Tema 37.- Modelo de datos relacional. Estructuras. Operaciones. Álgebra relacional.

Índice

1.- Introducción

Una **base de datos** es una colección de datos relacionados a la que puede acceder un usuario para modificar, añadir, consultar o eliminar datos. Las bases de datos basan su estructura en un **modelo de datos**, que es una colección de conceptos, normas y acuerdos que permiten representar y manipular los datos.

Estos modelos de datos pueden ser de varios tipos: siendo los **relacionales**, basados en el uso de relaciones y de concurrencia e integridad, y los **NoSQL**, cuyas mejores cualidades son la velocidad y la gran cantidad de datos que maneja, los más utilizados hoy en día**.**

El **modelo relacional** es un modelo matemático, basado en la teoría de conjuntos. Los elementos del modelo se representan por medio de relaciones, y los mecanismos para procesar la información tienen también una base matemática: el álgebra y el cálculo relacional, que se verá en detalle en este tema.

2.- Modelo de datos relacional

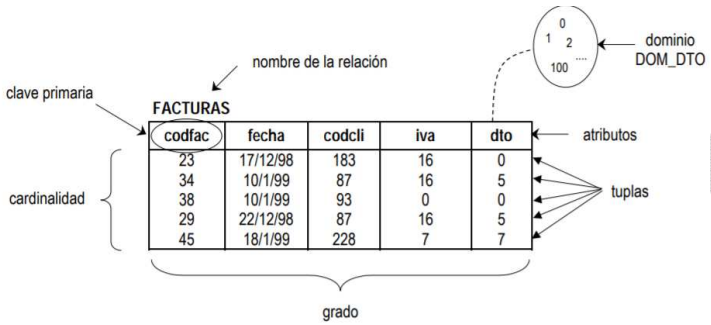
El **modelo relacional** está constituido por estructuras de datos que recogen la información del mundo real, pero sometidas a una serie de premisas o reglas de integridad. Sobre estas estructuras se podrán realizar una serie de operaciones que nos permitirán la manipulación de dichos de datos.

El objetivo general del modelo relacional es abstraerse del SGBD, de forma que pueda ser implementado en cualquiera de ellos (Oracle, MySQL…). Fue desarrollado por E. F. Codd en 1970, con los siguientes objetivos particulares:

- Independencia de datos: La forma de almacenar los datos (esquema físico), no debe afectar a cómo se ha definido su estructura (esquema lógico), y la modificación de su estructura no debe afectar a las aplicaciones que hacen uso de la base de datos.
- Vista de datos: Los usuarios tendrán una vista de los datos que necesitan y solamente de estos, para poder adaptarse a sus necesidades.
- Uniformidad: las estructuras lógicas de los datos presentan un aspecto uniforme, lo que facilita la concepción y manipulación de la base de datos por parte de los usuarios.
- Flexibilidad: Los usuarios podrán crear y modificar la base de datos con facilidad
- Sencillez: Al ser estructuras sencillas, deben ser fácilmente comprensibles.

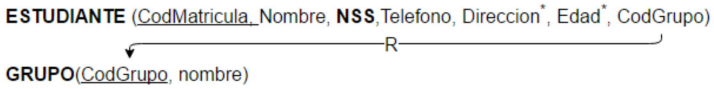
El modelo relacional es un modelo teórico con una sólida base matemática. Se implementa en los SGBD relacionales por medio de tablas (relaciones), compuestas de filas (tuplas) y columnas (atributos):

- Relación: Es un conjunto ordenado de ocurrencias con aspectos en común. R (A1, A2, ..., An). (tabla).
- Atributo: Son cada una de las propiedades de cada relación. A_i, i=1,2… (columna). Cada atributo almacena información sobre una propiedad determinada de la relación.
- Dominio: Es el conjunto de valores que pueden tomar los atributos. D_i, i =1,2… (por ejemplo, el Dominio 'sexo' de una persona, contendría los valores Varón, Mujer)
- Tupla: Contienen todas las características o atributos de un solo individuo o valor (fila). Cada tupla posee una ocurrencia de la relación representada por la tabla.
- Grado de una relación: Es el número de atributos que tiene una tabla.
- Cardinalidad: Es el número de tuplas que tiene una tabla.



