Sistemas de Información

Práctica 3 Curso 2016/17 3º curso

Agrupando sentencias SQL

hecha en BD: Practica2_buena

Introducción

Aunque la utilización de sentencias SQL de manera individual e interactiva permite realizar multitud de operaciones sobre una base de datos, en ocasiones esta forma de trabajar puede no ser suficiente.

En esta práctica introduciremos diferentes herramientas que nos van a permitir utilizar SQL como muchos otros lenguajes de programación, combinando sentencias con instrucciones de control flujo y utilizando variables para almacenar resultados.

Además de introducir estos conceptos, comunes a la mayoría de los lenguajes de programación y, por tanto, ya conocidos, también abordaremos otros conceptos más específicos del ámbito de las bases de datos y también relacionados con la combinación de varias sentencias:

- las transacciones, que nos permiten agrupar varias sentencias, de tal forma que el SGBD siempre asegurará que se realizan, o bien todas, o bien ninguna de ellas.

- los disparadores, que nos permitirán asociar un conjunto de sentencias a una operación sobre la base de datos, lo cual nos permitirá implementar funcionalidades como pueden ser determinadas restricciones de integridad o funciones de auditoría de cambios.

Los procedimientos almacenados, los disparadores y las transacciones nos permiten ampliar las posibilidades de utilización de nuestras bases de datos.

Al finalizar está práctica seremos capaces de utilizar estas herramientas para completar nuestras bases de datos definiendo sobre ellas operaciones y restricciones específicas para ellas.

Objetivos

Esta práctica tiene como objetivos fundamentales:

- Aprender a combinar sentencias SQL en procedimientos almacenados, y comprobar su utilidad.
- Aprender a utilizar variables SQL.
- Comprender y comprobar el concepto de transacción.
- Aprender a utilizar los disparadores y comprender sus posibilidades.

Entorno

Si el alumno desea realizar las prácticas en su ordenador deberá:

1. Tener instalado el entorno MySQL.
2. Configurar dicho entorno siguiendo las indicaciones dadas en el enunciado de la práctica 1.

Procedimientos almacenados

Introducción

Los procedimientos almacenados de SQL permiten definir rutinas que agrupen varias sentencias SQL, proporcionando elementos de control de flujo y la posibilidad de definir y utilizar variables.

Una vez definido, un procedimiento almacenado se compila y se almacena en el catálogo de la base de datos pudiéndose, a partir de ese momento ejecutar cuantas veces queramos y desde cualquier punto en el que se pueda ejecutar una sentencia SQL (intérprete de comandos SQL, otro procedimiento almacenado, un disparador, ….)

La utilización de procedimientos almacenados, además de la ventaja obvia de permitir agrupar sentencias simples para realizar tareas complejas, presentan otra serie de ventajas que los hacen una opción muy interesante incluso para realizar tareas simples. En los siguientes ejercicios veremos algunas de estas ventajas.

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/5.7/en/stored-programs-views.html*

Creando un procedimiento almacenado

Los procedimientos almacenados se crean con la sentencia `CREATE PROCEDURE`.

Una vez creado, un procedimiento se puede ejecutar desde nuestro intérprete interactivo *mysql* mediante la sentencia `CALL`.

En esta práctica vamos a crear un procedimiento almacenado muy simple que no tome parámetros y que ejecute una sentencia SQL simple. Aunque puede parecer un poco absurdo definir un procedimiento que ejecute una sola sentencia, ésta es una práctica que se utiliza a veces, ya que los procedimientos almacenados presentan algunas ventajas sobre las sentencias introducidas interactivamente en el momento que se quieren utilizar. Así:

- Un procedimiento almacenado se compila, se optimiza y se almacena en el catálogo de la base de datos en el momento en que se define. Esto hace que muchas veces, la ejecución de un procedimiento almacenado sea más rápida que la de la sentencia interactiva correspondiente.
- El nombre y lista de parámetros de un procedimiento almacenado suelen ser bastante más cortos que una sentencia compleja completa, por lo que permite reducir el tráfico entre el cliente y el servidor.
- Los procedimientos almacenados son una forma de reutilizar código, por lo que si vamos a utilizar muchas veces la misma sentencia, almacenarla como un procedimiento almacenado es una buena opción.

