TEMA 3. El modelo relacional

Para realizar esta práctica nos serviremos de Internet y buscaremos respuestas a estas preguntas donde la misma tenga que ver con todo lo explicado anteriormente.

Lenguajes de datos relacionales: Algebraicos y Predicativos

Los lenguajes relacionales (LR) operan sobre conjuntos de tuplas, es decir, no son lenguajes navegacionales (que manipulan registros, como Pascal, Basic, Cobol, XBase, ...) sino de especificación
. Se dividen en dos tipos:

–
Algebraicos
: los cambios de estado se especifican mediante operaciones, cuyos operandos son relaciones y cuyo resultado es otra relación. Genéricamente se conocen como álgebra relacional
.

–
Predicativos
: los cambios de estado se especifican mediante predicados que definen el estado objetivo sin indicar las operaciones que hay que realizar para llegar al mismo. Genéricamente se conocen como cálculo relacional
. Se dividen en dos subtipos:

• orientados a tuplas.
• orientados a dominios.

Estructura del modelo relacional

Se trata de un modelo bastante potente y a la vez bastante simple , que nos representas problemas. El elemento principal de este modelo es la relación. Por lo que podemos decir que una base de datos relacional está compuesta por un conjunto de relaciones.

Elementos del modelo relacional

Vistas

Se trata de una tabla ficticia la cual muestra atributos de otras tablas relacionadas. De esta forma obtenemos los datos que nos interesan de una o varias tablas. Es importante señalar que no se pueden realizar operaciones sobre vistas.

Claves

Cada tupla de una tabla tiene que estar asociada a una clave única que permita identificarla.
Una clave puede estar compuesta por uno o más atributos.
Una clave tiene que ser única dentro de su tabla y no se puede descartar ningún atributo de la misma para identificar una fila.
Existen dos tipos de claves:

- Clave prmaria (Primary Key): es el valor o conjunto de valores que identifican una fila dentro de una tabla. Nunca puede ser NULL. Un ejemplo claro de clave primaria seria el DNI, que es único para cada persona y no puede ser NULL.
- 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 la que representa las relaciones entre las tablas.

Dominios

El dominio dentro de la estructura del modelo relacional es el conjunto de valores que puede tomar un atributo. Existen dos tipos de dominios:

- dominios generales: son aquellos que están comprendidos entre un máximo y un mínimo.
- dominios restringidos: son los que pertenecen a un conjunto de valores específicos.

Relación

La relación se representa mediante una tabla, esta tabla representa a lo que en el modelo entidad-relación llamábamos entidad. Esta tabla contiene los atributos (columnas) y las tuplas (filas).

- Atributo: se trata de cada una de las columnas de la tabla. Vienen definidas por un nombre y pueden contener un conjunto de valores.
- Tupla: se trata de cada una de las filas de la tabla. Es importante señalar que no se pueden tener tuplas duplicadas en una tabla.

Restricciones de este modelo

- No existen tuplas repetidas (obligatoriedad de clave primaria). La relación se ha definido como un conjunto de tuplas, y en matemáticas los conjuntos por definición no incluyen elementos repetidos. Hay que decir, sin embargo, que muchos de los SGBD relacionales sí admiten duplicidad de tuplas.
- El orden de las tuplas y el de los atributos no es relevante.
- Cada atributo de cada tupla solo puede tomar un único valor sobre el dominio sobre el que está definido.
- Ningún atributo que forme parte de la clave primaria de una relación puede tomar un valor nulo (regla de integridad de entidad)

Estas restricciones son las que marcan la diferencia entre una tabla y una relación, ya que una tabla presenta las filas y las columnas en un orden, del cual carecen las relaciones. Por otro lado, una tabla podría contener filas repetidas. De todos modos, es muy común utilizar el término tabla para referirse a una relación.

