Grupo 29

Item 3. Stored Procedure

Manual de instrucciones

Khawla Al Asfar

Juan Carlos Utrilla Martín

Juan Rodríguez Dueñas

Yassine Taziny

José Manuel Lara Morilla

1. Contenido

1. Introducción

Los procedimientos almacenados (stored procedure) son un medio para implementar algunos servicios en bases de datos. Estos procedimientos se esperan que sean más eficientes que sus homólogos en Java. Estos permiten agrupar de forma exclusiva parte de algo específico que se desee realizar.

Estos procedimientos se presentan con una ventaja por ejemplo cuando una base de datos es manipulada desde muchos programas externos. Al incluir la lógica de la aplicación en la base de datos utilizando procedimientos almacenados, la necesidad de embeber la misma lógica en todos los programas que acceden a los datos es reducida.

Aunque no están exentos de inconvenientes ya de que son más difíciles de implementar y depurar.

2. Entorno tecnológico

Para todas las pruebas que hemos realizado utilizaremos MYSQL como gestor de bases de datos.

3. Implementación

La sintaxis que se utiliza para implementar un procedimiento almacenado (stored procedure) es la siguiente:

```
CREATE
[DEFINER = { user | CURRENT_USER }]
PROCEDURE sp_name ([proc_parameter[,...]])
[characteristic ...] routine_body
```

La tarea a realizar consiste en realizar un procedimiento almacenado que calcule una tabla con una tupla de la forma (m, a, b) para cada manager m, donde a denota el número de explorer que tienen una solicitud aceptada para los trips que gestiona cada manager m y b denota el cuartil correspondiente.

Comenzamos refiriéndonos a la base de datos con la que vamos a trabajar:

```
USE `acme-explorer`;
```

Para facilitar la tarea, hemos desarrollado una función en la que, según el porcentaje recibido, devuelva el cuartil que corresponda.

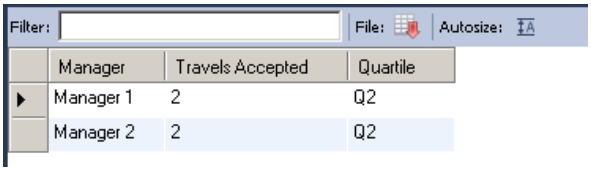
```
BEGIN
    DECLARE result VARCHAR(255) DEFAULT "";

    SET calc = ROUND(calc, 0);

    CASE
        WHEN (calc <= 25) THEN SET result = "Q1";
        WHEN (calc <= 50) THEN SET result = "Q2";
        WHEN (calc <= 75) THEN SET result = "Q3";
        ELSE SET result = "Q4";
    END CASE;

    RETURN result;
END //
```

Por último, realizaremos el procedimiento. Debemos tener en cuenta que en Mysql necesitamos acotar el “procedure” mediante delimitadores con la etiqueta DELIMITER. Como se puede observar en el código, declaramos variables y realizamos una query adicional para obtener el total de trips aceptados.

```
DELIMITER //
CREATE PROCEDURE `Acme-Explorer`.`getTripsAcceptedByExplorer` ()
BEGIN
    # Declaramos las variables que vamos a necesitar
    DECLARE totalAcceptedTrip INT DEFAULT 0;

    SELECT  count(t.id) INTO totalAcceptedTrip
        FROM MANAGER m
        INNER JOIN TRIP t ON (m.id = t.manager_id)
        INNER JOIN APPLICATION a ON (t.id = a.trip_id)
        WHERE a.status LIKE "ACCEPTED";

    # Ejecutamos la query
    SELECT  m.name AS 'Manager', count(t.id) AS 'Travels Accepted',
            quartile((count(t.id)/totalAcceptedTrip) * 100) AS 'Quartile'
        FROM MANAGER m
        INNER JOIN TRIP t ON (m.id = t.manager_id)
        INNER JOIN APPLICATION a ON (t.id = a.trip_id)
        WHERE a.status LIKE "ACCEPTED"
        GROUP BY m.name;

END //
DELIMITER ;
```

4. Pruebas

Tal y como podemos ver desde Mysql, el procedimiento se ejecuta correctamente.

