Sistemas de Información

Práctica 5 Curso 2016/17 3º curso

Aprendiendo más SQL

Introducción

Aunque después de realizar las prácticas anteriores ya disponemos de los conocimientos necesarios para introducir esquemas de bases de datos en nuestro SGBD (utilizando SQL como un DDL) y para gestionar los datos almacenados (utilizando SQL como un DML), para poder aprovechar todas las posibilidades que nos proporciona un SGBD actual, como es MySQL, así como para poder poner en servicio nuestro Sistema de Bases de Datos atendiendo las necesidades de los usuarios finales en lo referente a cuestiones como la eficiencia y la seguridad, debemos conocer y utilizar un conjunto de posibilidades que nos proporciona el lenguaje SQL y que aún no conocemos.

Esta práctica tiene por objetivo introducir alguno de estos aspectos, con lo que se dará por concluida la parte de prácticas relacionadas con SQL. No obstante, el alumno debe tener en cuenta que éste es un curso en el que se han introducido los principales elementos de SQL a un nivel no avanzado, por lo que no debe crearse la falsa idea de que domina totalmente dicho lenguaje. Sólo la experiencia y el enfrentarse al diseño y mantenimiento de sistemas de bases de datos reales puede proporcionar dicho nivel de dominio

El dominio de SQL sólo se puede adquirir mediante la experiencia, diseñando y manteniendo sistemas de bases de datos reales.

Objetivos

Esta práctica tiene como objetivos fundamentales:

- Aprender conceptos no vistos en prácticas anteriores sobre SQL.
- Comprobar la utilidad de los diferentes tipos de datos soportados por MySQL.
- Aprender algunos conceptos básicos relacionados con la optimización y seguridad de bases de datos relacionales.

Entorno

Si el alumno desea realizar las prácticas en su ordenador deberá tener configurado su ordenador tal y como se indico en los enunciados de las prácticas anteriores.

Restricciones sobre dominios

Introducción

Como ya sabemos, los dominios nos permiten introducir restricciones sobre los valores que puede tomar un determinado atributo.

Así, el tipo de datos que definamos para un atributo le da su dominio básico. Además ya hemos utilizado las restricciones de: clave primaria, clave foránea, unicidad y permisividad de valores nulos, para introducir restricciones adicionales sobre los valores que un atributo o conjunto de atributos puede tomar.

Los tipos de datos utilizados hasta ahora son los más básicos y habituales: numéricos, cadenas de caracteres y fechas. Sin embargo, en muchas aplicaciones se nos presentarán necesidades de almacenar otros tipos de datos. En esta parte de la práctica vamos a ver algunos de esos otros tipos de datos.

Además aprenderemos el uso de los valores por defecto para un atributo, de tal manera que no sea obligatorio indicar un dato para dichos atributos cuando se introducen o modifican tuplas.

*Además de utilizar el comando **help**, para examinar la sintaxis y uso de los diferentes comandos, deberías utilizar la documentación de MySQL, que, como ya sabes, se encuentra disponible en http://dev.mysql.com/doc/. En concreto, para esta práctica te será muy útil la parte del manual de referencia relativa a procedimientos almacenados: http://dev.mysql.com/doc/refman/5.6/en/stored-programs-views.html*

Nuevos tipos de datos

Como podrás ver en la ayuda, MySQL define 31 tipos de datos básicos, además de un conjunto de tipos de datos relacionados con el almacenamiento de información geométrica.

Estos tipos de datos se pueden clasificar en 4 grupos básicos: cadenas de caracteres, numéricos, temporales (fecha/hora) y binarios.

Es importante seleccionar siempre el tipo de datos más adecuado para un dominio, ya que, por una parte definirá el conjunto de operaciones que podremos hacer sobre los datos almacenados y, por otra, definirá la cantidad de espacio necesario para almacenar dichos datos. Aunque hasta ahora seguramente hayas elegido uno al azar entre los diferentes tipos numéricos, cadenas de caracteres o temporales cuando has necesitado utilizar uno, cuando diseñes un sistema real deberás dedicar la atención necesaria a elegir el dominio de cada atributo.

En esta práctica vamos a usar algunos de los tipos de datos que aún no hemos utilizado.

Ejercicio 1:

Modifica tu base de datos de películas para que las relaciones que guardan la información de directores y actores tengan un atributo adicional que almacene la fotografía de dicho actor/director.

Prueba su funcionamiento, creando un programa que permita almacenar y recuperar imágenes de los actores/directores en formato jpg.

Es muy importante que dediques el tiempo necesario a seleccionar el tipo de datos más adecuado para cada atributo. De ello dependerán las operaciones que puedas realizar posteriormente sobre dichos datos y también las prestaciones en cuanto a espacio y rendimiento de tu base de datos.