Las 12 reglas de Codd

Preocupado por los productos que decían ser sistemas gestores de bases de datos relacionales (RDBMS) sin serlo, Codd publica las 12 reglas que debe cumplir todo DBMS para ser considerado relacional. Estas reglas en la práctica las cumplen pocos sistemas relaciónales. Las reglas son:

1. **Información.** Toda la información de la base de datos debe estar representada explícitamente en el esquema lógico. Es decir, todos los datos están en las tablas.
2. **Acceso garantizado.** Todo dato es accesible sabiendo el valor de su clave y el nombre de la columna o atributo que contiene el dato
3. **Tratamiento sistemático de los valores nulos.** El DBMS debe permitir el tratamiento adecuado de estos valores.
4. **Catálogo en línea basado en el modelo relacional.** Los metadatos deben de ser accesibles usando un esquema relacional.
5. **Sublenguaje de datos completo.** Al menos debe de existir un lenguaje que permita el manejo completo de la base de datos. Este lenguaje, por lo tanto, debe permitir realizar cualquier operación.
6. **Actualización de vistas.** El DBMS debe encargarse de que las vistas muestren la última información.
7. **Inserciones, modificaciones y eliminaciones de dato nivel.** Cualquier operación de modificación debe actuar sobre conjuntos de filas, nunca deben actuar registro a registro.
8. **Independencia física.** Los datos deben de ser accesibles desde la lógica de la base de datos aún cuando se modifique el almacenamiento.
9. **Independencia lógica.** Los programas no deben verse afectados por cambios en las tablas.
10. **Independencia de integridad.** Las reglas de integridad deben almacenarse en la base de datos (en el diccionario de datos), no en los programas de aplicación.
11. **Independencia de la distribución.** El sublenguaje de datos debe permitir que sus instrucciones funcionen igualmente en una base de datos distribuida que en una que no lo es.
12. **No subversión.** Si el DBMS posee un lenguaje que permite el recorrido registro a registro, éste no puede utilizarse para incumplir las reglas relacionales.

2. Restricciones Semánticas o de Usuario.

1. **Restricción de Clave Primaria** *(PRIMARY KEY)*, permite declarar un atributo o conjunto de atributos como la clave primaria de una relación.
2. **Restricción de Unicidad** *(UNIQUE)*, permite que una clave alternativa o secundaria pueda tomar valores únicos para las tuplas de una relación (como si de una clave primaria se tratara). Se entiende que la clave primaria siempre tiene esta restricción.
3. **Restricción de Obligatoriedad** *(NOT NULL)*, permite declarar si uno o varios atributos de una relación debe tomar siempre un valor.
4. **Restricción de Integridad Referencial o de Clave Foránea** *(FOREIGN KEY)*, se utiliza para que mediante claves foráneas podamos enlazar relaciones de una base de datos. La integridad referencial nos indica que si una relación tiene una clave foránea que referencia a otra relación, cada valor de la clave foránea o ajena tiene que ser igual a un valor de la clave principal de la relación a la que referencia, o bien, ser completamente nulo. Los atributos que son clave foránea

en una relación no necesitan tener los mismos nombres que los atributos de la clave primaria con la cual ellos se corresponden. El diseñador de la base de datos deberá poder especificar qué operaciones han de rechazarse y cuáles han de aceptarse, y en este caso, qué operaciones de compensación hay que realizar para mantener la integridad de la base de datos. Para ello el diseñador debe plantearse tres preguntas por cada clave foránea:

1. **¿Puede aceptar nulos esa clave foránea?** Por ejemplo, (tomando como referencia las relaciones PROVEEDORES, ARTICULOS) ¿tiene sentido la existencia de un articulo cuyo proveedor se desconoce? Evidentemente, no. En algunos casos esta respuesta podría ser distinta, por ejemplo, en una base de datos con las relaciones EMPLEADOS y DEPARTAMENTOS, podría existir un empleado no asignado de momento a un departamento.
2. **¿Qué deberá suceder si se intenta eliminar una tupla referenciada por una clave foránea?** Por ejemplo, si se intenta eliminar un proveedor del cual existe algún artículo. En general, para estos casos existen por lo menos tres posibilidades:
    - **Restringir:** La operación de eliminación se restringe sólo al caso en el que no existe alguna tupla con clave foránea que la referencie, rechazándose en caso contrario. En nuestro ejemplo, un proveedor podrá ser borrado, si y sólo si, por ahora, no suministra artículos.
    - **Propagar en cascada:** La operación de borrado se propaga en cascada eliminando también todas las tuplas cuya clave foránea la referencien. En nuestro ejemplo, se eliminaría el proveedor y todas las tuplas de artículos suministrados por él.
    - **Anular:** Se asignan nulos en la clave foránea de todas las tuplas que la referencien y se elimina la tupla referenciada. En nuestro ejemplo, no tiene mucho sentido, pero consistiría en asignar nulos al NIF-PROV de todas las tuplas de articulos pertenecientes al proveedor que queremos borrar, y posteriormente borrar al proveedor.
3. **¿Qué deberá suceder si hay un intento de modificar la clave primaria de una tupla referenciada por una clave foránea?** Por ejemplo, si se intenta modificar la clave de un proveedor del cual existe algún artículo. Se actua con las mismas tres posibilidades que en el caso anterior:
    - Restringir
    - Propagar en cascada.
    - Anular

5. **Restricción de Valor por Defecto** *(DEFAULT)*, permite que cuando se inserte una tupla o registro en una tabla, para aquellos atributos para los cuales no se indique un valor exacto se les asigne un valor por defecto.
6. **Restricción de Verificación o Chequeo** *(CHECK)*, en algunos casos puede ocurrir que sea necesario especificar una condición que deben cumplir los valores de determinados atributos de una relación de la BD, aparte de las restricciones vistas anteriormente.
7. **Aserciones** *(ASSERTION)*: Esta restricción generaliza a la anterior, lo forman las aserciones en las que la condición se establece sobre elementos de distintas relaciones (por ello debe tener un nombre que la identifique).

8. **Disparadores** *(TRIGGERS)*, a veces puede interesar espeficar una acción distinta del rechazo cuando no se cumple una determinada restricción semántica. En este caso, se recurre al uso de disparadores o triggers que nos permiten además de indicar una condición, especificar la acción que queremos que se lleve a cabo si la condición se hace verdadera. Los disparadores pueden interpretarse como reglas del tipo evento-condición-acción (ECA) que pueden interpretarse como reglas que especifican que cuando se produce un evento, si se cumple una condición, entonces se realiza una determinada acción

Similitudes entre el modelo relacional y la arquitectura ANSI

El MR se adapta a la arquitectura ANSI, pero con las siguientes excepciones importantes:

Al usuario se le permite ver, si tiene las correspondientes autorizaciones, tanto las relaciones base como las vistas, mientras que en la arquitectura ANSI, para un usuario, la BD está limitada al esquema externo

Aunque las vistas se corresponden con los esquemas externos de ANSI y éstos pueden actualizarse, en el MR no todas las vistas son actualizables.

Además, en la práctica muchos SGBD relacionales no responden a la arquitectura a tres niveles, ya que las definiciones del esquema conceptual y del esquema interno no están claramente diferenciadas.

Operadores principales del álgebra relacional

Básicas

Cada operador del álgebra acepta una o dos relaciones y retorna una relación como resultado. σ y Π son operadores unarios, el resto de los operadores son binarios. Las operaciones básicas del álgebra relacional son:

- **Selección - restricción (σ)**

Permite seleccionar un subconjunto de tuplas de una relación (**R**), todas aquellas que cumplan la(s) condición(es) **P**, esto es:

Ejemplo:

Selecciona todas las tuplas que contengan Gómez como apellido en la relación Alumnos.

Una condición puede ser una combinación booleana, donde se pueden usar operadores

como: , , combinándolos con operadores .

- **Proyección (Π)**

Permite extraer columnas (atributos) de una relación, dando como resultado un *subconjunto vertical* de atributos de la relación, esto es:

donde son atributos de la relación **R** .