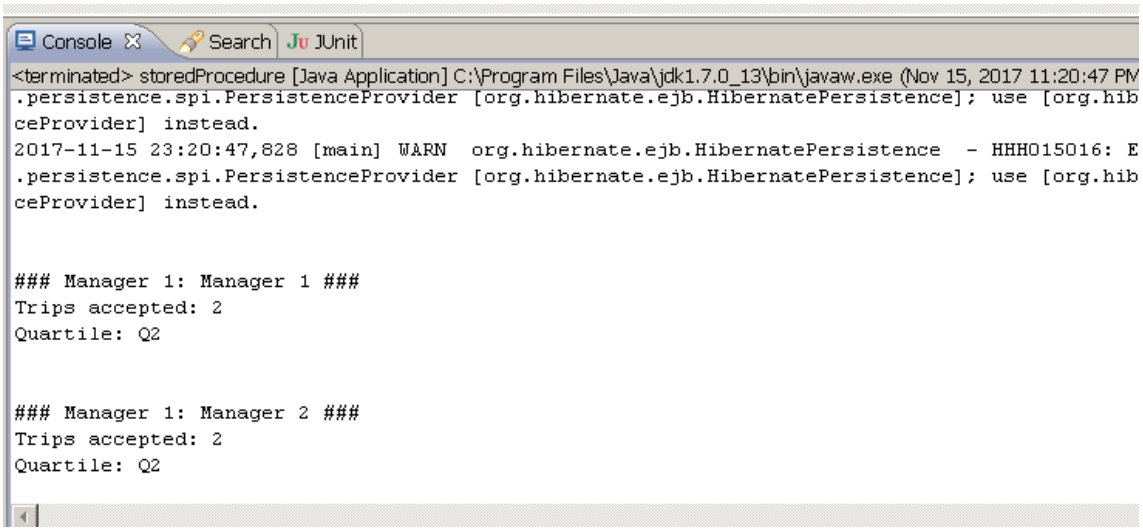


Filter: | File: | Autosize:

	Manager	Travels Accepted	Quartile
▶	Manager 1	2	Q2
	Manager 2	2	Q2

Para verificar que esto funciona correctamente en Java, hemos creado una clase en el paquete Utilities de nombre storedProcedure.java, en el que podemos realizar el testing.

```
CurriculumServiceTest.java | CurriculumService.java | TransactionAspectSupport.class | CreditCard.java | Curriculum.java | storedProcedure.java

public class storedProcedure {

    @SuppressWarnings("unchecked")
    public static void main(final String[] args) {
        EntityManagerFactory entityManagerFactory;
        EntityManager entitymanager;

        entityManagerFactory = Persistence.createEntityManagerFactory("Acme-Explorer");
        entitymanager = entityManagerFactory.createEntityManager();

        final Query query = entitymanager.createNativeQuery("call getTripsAcceptedByExplorer()");
        final List<Object> result = query.getResultList();

        final String newLine = System.lineSeparator();

        for (final Object object : result) {
            Object obje[] = new Object[10];
            obje = (Object[]) object;
            System.out.println(newLine);
            System.out.println("###" + obje[0] + " ###");
            System.out.println("Trips accepted: " + obje[1]);
            System.out.println("Quartile: " + obje[2]);
        }
        entitymanager.close();
        entityManagerFactory.close();
    }
}
```



```
Console | Search | JUnit
<terminated> storedProcedure [Java Application] C:\Program Files\Java\jdk1.7.0_13\bin\javaw.exe (Nov 15, 2017 11:20:47 PM
.persistence.spi.PersistenceProvider [org.hibernate.ejb.HibernatePersistence]; use [org.hib
ceProvider] instead.
2017-11-15 23:20:47,828 [main] WARN  org.hibernate.ejb.HibernatePersistence  - HHH015016: E
.persistence.spi.PersistenceProvider [org.hibernate.ejb.HibernatePersistence]; use [org.hib
ceProvider] instead.


### Manager 1: Manager 1 ###
Trips accepted: 2
Quartile: Q2


### Manager 1: Manager 2 ###
Trips accepted: 2
Quartile: Q2
```

5. Referencias

- Procedimientos almacenados: https://es.wikipedia.org/wiki/Procedimiento_almacenado
- Sintaxis de stored procedure: https://dev.mysql.com/doc/refman/5.7/en/create-procedure.html
- Cuartiles: https://www.vitutor.com/estadistica/descriptiva/a_11.html