Las flechas que conectan un grafo relacional nacen en una clave externa y terminan en la tabla primaria a la que referencian.

PK = UK + NN

PK (Clave primaria).

UK (Restricción de unicidad).

NN (Restricción de obligatoriedad).

FK (Clave externa).

POLÍTICAS DE BORRADO:

- Cada FK tiene su propia política de borrado.

- Las políticas de borrado deben tener coherencia en el caso de tener varias tablas diferentes relacionadas.

- Cada FK está ahí PORQUE TIENE UN SIGNIFICADO.

**A(id,...) <-----(Cascade)------- B(id,...,id\_a) <-----(Cascade)------- C(id,..., id\_b)**

**1-** NO ACTION

**2-** CASCADE  **(A—B—C)**

**3-** SET NULL **(\_)**

**4-** SET DEFAULT **(1)**

**5-** SOFT DELETE **(T/F)**

**UD2 BASES DE DATOS: INTRODUCCIÓN AL MODELO RELACIONAL**

**1- Modelo de datos**

En informática, un **modelo de datos** es un lenguaje utilizado para la descripción de una base de datos.

Según el nivel de abstracción, podemos clasificar diferentes modelos:

* **Modelos de datos conceptuales:** son aquellos que describen las estructuras de datos y restricciones de integridad. Se utilizan durante la etapa del análisis de un problema dado, y están orientados a representar los elementos que intervienen y sus asociaciones. Por ejemplo, el modelo entidad-relación.
* **Modelos de datos lógicos:** se centran en las operaciones y se implementan por un tipo de sistema gestor de base de datos. Por ejemplo, el modelo relacional.
* **Modelos de datos físicos:** son estructuras de datos de bajo nivel, implementadas por un sistema gestor de base de datos concreto.

Es decir, un mismo modelo conceptual se podría representar mediante un modelo relacional o mediante un modelo NoSQL.

Por otro lado, un mismo modelo lógico se representará con diferentes modelos físicos si lo implementamos con MySQL o con Postgresql.

**2- El modelo relacional**

**2.1- Introducción**

Se utilizó el **modelo jerárquico** como la primera forma de describir de forma más humana una base de datos. Después reinó el **modelo en red** especialmente en su norma Codasyl.

Edgar Frank Codd fue el creador del **modelo relacional** de bases de datos.

**2.2- Objetivos**

* **Independencia física.** La forma de almacenar los datos, debe ser absolutamente independiente del modelo conceptual de los mismos. Si la forma de almacenar los datos cambia (si cambia el esquema físico) , no es necesario cambiar los esquemas lógicos, funcionan perfectamente. Esto permite que los usuarios y usuarias se concentren en qué resultados desean obtener de la base de datos independientemente de cómo estén realmente almacenados los datos.
* **Independencia lógica.** Se refiere a que la lógica de la base de datos debe de ser independiente de la forma externa de acceso a la base de datos (los esquemas externos). Las aplicaciones que utilizan la base de datos no deben ser modificadas porque se modifique el esquema lógico de la misma. De una manera más precisa: gracias a esta independencia el esquema externo de la base de datos es realmente independiente del modelo lógico. En la práctica, esta independencia es difícil de conseguir.
* **Flexibilidad.** La base de datos ofrece fácilmente distintas vistas en función de los usuarios y aplicaciones. La visión de los datos se adapta al usuario que los requiere.
* **Uniformidad.** Las estructuras lógicas siempre tienen una única forma lógica (las tablas). Es decir, manejar el modelo relacional es manejar las tablas. Sencillez. Facilidad de manejo (algo cuestionable, pero ciertamente verdadero si comparamos con los sistemas gestores de bases de datos anteriores a este modelo).

**3- Estructuras de una base de datos relacional**

**3.1- Relación o tabla**

Según el modelo relacional (desde que Codd lo enunció), el elemento fundamental del modelo es lo que se conoce como **relación**, aunque más habitualmente se le llama tabla (o también array o matriz).

Las relaciones constan de:

* **Atributos.** Es cada una de las propiedades de los datos de la relación (nombre, dni,...). Las relaciones representan conjuntos de objetos o elementos reales, cada atributo es propiedad o característica de dicho elemento.
* **Tuplas.** Referido a cada ejemplar, objeto o elemento de la relación. Por ejemplo, si una relación almacena personas, una tupla representaría a una persona en concreto.

La forma de representar relaciones es mediante tablas regulares en las que las columnas se corresponden con los atributos y las filas con las tuplas. Esta forma es más visual y entendible.