FACTURAS(codfac, fecha, codcli, iva, dto)

- Esquema relacional: Es la clave del modelo relacional. Se compone de una **relación**, un conjunto de **atributos** y un conjunto de **dominios**.
    - *En tablas relacionales*: Mostrando cada relación como una tabla, los atributos en columnas y las ocurrencias en filas, las claves primarias remarcadas y las ajenas apuntando a su referenciada
    - *En grafo relacional*: Mostrando cada relación en mayúsculas, los atributos entre paréntesis, las claves primarias subrayadas, las claves ajenas referenciando la clave primaria de su relación:



3.- Estructura

El modelo de datos relacional consta de:

- Una estructura: que en el caso que nos ocupa dicha estructura se denomina *relación*.
- Una manipulación: formada por los *operadores* que manipulan la estructura.
- Una semántica: representada por las *restricciones* que definen el modelo junto con unas *reglas* que garanticen la integridad de los datos.

3.1.- Relaciones

Como hemos visto, una **relación** es un conjunto de tuplas, donde cada tupla tiene ciertos atributos y cada ocurrencia representa un valor que describe un aspecto del mundo real. En una relación no pueden repetirse tuplas ni existe un orden específico para los valores o atributos. La estructura tupla coincide con el tipo de datos registro presente en todos los lenguajes de programación y en los modelos de datos anteriores al relacional.

Para manipular la información utilizamos un **lenguaje relacional**. E.F. Codd definió dos lenguajes formales a principios de los 70: el Álgebra relacional y el Cálculo relacional. El Álgebra relacional permite describir la forma de realizar una consulta, en cambio, el Cálculo relacional solamente indica lo que se desea devolver.

Dados los atributos A_i, i=1,2..., n con dominios D_i, i=1,2...n definimos **relación** asociada a $A_1...A_n$, y notaremos por R ($A_1...A_n$) a cualquier subconjunto del producto cartesiano D_1 x D_2 x ... x Dn. Las relaciones se identifican con letras (R, S, T) y los atributos también (A, B…). Las tuplas de una instancia se anotan como x1,x2… ϵ R. Los valores de un atributo (A_i)se representan como $x_j:x_i[A_i]$. Un esquema de relación se representaría R ($A_1:D_1$, $A_2:D_2$, . . . $A_n:D_n$).

Por ejemplo: *CURSO (cod_curso:códigos, nombre:nombres, n_horas:horas)*

3.1.1.- Dependencias

Las relaciones pueden tener diferentes dependencias, en función de los requisitos de la relación. Estas dependencias pueden ser:

- Dependencia funcional: Se dice que existe una dependencia funcional en una relación R del atributo a_2 con respecto al atributo a_1, cuando para cada valor de a_1, le corresponde un único valor del atributo a_2. Su representación es: $R.a_1 \rightarrow R.a_2$. Armstrong desarrolló los tres axiomas de dependencia funcional:
  - *Dependencia reflexiva*: Si "Y" está incluido en "X" entonces $X \rightarrow Y$.
  - *Dependencia argumentativa*: $X \rightarrow Y$, entonces $XZ \rightarrow YZ$
  - *Dependencia transitiva*: $X \rightarrow Y \rightarrow Z$, entonces $X \rightarrow Z$
- Dependencias multivaluadas: Cuando la dependencia se corresponde con varios atributos R.a ($R.a_1$, $R.a_2$….$R.a_n$). Para cada valor de $R.a_1$, hay un conjunto de valores de $R.a_2$ y un conjunto de valores de $R.a_3$. Sin embargo, los valores de $R.a_2$ y $R.a_3$ son independientes entre sí.
- Dependencias jerárquicas o join: Cuando los atributos de una relación se forman a partir de las proyecciones de su relación.

3.1.2.- Restricciones

Una **restricción** es aquello no permitido, en este caso, en la estructura del modelo, pudiendo ser de dos tipos:

- Restricciones inherentes. Se refiere a aquellas restricciones que son originadas por el propio modelo:
  - No hay dos tuplas iguales.
  - El orden de las tuplas y de los atributos no es significativo.
  - Cada atributo sólo puede tomar un único valor del dominio.
  - Todas las tuplas deben tener el mismo número de atributos.
- Restricciones de usuario o semánticas. Podemos considerarlas, dentro del contexto relacional, como un predicado definido sobre un conjunto de atributos, de tuplas o de dominios, que debe ser verificado por los correspondientes objetos para que estos constituyan una ocurrencia válida del esquema. Por ejemplo, “los sueldos no pueden ser mayores que 1500 €”.

