherencias
- Ambigüedades
- Pérdida de información (aparición de tuplas espúreas)
- Pérdida de dependencias funcionales, es decir, ciertas restricciones de integridad que dan lugar a interdependencias entre los datos.
- Aparición en la BD de estados no válidos, es decir, anomalías de inserción, borrado y modificación.

En conclusión, el esquema relacional obtenido debe ser analizado para comprobar que no presenta los problemas anteriores.

Teoría de la Normalización

Formas normales

- La normalización es el proceso de organizar los datos de una base de datos. Evitar complejidades, eliminar duplicados, y organizar los datos de una manera consistente.
- Se incluye la creación de tablas y el establecimiento de relaciones entre ellas según reglas (formas normales) diseñadas tanto para proteger los datos como para hacer que la base de datos sea más flexible al eliminar la redundancia y las dependencias incoherentes, garantizar la integridad.

Primera forma normal (1FN)

No hay grupos repetidos de columnas, ni una columna guarda múltiples valores.

Por ejemplo, si de una persona queremos guardar varios teléfonos deberíamos crear una tabla de teléfonos y relacionarla con la tabla de usuarios.

Persona (idPersona, nombre, teléfono1, teléfono2, telefono3) ✗

Persona (idPersona, nombre) Teléfono (IdPersona, numTel)

Formas normales

Segunda forma normal(2FN)

Cada columna de una tabla está relacionada con todas las columnas de la clave primaria y no solo por una combinación de parte de la clave primaria. Todos los atributos no claves son totalmente dependientes de la clave primaria.

Si guardamos las personas que trabajan en una empresa
Persona (idPersona, idEmpresa, direccion_empresa, puesto, horas semanales)

cumple la 1FN al no tener columnas repetidas ni múltiples valores en una columna
no cumple la 2FN estando la clave primaria formada por los campos *idPersona* e *idEmpresa* y el campo *direccion_empresa* siendo solo dependiente del campo *idEmpresa*
Además, puede contener posibles inconsistencias en los valores de las direcciones, si se quiere actualizar la dirección de una empresa habría que actualizar todos los registros de los empleados y empresa

Persona (idPersona, IdEmpresa, puesto, horas semanales)
Empresa (idEmpresa, direccion_empresa)

Este modelo sí cumple la 2FN

Formas normales

Tercera forma normal(3FN)

Cada columna de una tabla está relacionada directamente con las columnas de la clave primaria, no de forma transitiva a través de otro campo.

Continuando con el caso anterior, cumple la 2FN no cumple la tercera **Si el campo *horasSemanales* depende del puesto, mismo puesto, mismas horas semana**. Podríamos tener inconsistencia en los datos si dos personas tuviesen diferentes horas semanales para el mismo puesto.

Persona (idPersona, idEmpresa, puesto, horas semanales) ✗

Persona (idPersona, idEmpresa, puestoP)
Puesto (puestoP, horas semanales)

Problemas

"hechos distintos, deben almacenarse en objetos distintos"

ESCRIBE

AUTOR	NACIONALIDAD	COD_LIBRO	TITULO	EDITORIAL	AÑO
Pérez, C	Española	98987	Database	Ra-Ma	2003
Pérez, C.	Española	97777	SQL Stan	Ra-Ma	2004
Pérez, C.	Española	98987	Guía para	Paraninfo	2000
Codd,E.	Norteamericana	7890	Relational	Addison,W.	1990
Gardarin	Francesa	12345	Basi Dati	Paraninfo	1986
Gardarin	Francesa	67890	Comp BD	Eyrolles	1984
Valduriez	Francesa	67890	Comp BD	Eyrolles	1984
Kim,W.	Norteamericana	11223	BD OO	ACM	1989
Lochovsky	Canadiense	11223	BD OO	ACM	1989

- *Redundancia*, ya que la nacionalidad del autor se repite por cada ocurrencia del mismo. Lo mismo sucede cuando un libro tiene más de un autor, se repite la editorial y el año de publicación.
- *Anomalías de modificación*, es fácil cambiar el nombre de una editorial en una tupla sin modificar el resto de las que corresponden al mismo libro, lo que da lugar a incoherencias.
- *Anomalías de inserción*, si queremos introducir información de algún autor, del que no hubiera ningún libro en la base datos, no sería posible, ya que cod_libro es parte de la clave primaria de la relación (regla de integridad de la entidad). La inserción de un libro, que tiene dos autores obliga a insertar dos tuplas en la relación.
- *Anomalías de borrado*, si queremos eliminar un cierto libro, deberíamos perder los datos de su autor y viceversa.

Solución

LIBRO(cod_libro, titulo, editorial, año **)**

AUTOR(nombre, nacionalidad **)**

ESCRIBE(cod_libro, nombre **)**

Un buen diseño debe poseer:

- Reflejar la estructura del problema en el mundo real.
- Ser capaz de representar todos los datos esperados, incluso con el paso del tiempo.
- Evitar el almacenamiento de información redundante.
- Proporcionar un acceso eficaz a los datos.
- Mantener la integridad de los datos a lo largo del tiempo.
- Ser claro, coherente y de fácil comprensión.

aunque algunas de ellas pueden llegar a ser contradictorias entre sí, en este caso el diseñador deberá elegir

Bibliografía

- Fundamentos de Bases de Datos, 6ª edición, Abraham Silberschatz, Henry E. Korth y S. Sudarshan, McGraw-Hill, 2014
- Fundamental of Database Systems, 7ª edición, Ramez Elmasri y Shamkant B. Navathe, editorial Addison-Wesley, 2015
- Database Systems. The Complete Book, 2ª edición, Hector García-Molina, Jeffrey D.Ullman y Jennifer Widom, editorial Prentice-Hall, 2009
- A First Course in Database Systems, 3ª edición, Jeffrey D. Ullman y Jennifer Widom, editorial Prentice-Hall, 2007
- Tecnología y diseño de bases de datos, Mario G. Piattini Velthuis y otros, editorial Ra-Ma, 2006

ntic master
revolucionamos la comunicación

UNIVERSIDAD COMPLUTENSE MADRID