Ejercicio 2:

Modifica tu base de datos de películas para que junto a cada película se almacene un valor entre 0,00 y 9,99 que represente tu valoración de la película.

si pongo 0.10 tiene que salir eso, o puede salir 0.1?

Valores por defecto

A veces, un determinado atributo suele tomar el mismo valor para la mayoría de las tuplas. En este caso, podemos evitar el tener que indicar dicho valor cada vez que introduzcamos una nueva tupla especificando un valor por defecto para él. De esta forma sólo habrá que indicar el valor del atributo cuando éste no sea el de defecto.

Por ejemplo, es habitual, para atributos que admiten valores nulos, indicar dicho valor como valor por defecto.

Para indicar un valor por defecto para un atributo se utiliza la cláusula `DEFAULT` a la hora de definir ese atributo en el comando `CREATE TABLE`.

Ejercicio 3:

Modifica las relaciones de tu base de datos para que la nacionalidad por defecto de las películas, actores y directores sea "Estadounidense".

Prueba el funcionamiento, introduciendo alguna tupla sin indicar un valor para la nacionalidad.

Además de valores constantes, también se pueden utilizar funciones de nuestro SGBD (ya definidas, o definidas por nosotros) que devuelvan un valor del tipo adecuado para definir un valor por defecto. De hecho, algunas de las funciones definidas en MySQL tienen éste como su cometido fundamental.

Ejercicio 4:

Modifica las relaciones de tu base de datos añadiéndoles un atributo que almacene el día y la hora en que cada una de las tuplas se introdujo en la base de datos.

Utiliza una función para obtener el valor por defecto de este nuevo atributo, de tal forma que le asigne a cada nueva tupla el instante en que ésta se introduce en el sistema. Utiliza para ello la función `NOW()`.

Claves asignadas por el sistema

En muchas ocasiones nos interesa utilizar un número entero único para identificar cada una de las tuplas de una relación. Este valor, que no tendrá ningún significado semántico, será la clave primaria de la relación, ya que identifica de manera unívoca cada tupla.

Esta aproximación se toma a menudo cuando la clave de una relación es compleja (formada por varios atributos, que además no son enteros) por motivos de eficiencia (por ejemplo, será mucho más rápido realizar una operación de join). Incluso hay autores que defienden que en un diseño de base de datos se debería tomar esta aproximación para todas las relaciones.

Puesto que este tipo de claves no tienen ningún significado semántico y, por tanto, da igual que

valor se le asigne a cada tupla, se suele delegar en el SGBD la gestión de estos valores, de tal forma que no tengamos que gestionar explícitamente cuestiones como cuál es el siguiente valor no asignado para utilizarlo en una nueva tupla, etc. Esto se consigue utilizando la cláusula AUTO_INCREMENT cuando se define el atributo, que define un valor por defecto especial para el atributo que tiene el comportamiento indicado.

Si introduzco un valor explícito lo añade,
pero el siguiente que no indique, contará a partir del último que añadí

Aunque elimine sigue a partir del ultimo asignado, no lo recupera

Ejercicio 5:

Modifica tu base de datos para que todas las relaciones utilicen claves asignadas por el sistema (obviamente, tendrás que modificar también tus claves foráneas). Deberás hacer todas tus modificaciones de forma que toda la base de datos quede en estado consistente sin tener que volver a cargar los datos.

Comprueba el comportamiento de las nuevas claves introduciendo alguna tupla adicional sin indicar valor para la clave.

Comprueba si el SGBD reutiliza los valores que se liberan al borrar tuplas y qué ocurre cuando indicas un valor explícitamente para la clave asignada por el sistema que no sea el que correspondería en la secuencia natural.

A las claves foraneas si no tiene valor se va autoincrementando?
es decir, si a una pelicula no le doy un valor para la clave foranea de ID Director autoincrementa?
Es decir, le asigna cq director????

Vistas

Introducción

Como ya sabes, el nivel de vistas nos permite particularizar la "imagen" que se le da de la base de datos a cada tipo de usuarios.

Esto facilita no sólo el mantenimiento de los datos por parte de esos usuarios si no que también implementa un nivel de seguridad, de tal forma que a cada usuario sólo se le hacen visibles aquellos datos a los que nos interesa que tenga acceso.

Creación de vistas

En SQL las vistas se definen mediante el comando `CREATE VIEW`. Una vista se define siempre en función de un comando `SELECT` que dará como resultado aquello que queremos hacer visible a través de dicha vista.

Ejercicio 6:

Crea una vista que permita ver sólo el título y la nacionalidad de una película.