Ejemplo:

Selecciona los atributos Apellido, Semestre y NumeroControl de la relación Alumnos, mostrados como un subconjunto de la relación Alumnos

- **Producto cartesiano (x)**

El producto cartesiano de *dos relaciones* se escribe como:

y entrega una relación, cuyo *esquema* corresponde a una combinación de todas las tuplas de **R** con cada una de las tuplas de **S**, y sus atributos corresponden a los de **R** seguidos por los de **S**.

Ejemplo:

Muestra una nueva relación, cuyo esquema contiene cada una de las tuplas de la relación Alumnos junto con las tuplas de la relación Maestros, mostrando primero los atributos de la relación Alumnos seguidos por las tuplas de la relación Maestros.

- **Unión (∪)**

La operación

retorna el conjunto de tuplas que están en R, o en S, o en ambas. R y S deben ser *uniones compatibles*.

- **Diferencia (-)**

La diferencia de dos relaciones, R y S denotada por:

entrega todas aquellas tuplas que están en R, pero **no** en S. R y S deben ser *uniones compatibles*.

Estas operaciones son fundamentales en el sentido en que (1) todas las demás operaciones pueden ser expresadas como una combinación de éstas y (2) ninguna de estas operaciones pueden ser omitidas sin que con ello se pierda información.

No básicas o Derivadas

Entre los operadores no básicos tenemos:

Intersección (∩)

La intersección de dos relaciones se puede especificar en función de otros operadores básicos:

La intersección, como en Teoría de conjuntos, corresponde al conjunto de todas las tuplas que están en R y en S, siendo R y S *uniones compatibles*.

Unión natural (⋈) (Natural Join)

La operación unión natural en el álgebra relacional es la que permite reconstruir las tablas originales previas al proceso de normalización. Consiste en combinar las proyección, selección y producto cartesiano en una sola operación, donde la condición es la igualdad Clave Primaria = Clave Externa (o Foránea), y la proyección elimina la columna duplicada (clave externa).

División (/)

Supongamos que tenemos dos relaciones A(x, y) y B(y) donde el dominio de y en A y B, es el mismo.

El operador división **A / B** retorna todos los distintos valores de x tales que para todo

valor y en B existe una tupla en A.

Agrupación (Ɠ)

Permite agrupar conjuntos de valores en función de un campo determinado y hacer operaciones con otros campos.

Características del lenguaje SQL

Bases de datos

El lenguaje SQL nos permite manipular las bases de datos. En primer lugar, aunque es posible que ya lo sepamos, indicaremos cómo funcionan las bases de datos.

Una base de datos consiste fundamentalmente un una o más tablas.

Cada tabla consiste en una o más columnas llamadas "campos", y una o más filas llamadas "registros". Cada columna o campo contiene un tipo de dato diferente y cada fila o registro contiene una "entidad", es decir objeto, persona ... que está registrado como un ente único.

La intersección de cada columna o "campo" con cada fila o "registro" es una "celda". La celda es la unidad mínima de la base de datos, y en cada una de ellas sólo podemos guardar un dato.

Cada base de datos (dentro de MySQL) debe tener un nombre único que la distinga de las demás. De la misma forma cada tabla, dentro de cada base de datos, debe tener también su nombre, no pudiendo haber nombres repetidos para distintas tablas dentro de una misma base.

Dentro de cada tabla, cada columna o "campo" tiene también un nombre. No puede haber dos columnas con el mismo nombre dentro de una tabla.

Dentro de cada base las tablas pueden estar relacionadas entre sí. Por ejemplo tenemos una base de datos llamada "literatura". Esta puede contener diferentes tablas, por ejemplo, "autores", "obras", "epoca", "genero". Cada una de estas tablas se relaciona con las otras para poder mostrar no sólo sus propios datos sino también algunos datos de las otras.

Esto nos lleva a otro tipo de elemento, la consulta. Una consulta es una serie de datos que extraemos de la tabla mediante el lenguaje SQL. En una consulta podemos extraer, por ejemplo una tabla entera, o datos que buscamos en una tabla que coincidan con un cierto criterio, o datos de varias tablas relacionados entre sí, etc.