**3.2- Tupla**

Se le llama tupla a cada uno de los elementos de una relación. **UNA TUPLA ES UNA FILA.**

Las tuplas, en el modelo relacional, cumplen estas premisas:

* Cada tupla se debe corresponder con un elemento del mundo real.
* No puede haber dos tuplas iguales (con todos los valores iguales).

**3.3- Dominio**

Un dominio contiene todos los posibles valores que puede tomar un determinado atributo. Dos atributos distintos pueden tener el mismo dominio.

Un dominio está formado por un conjunto finito de valores del mismo tipo. A los dominios se les asigna un nombre y así podemos referirnos a ese nombre en más de un atributo, facilitando la aplicación de los mismos.

**Es decir, un dominio expresa de forma inequívoca los valores posibles dentro de un atributo.**

Un ejemplo de dominio: el que hace que un valor para el atributo dni sólo se considere perteneciente a su dominio si tiene ocho números, una letra y además la letra cumple una regla matemática concreta sobre los números.

Una característica fundamental de los dominios es que sean **atómicos**, es decir, que los valores contenidos en los atributos no se pueden separar en valores de dominios más simples.

Un dominio debe tener: **Nombre**, **Definición lógica**, **Tipo de datos** y **Formato**.

Por ejemplo, si consideramos el Sueldo de un empleado, tendremos:

* **Nombre:** Sueldo.
* **Definición lógica:** Sueldo neto del empleado
* **Tipo de datos:** número entero positivo.
* **Formato:** 9.999€.

La forma de indicar un dominio se puede hacer utilizando dos posibles técnicas:

* **Intensión.** Se define el dominio indicando la definición exacta de sus posibles valores. Por intensión se puede definir el dominio de edades de los trabajadores como: números enteros entre el 16 y el 65 (un trabajador sólo podría tener una edad entre 16 y 65 años).
* **Extensión.** Se indican algunos valores y se sobreentiende el resto gracias a que se autodefinen con los anteriores. Por ejemplo el dominio localidad se podría definir por extensión así: Sevilla, Carmona, San Juan de Aznalfarache,...

Además pueden ser:

* **Generales.** Los valores están comprendidos entre un máximo y un mínimo y pueden tomar cualquier valor intermedio.
* **Restringidos.** Solo pueden tomar un conjunto de valores.

**3.4- Grado y cardinalidad**

**Grado:** tamaño de una tabla en base a su número de atributos (columnas). Mientras mayor sea el grado, mayor será su complejidad para trabajar con ella.

**Cardinalidad:** número de tuplas o filas de una relación o tabla.

**3.5- Sinónimos**

Los términos vistos anteriormente tienen distintos sinónimos según la nomenclatura utilizada. A ese respecto se utilizan tres nomenclaturas:

**Nomenclatura relacional Nomenclatura visual Nomenclatura ficheros**

relación tabla fichero

tupla fila registro

atributo columna campo

grado nº columnas nº campos

cardinalidad nº filas nº registros

**3.6- Propiedades de las relaciones del modelo relacional**

* Cada tabla debe tener un nombre distinto.
* Cada atributo de la tabla solo puede tener un valor en cada tupla.
* Cada atributo tiene un nombre distinto en cada tabla (aunque puede coincidir en tablas distintas).
* Cada tupla es única (no hay tuplas duplicadas).
* El orden de los atributos no importa.
* El orden de las tuplas no importa.

**3.7- Tipos de tablas**

PUM

**4- Tipos de datos**

Hemos visto que esos atributos se mueven dentro de un dominio, que formalmente es un conjunto de valores. Pues bien, en términos de sistemas de base de datos, se habla más de tipos de datos que de dominios. Al crear la relación (tabla) decidimos qué conjunto de datos deberá ser almacenado en las filas de los atributos que hemos considerado. Tenemos que asignar un tipo de dato a cada atributo.

Con la asignación de tipos de datos, también habremos seleccionado un dominio para un atributo.

Cada campo:

* Debe poseer un Nombre (relacionado con los datos que va a contener)
* Debe tener asociado un tipo de dato.

Tipos de datos más comunes que se encuentran generalmente:

* **Texto:** almacena cadenas de caracteres (números con los que no vamos a realizar operaciones matemáticas, letras o símbolos).
* **Numérico:** almacena números con los que vamos a realizar operaciones matemáticas.
* **Fecha/hora:** almacena fechas y horas.
* **Sí/No (booleanos):** almacena datos que solo tienen dos posibilidades (verdadero/falso).
* **Autonumérico:** valor numérico secuencial que el SGBD incrementa de modo automático al añadir un registro (fila).
* **Binario:** valor que permite guardar dentro ficheros o cadenas de bytes.
* **Fechas:** valores que permiten guardar fechas, fechas y horas, intervalos de tiempo, momentos, duraciones, ...