Ejercicio 1:

Crea un procedimiento almacenado *listar_directores_peliculas()* en tu base de datos de películas que no tome parámetros y presente la lista de títulos de todas las películas junto con el nombre de su director.

Prueba su correcto funcionamiento.

Variables

Dentro de los procedimientos almacenados de SQL podemos definir y utilizar variables para almacenar valores que necesitemos en pasos posteriores de la ejecución del procedimiento.

Las variables en SQL son tipadas y las principales sentencias relacionadas con su uso son:

- `DECLARE`, para declarar una nueva variable.
- `SET`, para asignar un valor a una variable
- Cláusula `INTO` que se define en algunas sentencias como puede ser `SELECT` para almacenar valores resultado de una operación en una variable.

La utilización de variables, como en cualquier otro lenguaje de programación, son un elemento fundamental para poder elaborar programas de una cierta complejidad y, en definitiva, lo que estamos haciendo al escribir procedimientos almacenados es escribir programas en SQL.

Ejercicio 2:

Crea un procedimiento almacenado *contar_directores()* en la base de datos de películas que no tome parámetros y haga lo siguiente:

- Cada vez que se ejecute guardará en la relación *cuentaDirectores* la fecha y hora actual y el número de directores almacenados en la base de datos en ese instante.
- Si cuando se ejecute aún no existe en la base de datos la relación *cuentaDirectores*, deberá crearla antes de almacenar la primera tupla.

Variables de sesión

Las variables tienen un ámbito, que en SQL es el siguiente:

- Si una variable se declara dentro de un procedimiento, su ámbito es dicho procedimiento en el que se definen.
- Si una variable de declara dentro de un bloque BEGIN-END, su ámbito es dicho bloque.

Por tanto, en principio no podremos acceder al valor almacenado en variables desde fuera de los procedimientos. Sin embargo, SQL define un tipo especial de variables, que se denominan "variables de sesión" y que existen desde el momento en que las definimos hasta el fin de la sesión en que se hayan definido.

Ejercicio 3:

Modifica el procedimiento del ejercicio anterior, de tal manera que podamos saber, desde la línea de comandos de *mysql*, cuántas veces se ha llamado a dicho procedimiento en la sesión actual.

Paso de parámetros

Como en muchos otros lenguajes, los procedimientos almacenados de SQL permiten el paso de parámetros en su definición, de tal manera que podamos condicionar su funcionamiento en tiempo de ejecución en función de los valores que le demos a dichos parámetros.

Los parámetros pueden ser de tres tipos:

- `IN`: Parámetro de entrada.
- `OUT`: Parámetro de salida.
- `INOUT`: Parámetro de entrada y salida.

En las prácticas de este apartado vamos a practicar estos tres tipos de parámetros. Cuando tengas que utilizar parámetros de salida (`OUT` o `INOUT`), utiliza variables de sesión en la llamada para poder comprobar después los resultados.

Ejercicio 4:

Modifica tu base de datos de películas para añadir a cada una de las relaciones *Peliculas*, *Directores* y *Actores* un nuevo atributo *nacionalidad*.

Crea un procedimiento almacenado *consultaPorNacionalidad()* que tome como parámetro de entrada una nacionalidad y muestre todas las películas, directores y actores con dicha nacionalidad.

Ejercicio 5:

Crea un procedimiento almacenado *peliculasPorNacionalidad()* que tome como parámetro de entrada una nacionalidad y devuelva en un segundo parámetro entero el número de películas que tienen dicha nacionalidad.

Ejercicio 6:

Crea un procedimiento almacenado *ponerEnMayusculas()* que tome como parámetro una cadena de caracteres y la convierta a letras mayúsculas[1].

[1] Como podemos ver en este ejemplo, en un procedimiento almacenado SQL no tiene por qué haber interacciones con la base de datos.