Define alguna consulta sobre dicha vista.

de UNA pelicula? Me sale de todas

Ejercicio 7:

Define una vista que dé acceso al título de una película, su nacionalidad, su director y los actores que participan en ella.

Define alguna consulta sobre dicha vista.

Ejercicio 8:

Define una vista que dé acceso a todos los datos de las películas del siglo XX.

Define alguna consulta sobre dicha vista.

preguntar si la tengo bien

Modificaciones a través de vistas

Puesto que las vistas muchas veces evitan el acceso a determinada información, no siempre es posible realizar operaciones que supongan modificación de datos sobre ellas.

Comprueba experimentalmente si tus conclusiones eran correctas.

Ejercicio 9:

Razonar sobre qué operaciones de modificación de datos (INSERT, DELETE, UPDATE) son posibles a través de las tres vistas definidas en los ejercicios anteriores.

Ejercicio 10:

Modifica tu base de datos para que se puedan realizar el mayor número de operaciones posibles a través de las vistas definidas.

Otros elementos de SQL

Gestión de permisos

Una parte fundamental de la adecuada definición de una base de datos es la definición de permisos.

Así, por poner un ejemplo obvio, si definimos las vistas que hemos definido anteriormente para que determinados usuarios no puedan acceder a determinados datos, pero a esos usuarios no les impedimos el acceso directo a las relaciones, éstos siempre podrán realizar un `SELECT, INSERT,` … directo sobre las relaciones que forman nuestra base de datos y hacer inútil, por tanto, nuestro intento de proteger el acceso a determinados datos mediante vistas.

En SQL se puede gestionar directamente, para cada usuario qué operaciones concretas pueden hacer sobre cada uno de los elementos que forman nuestra base de datos (cada relación, cada vista, cada procedimiento almacenado, …). Para realizar esta gestión de permisos se utilizan básicamente dos sentencias:

- `GRANT`: Para dar permisos.
- `REVOKE`: Para quitar permisos.

Ejercicio 11:

Crea dos nuevos usuarios en tu SGBD haz las modificaciones de permisos necesarias para que cada uno de ellos sólo pueda acceder a tu base de datos mediante las tres vistas definidas anteriormente.

Índices

Puesto que, como hemos visto en el tema 4 de teoría, los índices son un elemento que influye de manera fundamental en la eficiencia de las operaciones sobre los datos de nuestras bases de datos, debemos conocer cómo utilizarlos en la práctica.

En SQL se puede definir un índice sobre una o varias columnas, tanto si esa columna o conjunto de columnas presentan la característica de unicidad como si no. Además se pueden definir índices sobre parte de un atributo (por ejemplo, los primeros *n* caracteres de un atributo)

Algunos gestores/motores de bases de datos permiten elegir el tipo de índice a crear. No es el caso de MySQL con el motor InnoDB que estamos utilizando, donde todos los índices que definamos serán de tipo *B-tree*.

Normalmente crearemos nuestros índices en el proceso de creación de las relaciones mediante las opciones correspondientes del comando `CREATE TABLE`. Además, SQL nos proporciona los comandos `CREATE TABLE/DROP INDEX` para crear o eliminar índices sobre una relación con posterioridad a la creación de ésta.

Ejercicio 12:

Crea una nueva base de datos con dos relaciones iguales que tengan dos atributos, uno entero y uno que almacene cadenas de caracteres. Sobre una de ellas define dos índices, uno para cada atributo, y en la otra no los definas.

Utiliza el procedimiento que estimes oportuno para llenarlas con gran cantidad (miles) de datos generado aleatoriamente[1].

Comprueba las diferencias de prestaciones realizando diferentes tipos de consultas sobre ellas.

No me añade las cadenas de texto con LOAD procedimiento que hice en un .txt

Ejercicio 13:

Define y realiza un experimento que permita comprobar las diferencias de prestaciones al realizar un join entre dos relaciones utilizando atributos que estén indexados y atributos que no lo estén.

A QUE SE REFIERE CON LAS PRESTACIONES?

[1] Para operaciones como ésta te puede ser muy útil el comando SQL: `LOAD DATA`.

Resumen

Los principales resultados esperados de esta práctica son:

- Conocer la utilidad de otros comandos y tipos de datos de SQL.
- Comprobar el funcionamiento de los índices y los permisos.
- Conocer la utilización de las vistas SQL.

Como trabajo adicional del alumno, se proponen las siguientes líneas:

- Mira en el manual de referencia de las versión de MySQL que estés utilizando qué comandos SQL no conoces todavía y analiza su funcionalidad.
- Crea algún ejemplo que te permita comprobar la utilidad de dichos comandos.