3.1.3.- Reglas de normalización

La normalización es una técnica que se ha desarrollado para obtener estructuras de datos eficientes, garantizando un buen diseño lógico de la base de datos. Las formas normales son reglas pensadas para evitar anomalías en la puesta al día e inconsistencias en los datos.

En un principio **Codd** definió las tres primeras formas normales, posteriormente, la tercera forma normal fue revisada por Boyce y el propio Codd, aumentando el número de restricciones, pasándose a llamar Forma Normal de Boyce/Codd (FNBC). En los años setenta fue Fagin quien definió la cuarta y quinta forma normal, que en muchas ocasiones no son desarrolladas para el modelo relacional.

Las bases de datos relacionales se normalizan para:

- Evitar la redundancia de los datos.
- Disminuir problemas de actualización de los datos en las tablas.
- Proteger la integridad de los datos.

Para que las tablas de nuestra base de datos estén normalizadas deben cumplir las siguientes reglas:

- Cada relación debe tener su nombre único.
- No puede haber dos tuplas iguales.
- No se permiten los duplicados.
- Todos los datos en un atributo deben ser del mismo tipo.

Para entender el proceso de normalización, utilizaremos una relación de prueba:

ALUMNOS

alumno	estudio_nivel	estudio_nombre	materia_1	materia_2
Juanito	Maestría	Medios Virtuales	MySQL	PHP
Pepito	Licenciatura	Diseño Digital	MySQL	PHP

1ª Forma normal: Consiste básicamente en no repetir campos en las tablas, es decir, atributos. Puede tener información duplicada, pero no en los atributos.

ALUMNOS

alumno_id	alumno_nombre	estudio_nivel	estudio_nombre	materia
1	Juanito	Maestría	Medios Virtuales	MySQL
1	Juanito	Maestría	Medios Virtuales	PHP
2	Pepito	Licenciatura	Diseño Digital	MySQL
2	Pepito	Licenciatura	Diseño Digital	PHP

2ª Forma normal: Debe estar como mínimo en 1FN. Cada atributo de la relación debe depender de una clave única, si tuviéramos alguna columna que se repite a lo largo de todos los registros, dichos datos deberían atomizarse en una nueva tabla.

ALUMNOS

alumno_id	alumno_nombre	estudio_nivel	estudio_nombre
1	Juanito	Maestría	Medios Virtuales
2	Pepito	Licenciatura	Diseño Digital

MATERIAS

materia_id	alumno_id	materia_nombre
1	1	MySQL
2	1	PHP
3	2	MySQL
4	2	PHP

3ª Forma normal: Debe estar en 2FN. Los campos que NO son clave NO deben tener dependencias. Se debe descomponer la relación en dos relaciones como mínimo de manera que se eliminen estas dependencias.

ALUMNOS

alumno_id	alumno_nombre	estudio_id
1	Juanito	1
2	Pepito	2

ESTUDIOS

estudio_id	estudio_nivel	estudio_nombre
1	Maestría	Medios Virtuales
2	Licenciatura	Diseño Digital

MATERIAS

materia_id	alumno_id	materia_nombre
1	1	MySQL
2	1	PHP
3	2	MySQL
4	2	PHP

Forma normal de Boyce-Codd FNBC: Se basa en el concepto de determinante funcional: uno o varios atributos de una relación de los cuales dependen funcionalmente de forma completa algún otro atributo de la misma relación. Una relación está en FNBC si cada determinante funcional es una clave candidata de la tabla. Normalmente, si el diseño es correcto, una relación en 3FN estará también en FNBC

TUTORÍAS		
DNI	**Asignatura**	**Tutor**
12121219A	Lenguaje	Eva
12121219A	Matemáticas	Andrés
3457775G	Lenguaje	Eva
5674378J	Matemáticas	Guillermo
5674378J	Lenguaje	Julia
5634823H	Matemáticas	Guillermo