Control de flujo

Las principales sentencias de control de flujo que proporciona SQL para su utilización en los procedimientos almacenados son similares a las que podemos encontrar en otros lenguajes de programación y que ya conocemos:

- IF y CASE para ejecución condicional.
- WHILE, REPEAT y LOOP para ejecución iterativa.

Ejercicio 7:

Modifica el procedimiento almacenado *contar_directores()* que creaste en los ejercicios 2 y 3 para que, si la relación *cuentaDirectores* tiene 10 tuplas, elimine la más antigua antes de introducir la nueva. De esta forma, la relación guardará siempre las 10 últimas entradas

Cursores

Los cursores son un elemento fundamental en la programación sobre bases de datos y que aparecerá con frecuencia en próximas prácticas.

Los cursores permiten realizar operaciones sobre cada uno de los resultados de una consulta, de tal manera que podremos acceder al resultado de dichas consultas tupla a tupla para realizar la operación correspondiente.

Normalmente se utilizan asociados a bucles, de tal forma que dentro del bucle, en cada iteración accederemos a una tupla, operaremos sobre ella y pasamos a la siguiente iteración.

Las principales sentencias relacionadas con cursores en SQL son:

- `DECLARE CURSOR`: para declarar un nuevo cursor.
- `OPEN`: para abrir el cursor y empezar a utilizarlo.
- `FETCH`: para obtener la siguiente tupla del resultado de la consulta. Asociada a esta sentencia tenemos la cláusula `INTO` que nos per-

mite almacenar en variables los valores de los atributos de la tupla actual.
- `CLOSE`: para cerrar el cursor cuando no se va a utilizar más.

Además, para detectar que hemos llegado al final del resultado de la consulta (ya no quedan más tuplas) y finalizar el bucle, debemos definir un handler, mediante la sentencia `DECLARE CONTINUE HANDLER FOR NOT FOUND`.

Ejercicio 8:

Crea un procedimiento almacenado *extraer_imdbs()* que tome como parámetro una nacionalidad y cree una nueva relación con el nombre de dicha nacionalidad y que almacene los códigos Imdb de todas las películas, actores y directores que tengan esa nacionalidad[2].

[2] Este procedimiento se puede implementar sin utilizar cursores, pero deberás utilizarlos obligatoriamente para practicar su uso.

Transacciones

Introducción

Ya conocemos el concepto de transacción: una transacción está formada por un conjunto de operaciones para el que necesitamos que el SGBD garantice que, o bien se realizan todas, o bien no se realiza ninguna.

Las transacciones son habituales en un entrono de bases de datos, de su gestión se encarga el SGBD, y el lenguaje SQL nos proporciona las sentencias necesarias para su adecuada utilización.

A la hora de utilizar transacciones ten en cuenta que hay determinadas operaciones que no se pueden deshacer y que, por tanto, no les afecta la utilización de transacciones. Son las sentencias `CREATE …`, `ALTER …` y `DROP …` Debido a esto, las transacciones se suelen utilizar solamente con sentencias DML.

Por defecto, MySQL considera cada operación que supone una modificación una transacción por sí misma. Por tanto, si queremos definir una transacción que englobe varias sentencias debemos variar este comportamiento e indicar que seremos nosotros quienes gestionaremos manualmente las transacciones. Para hacer esto debemos introducir la siguiente sentencia en:

```
mysql> SET autocommit = 0;
```

Comprobando el funcionamiento de las transacciones

En esta práctica vamos a comprobar el funcionamiento de las transacciones. Para ello vamos a comenzar una transacción, realizarla parcialmente y provocar un fallo que impida que la transacción se complete.

Para ello, debemos conocer las sentencias SQL relacionadas con la gestión de transacciones. Son las siguientes:

- START TRANSACTION: Comienza una nueva transacción.
- COMMIT: Confirma el final de una transacción, lo que hace que todas las modificaciones se hagan definitivas.
- ROLLBACK: Aborta la transacción actual, dejando, por tanto, la situación de la base de datos en el estado anterior al comienzo de la transacción.

falta de 9 en adelante

Ejercicio 9:

Deshabilita la gestión automática de transacciones, comienza manualmente una nueva transacción, realiza un par de operaciones que supongan una modificación de los datos almacenados en tu base de datos y, antes de concluir la transacción, simula un problema en tu SGBD[3].