**5- Claves**

**5.1- Superclaves**

Una **superclave** es el atributo o conjunto de atributos de una relación (o tabla) que nos permite diferenciar una tupla (fila) de otra.

El único modo de identificar las superclaves es conociendo el significado real de los atributos (campos), ya que así podremos saber si es posible que aparezcan duplicados.

Por ejemplo, podríamos pensar que Nombre y Apellidos podrían ser una clave candidata, pero ya sabemos que cabe la posibilidad de que dos personas puedan tener el mismo Nombre y Apellidos, así que lo descartamos.

Una superclave debe cumplir el siguiente requisito:

* **Unicidad:** no puede haber dos tuplas (filas) con los mismos valores para esos atributos.

**5.2- Claves candidatas**

Las **claves candidatas** serán aquellas superclaves que tengan un menor número de atributos (es decir, un menor grado).

Las claves candidatas pueden estar formadas por más de un atributo, siempre y cuando éstos identifiquen de forma única a la fila.

Cuando una clave candidata está formada por más de un atributo, se dice que es una **clave compuesta**.

Una clave candidata debe cumplir los siguientes requisitos:

* **Unicidad:** no puede haber dos tuplas (filas) con los mismos valores para esos atributos.
* **Irreducibilidad:** si se elimina alguno de los atributos deja de ser única.
* **Menor grado:** no puede existir otra clave candidata que esté formada por un número menor de columnas.

**5.3- Clave primaria**

La **clave primaria** de una relación es aquella clave candidata que se escoge para identificar sus tuplas de modo único.

De alguna manera podemos decir que la clave primaria es la mejor clave candidata.

Criterios para escoger la mejor clave candidata como clave primaria:

**Tamaño menor:** ya hemos comprobado que una clave primaria debe tener el menor grado posible (menor no de atributos). Además, nos puede interesar que el tamaño del dato a almacenar sea lo más pequeño posible. Esto permitirá tener algunas ventajas a la hora de buscar o comparar.

**Que sea un atributo clave natural:** es decir, que sea uno de los campos que tenemos que almacenar en dicha tabla.

**Que sea frecuentemente condición de búsqueda:** es decir, que sea un campo sobre el que vamos a realizar consultas con frecuencia.

**Es mejor añadir un campo numérico que tener una clave primaria natural compuesta:** es decir, si la relación sobre la que queremos definir la clave primaria sólo tiene claves candidatas de grado 2, en ocasiones es mejor añadir un atributo artificial y autonumérico, y definir el mismo como clave primaria.

Aunque, al final, el mejor criterio para escoger la clave primaria será la experiencia del diseñador de la base de datos.

**5.3- Claves alternativas**

Todas aquellas claves candidatas que no hemos escogido como clave primaria se denominan **claves alternativas.**

**Las superclaves engloban a las claves candidatas y las claves candidatas engloban a la clave primaria.**

**5.4- Claves externas, foráneas o ajenas**

Una **clave externa** (también llamada ajena, secundaria o foránea) es un atributo o conjunto de atributos de una relación cuyos valores coinciden con los valores de la clave primaria de alguna otra relación (o de la misma).

Las claves externas representan asociaciones entre datos. Dicho de otra manera, es la existencia de la clave primaria de una tabla en otra tabla.

Las claves externas tienen por objetivo establecer una conexión con la clave primaria que referencian. Por lo tanto, los valores de una clave externa deben estar presentes en la clave primaria correspondiente, o bien deben ser valores nulos. En caso contrario, la clave externa representaría una referencia o conexión incorrecta. Esta se llama **integridad referencial**.

**IMÁGENES Y EJEMPLOS DE CLAVES EXTERNAS**

**6- Valores nulos**

¿Qué sucede si al guardar los datos de los Usuarios hay algún dato que no tengo o no necesito guardarlo porque no corresponde?

Independientemente del dominio al que pertenezca un campo, éste puede tomar un valor especial denominado **NULO (NULL en inglés)** que designará la ausencia de dato.

Cuando trabajamos con claves externas el valor nulo indica que la tupla o fila no está relacionada con ninguna otra tupla o fila. Este valor NULO es común a cualquier dominio.

**NULO no es igual que una cadena de caracteres vacía, un espacio en blanco o el valor numérico 0.**

**Nulo es ausencia o desconocimiento de información.**