Esa tabla está en tercera forma normal (no hay dependencias transitivas), pero no en forma de Boyce - Codd, ya que (DNI, Asignatura) →Tutor y Tutor→Asignatura. En este caso la redundancia ocurre por mala selección de clave. La redundancia de la asignatura es completamente evitable. La solución sería:

ASIGNATURASTUTOR	
Asignatura	**Tutor**
Lenguaje	Eva
Matemáticas	Andrés
Matemáticas	Guillermo
Lenguaje	Julia

TUTORÍAS	
DNI	**Tutor**
12121219A	Eva
12121219A	Andrés
3457775G	Eva
5674378J	Guillermo
5674378J	Julia
5634823H	Guillermo

4ª Forma normal: La 4FN aplica únicamente para relaciones N:M, y nos ayuda a eliminar la redundancia de información generada por dicho tipo de relación, creando nuevas relaciones entre las tablas con este tipo de relación.

ALUMNOS

alumno_id	alumno_nombre	estudio_id
1	Juanito	1
2	Pepito	2

ESTUDIOS

estudio_id	estudio_nivel	estudio_nombre
1	Maestría	Medios Virtuales
2	Licenciatura	Diseño Digital

MATERIAS

materia_id	materia_nombre
1	MySQL
2	PHP

MATERIAS X ALUMNO

mxa_id	alumno_id	materia_id
1	1	1
2	1	2
3	2	1
4	2	2

5ª Forma normal: Una relación se dice que está en 5NF si y sólo si está en 4NF y cada dependencia de unión (join) en ella es implicada por las claves candidatas. Normalmente, no es necesario llegar a 5FN. Una relación está normalizada si llega a 3FN. No obstante, en diseños muy grandes o con muchas dependencias, seguramente será necesario desarrollarlo hasta la 5FN.

3.2.- Claves

Cada tupla de una relación tiene que estar asociada a una **clave única** que permita identificarla. Una clave puede estar compuesta por uno o más atributos. Además, una clave tiene que ser única dentro de su relación y no se puede descartar ningún atributo de la misma para identificar una fila.

Existen dos **tipos** fundamentales de claves:

- Clave primaria (Primary Key): es el valor o conjunto de atributos que identifican una tupla dentro de una tabla. Un ejemplo claro de clave primaria seria el DNI, que es único para cada persona.
- Clave ajena (Foreign Key): es el valor o valores de una tabla que corresponde con el valor de una clave primaria en otra tabla. Esta clave es necesaria para representar relaciones entre las tablas.

Además de estos tipos, una clave puede ser:

- Superclave: Está formada por uno o más atributos que identificará de forma única e inequívoca a una entidad de otra
- Clave candidata: Son aquellos atributos pertenecientes a la relación que pueden diferenciar de forma inequívoca cada ocurrencia o tupla. La PK es la clave candidata elegida por el diseñador.
- Clave alternativa: Son las posibles claves candidatas que no han sido elegidas como PK

Las claves deben cumplir unas reglas de integridad:

- Integridad de identidad: No pueden existir valores nulos en una clave primaria. El valor nulo quiere decir que no existe información. Nunca se almacena información que no se puede identificar.
- Integridad de referencia: El valor del atributo de r_2 al que se referencia, tiene que estar contenido en el conjunto de valores de la clave primaria de r_1. Esta regla debe cumplirse estrictamente en las claves ajenas. Mantiene coherente el estado de la base de datos.

3.3.- Valores nulos

Un **valor nulo** es una marca que indica que el dato asociado a un atributo de una entidad es desconocido. Los valores nulos son útiles a la hora de manejar información que presenta valores que sí existen en el universo del discurso, pero que se desconocen en un momento dado.