El lenguaje SQL:

El lenguaje SQL sirve para manejar las bases de datos, es un lenguaje de texto plano, que podemos usar desde cualquier editor de textos. Nos permite acceder y manipular las bases de datos. Para usarlo con páginas web debemos usarlo conjuntamente con otros lenguajes. En la página anterior de este manual indicamos cómo usar SQL junto a PHP para bases en MySQL, por lo que no vamos a insistir en ello.

Con SQL podemos hacer entre otras las siguientes operaciones en el programa que gestiona las bases de datos:

- Crear nuevas bases de datos.
- Crear nuevas tablas en una base de datos.
- Crear nuevos registros en las tablas de una base de datos
- Crear tablas de consulta en una base de datos.
- Borrar tablas o bases de datos.
- Borrar registros.
- Cambiar uno o varios datos de un registro.
- Mostrar los datos de una tabla o consulta.
- Buscar y mostrar sólo algunos datos de una tabla o consulta, poniendo una serie de condiciones para la búsqueda.

Es decir, el lenguaje SQL permite gestionar una base de datos en su totalidad. Con MySQL y PHP podemos hacer todas estas operaciones desde un sitio web.

Sintaxis de SQL.

El lenguaje SQL se compone de sentencias. Cada sentencia es una instruccón que enviamos a la base de datos. En las sentencias se incluyen dos tipos de palabras: las palabras clave propias del lenguaje SQL y los datos de la base (datos individuales, nombres de bases, tablas o columnas. etc.);

El lenguaje SQL NO DISTINGUE entre mayúsculas o minúsculas en sus palabras clave. es decir podemos escribir tanto "select" como "SELECT". Sin embargo en los nombres de bases de datos, tablas, columnas y otros elementos de la base SI se distingue, por lo que éstos debemos ponerlos tal como están en la base.

La mayoría de los manuales sobre SQL escriben las palabras clave propias de SQL en mayúsculas, para distinguir entre palabras clave y nombres de elementos de la base o datos. Nosotros en este manual lo haremos también así.

Cada sentencia empieza por una palabra que es un verbo e indica lo que queremos hacer (create, drop, select, update, etc) seguido por el nombre del tipo de elemento al que afecta. Esto último puede ser una o varias palabras. Además también pueden usarse algunas preposiciones o adverbios que indiquen restricciones o el tipo de búsqueda (where = condición, from = restringe a el elemento indicado luego, etc.). Sabiendo un poco de inglés el lenguaje SQL puede resultar un tanto lógico ya que estamos indicando lo que queremos hacer. Por ejemplo la instrucción:

SELECT nombre, telefono FROM agenda

Indica que queremos seleccionar (SELECT) los campos "nombre" y "telefono" dentro de la tabla llamada "agenda".

Acabar cada sentencia con un punto y coma es opcional, sin embargo es lo más recomendable, ya que si bien en MySQL esto es opcional, podría haber problemas con otros tipos de sistemas para bases de datos.

Al igual que otros lenguajes de programación como html, css o php, en SQL también se ignoran los espacios en blanco de más, las tabulaciones y los saltos de línea, es decir escribir más de un espacio en blanco o un salto de línea con tabulación o sin ella tiene el mismo efecto que escribir sólo un espacio en blanco, con lo cual el programador puede aprovechar esto para poner el código de una manera clara y estructurada.

Tipos de sentencias

Las sentencias de SQL se agrupan en tres tipos:

- **DDL : Lenguaje de definición de Datos :** (Data Definition Languaje) Entran aquí las sentencias que definen y crean los objetos que soportan la base de datos (creación o supresión de bases, creación o supresión de tablas, relaciones entre tablas, claves en las tablas, etc.
- **DML : Lenguaje de Manipulación de Datos :** (Data Management Languaje) Entran aquí las sentencias para manejar los datos almacenados en las tablas, a nivel de campos (columnas) o registros (filas). Por ejemplo crear, cambiar o consultar registros o sus datos.
- **DCL : Lenguaje de Control de Datos :** (Data Control Languaje) Entran aquí las sentencias para controlar las funciones de administración y control de las bases.