Una vez recuperado el SGBD observa qué ha pasado con los cambios que habías realizado.

Repite el experimento, pero simulando el problema después de confirmar la transacción.

no me da tiempo a apagarla

[3] Para simular un problema, puedes apagar la máquina en la que estés ejecutando el SGBD. Si estás trabajando con tu máquina, de forma alternativa puedes matar el proceso gestor como superusuario.

Utilizando transacciones

Ejercicio 10:

Crea un procedimiento almacenado denominado *introducir_pelicula()* que tome como parámetros todos los datos necesarios para introducir en nuestra base de datos de películas una nueva película junto con su director.

El procedimiento debe realizar la totalidad de la operación o bien dejar la base de datos inalterada.

Disparadores

Introducción

Un disparador es un tipo especial de procedimiento almacenado que se utiliza con mucha frecuencia en la programación de bases de datos.

La diferencia fundamental con un procedimiento normal es que no son ejecutados directamente por el usuario u otro programa, si no que un disparador se ejecuta o dispara automáticamente cuando ocurre un determinado evento con el que éste está asociado.

Los eventos a los que se pueden asociar disparadores son inserciones, modificaciones o eliminaciones de datos sobre una determinada relación.

Los disparadores son muy útiles para realizar determinadas operaciones de comprobación de integridad de los datos que almacena nuestra base de datos, así como para monitorizar determinados cambios en los datos almacenados. No obstante, debemos tener en cuenta que suponen una sobrecarga para el SGBD.

Un disparador se puede definir para que se ejecute antes o después de la operación que lo disparará. Por tanto, tenemos 6 tipos distintos de disparadores: `BEFORE INSERT`, `AFTER INSERT`, `BEFORE UPDATE`, `AFTER UPDATE`, `BEFORE DELETE`, `AFTER DELETE`.

Por último indicar que no todas las sentencias de SQL se pueden utilizar en un disparador. Así, no tiene sentido utilizar sentencias que producen una salida, como puede ser `SHOW`.

Creando disparadores

Las sentencias relacionadas con la gestión de disparadores son:

- `CREATE TRIGGER`: para definir un nuevo trigger. En la definición se indican las condiciones de disparo del disparador.
- `DROP TRIGGER`: elimina un disparador de la base de datos.

En esta práctica vamos a ver el funcionamiento de los disparadores sobre nuestra base de datos de películas. Ten en cuenta que la ejecución de un disparador es un efecto lateral de una operación sobre la base de datos. Por tanto no podrás ver su funcionamiento directamente, sólo lo podrás comprobar indirectamente viendo sus efectos.

Ejercicio 11:

Define disparadores asociados a las relaciones de películas, actores y directores para que mantengan automáticamente las relaciones que has definido en ejercicios anteriores para almacenar los códigos *imdb* de cada nacionalidad que definiste en el ejercicio 8.

Ejercicio final

Ejercicio 12:

Utiliza las herramientas que has aprendido en esta práctica para conseguir que en la base de datos que estamos creando la asociación varios-a-varios entre películas y actores presente la característica de participación total respecto a las películas. Es decir, para asegurar que en nuestra base de datos no pueda haber una película sin actores asociados.

Resumen

Los principales resultados esperados de esta práctica son:

- Ser capaces de crear procedimientos almacenados en SQL.
- Comprobar el funcionamiento de las transacciones y aprender su utilización.
- Aprender a definir y utilizar disparadores.

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

- Trabaja en completar tu base de datos de películas definiendo nuevas relaciones y diseñando y programando procedimientos almacenados para operaciones que consideres habituales.
- Prueba los diferentes tipos de disparadores e intenta encontrar usos útiles para cada uno de ellos.
- Intenta utilizar las herramientas aprendidas en esta práctica para intentar resolver los diez posibles tipos de asociaciones que pueden existir entre dos entidades.