El tratamiento de los nulos es complejo e implica, por ejemplo, la creación una lógica trivaluada para contemplar la existencia de valores nulos a la hora de determinar el resultado de las operaciones lógicas AND, OR y NOT.

p	q	p v q
1	1	1
1	*	*
1	0	0
*	1	1
*	*	1
*	0	*
0	1	1
0	*	1
0	0	1

3.4.- Dominios

Un **dominio** es un conjunto de valores del mismo tipo, atómicos, que puede tomar cada uno de los atributos de una relación. Toda columna ha de tener el mismo dominio que habrá sido previamente definido. Todos los dominios tendrán un nombre y un tipo. Hay tres tipos de dominios:

- Dominios por intensión: son los que están formados por un máximo y un mínimo. Por ejemplo el número de teléfono, formado por 9 números entre 0 y 9.
- Dominios por extensión: se refiere a los que forman parte de un grupo de valores determinados. Por ejemplo el sexo de una persona, que puede tomar dos valores: Hombre o Mujer.
- Dominios compuestos: son los que están formados por una combinación de dominios simples. Por ejemplo el dominio 'fecha', que tiene tres dominios: día (número entre 1 y 31) mes, (número entre 1 y 12) y año.

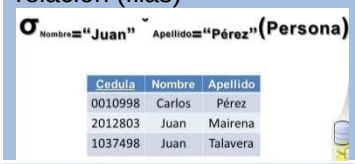
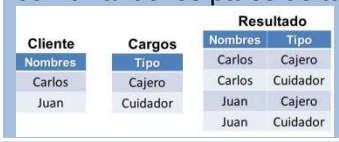
4.- Operaciones

Las operaciones utilizadas en el modelo relacional son la inserción, modificación y eliminación. Codd definió dos lenguajes para realizar estas operaciones: el Álgebra Relacional y el Cálculo Relacional, que explicaremos en los siguientes apartados.

4.1.- Álgebra relacional

El **Álgebra Relacional** consiste en un conjunto de operadores que actúan sobre relaciones. A partir de una o dos relaciones de entrada se obtiene una relación aplicando alguna de las operaciones fundamentales o adicionales. El **Álgebra Relacional** es aplicable al modelo relacional, pudiéndose aplicar una serie de operaciones:

4.1.1.- Operaciones fundamentales

Operación	Simbología	Descripción	Ejemplo
Selección o restricción	$\sigma_p(r)$	Extrae las tuplas especificadas de una relación (filas) 	σ cantidad > 1000 ^ hermanos < 6 (alumnos)
Proyección	$\Pi_{atributos}(r)$	Extrae atributos especificados de una relación (columnas) 	$\Pi_{nombre}(\sigma$ cantidad ="S"(alumnos))
Producto cartesiano	R X S	Construye una relación de otras dos combinando los pares de tuplas 	Cliente x Cargos = Resultado
Unión de conjuntos	R U S	Construye una relación combinando las tuplas que aparecen en una o ambas relaciones 	Cliente U Empleado
Diferencia de conjuntos	R - S	Construye una relación con las tuplas que aparecen en la primera y no en la segunda 	Cliente - Empleado
Renombrar*	$\rho_s(R)$	Resuelve la ambigüedad cuando hay dos atributos con el mismo nombre, renombrando de forma temporal.	ρ_c(Cliente)

*Propuesto por Korth y Silberschatz, no por Codd.

4.1.2.- Operaciones adicionales

Operación	Simbología	Descripción	Ejemplo
Intersección	R ∩ S	La relación resultante englobará a los elementos comunes de R en S	Cliente ∩ Empleado
Producto natural	R |x| S	Consiste en aplicar un producto cartesiano y una selección sobre la que se realizará una proyección	Cliente |x| Empleado
División	R ÷ S	Construye una relación con todos los valores de R que tienen enlace con S.	Cliente ÷ Empleado
Asignación	TR ← $\Pi_{atributos}$(r) TS ← $\Pi_{atributos}$(s)	Permite expresar una consulta como una secuencia de pasos de un lenguaje de programación.	TempCli← $\Pi_{idPersona}$(Cliente) TempEmp← $\Pi_{idPersona}$(Empleado) TemCli ∩ TempEmp

4.1.3.- Equivalencia con SQL

El **Álgebra Relacional** se utilizó para sentar las bases del lenguaje **SQL**, el cual es hoy en día el más utilizado en bases de datos relacionales. Existe una **correspondencia** clara entre las operaciones vistas anteriormente y su equivalente en SQL. En la siguiente tabla vemos algunos ejemplos de ello:

Álgebra	SQL
(Pacientes)	`Select *` `From Pacientes`

$\sigma_{idade>18}$ (Pacientes)	`Select *`
	`From Pacientes`
	`Where idade > 18`

$\pi_{CPF, nome}$ (Pacientes)	`Select CPF, nome` `From Pacientes`

$\pi_{CPF, nome}$ ($\sigma_{idade>18}$ (Pacientes))	`Select CPF, nome`
	`From Pacientes`
	`Where idade > 18`

4.2.- Cálculo relacional

El **cálculo relacional** fue propuesto por Codd en 1971 como alternativa al Álgebra Relacional. La diferencia fundamental entre un lenguaje algebraico y un lenguaje al que podríamos llamar predicativo es que en el primero hay que especificar qué operadores se tienen que aplicar a las relaciones para obtener el resultado, mientras que en los segundos sólo es preciso indicar cuál es el resultado que se quiere obtener.

4.2.1.- Cálculo relacional orientado a tuplas

En este lenguaje, expresamos variables que representan tuplas. Si por ejemplo existe una tupla t contenida en una relación r escribiremos t ∈ r. Si queremos expresar el valor que toma el atributo A para la tupla t, lo haremos de la

siguiente forma: t[A]. Si queremos obtener una relación con el conjunto de tuplas que cumplen el predicado P, tendremos que expresarlo de esta manera: {t / P(t)}.

Ejemplo: "Lista completa de clientes que tienen un préstamo cuyo importe sea más de un millón" (selección).

$$\{t / t \exists \ préstamo \wedge t[importe] > 1000000\}$$

4.2.2.- Cálculo relacional orientado a dominios

Los datos se guardan en variables correspondientes a atributos. Las variables se refieren a atributos, lo que en cálculo relacional orientado a tuplas era t[A1] es ahora A1, por tanto si queremos hablar de una tupla tenemos que nombrar todas las variables correspondientes a los atributos de esa tupla. (< x1, x2, ..., xn >). Por ejemplo, una tupla de préstamo se expresaría de la siguiente forma: < c, s, p, i > (cliente, sucursal, préstamo, importe)

Ejemplo: “Nombres de los clientes, número de sucursal, número de préstamo e importe de préstamos con importe superior a un millón de euros”:

$$\{ < c, s, p, i > / < c, s, p, i > \exists \ p \wedge i > 1000000 \}$$

El ejemplo más relevante de uso de este tipo de cálculo es el lenguaje **QBE (Query By Example)** creado por IBM en los 70 y que sigue utilizándose en la actualidad.

5.- Conclusión

Muchos de los SGBD actuales, como MySQL, Oracle, SQL Server, DB2, o MariaDB utilizan el **modelo relacional** como base de funcionamiento mediante el lenguaje **SQL.**

La **gran información** que se maneja hoy en día hace del modelo relacional una herramienta clave para tener un conjunto de datos con una estructura lógica rígida, gracias a su independencia e integridad de datos. El límite a la hora de utilizar bases de datos está en la capacidad de almacenamiento físico que tengamos, siendo en la actualidad prácticamente ilimitada con el almacenamiento distribuido.

No obstante, cuando pasamos una frontera del almacenamiento mayor, hablando de petabytes (**Big Data**), es cuando el modelo relacional deja de ser esencial como modelo único para el registro y explotación de información, siendo fundamental combinado con otras tecnologías para el análisis de esta información.

Esta es la tendencia hoy en día, donde los datos se han convertido en algo clave para el rendimiento, la innovación o el marketing. La explotación de las redes sociales, el posicionamiento de la marca, la minería de datos o la venta online se basan en la unión de bases de datos relacionales (que proporcionan consistencia) y bases de datos NoSQL (Not Only SQL) como MongoDB (que proporcionan escalabilidad y velocidad).

6.- Bibliografía

- Elmasri, R.; Navathe, S.B. (2000). Sistemas de bases de datos. Conceptos fundamentales. Addison-Wesley Iberoamericana.
- Silberschatz, A.; Henry F. Korth, H; Sudarshan, S.; (2006) Fundamentos de bases de datos. McGraw-Hill
- Hawryszkiewycz, I. T., Relational Database Design, Prentice-Hall Australia, 1990

uni10.es