Performance Tips: Stored Prepared Statements

Table Of Contents

Introduction

This paper describes how to tune your Cloudscape database applications using Stored Prepared Statements. In this paper we look at the different types of statements and how they affect performance by measuring the response time of a web based database application. This paper should help you determine how the use of Stored Prepared Statements can improve the performance of your application.

What Type of Statement Should I Use?

There are two types of statements you can use to execute SQL using JDBC: Statements and Prepared Statements. Which one you use depends on how, or more importantly, how many times your application submits a particular query.

- If your application is an ad-hoc query tool, you would most likely use a Statement.
- If you have a SQL statement in your application in which only parameters change a Prepared Statement would be most appropriate.

Any statement must be compiled at least once before it can be executed. Compiling a Statement or Prepared Statement for execution includes: parsing the query text, binding (verifying the existence of referenced objects), optimizing and finally generating executable code. This process takes time and processor cycles to complete.

Statements

Each time a Statement is executed the query is compiled. In an ad-hoc query tool , for example, (See: “Creating A Query Tool”) the performance impact of compiling the query is negligible. In this case, you do not know the query text ahead of time and the query is usually only executed once.

An ad-hoc query generator is the exception rather than the rule. In most web applications, the query text is known ahead of time and executed many times changing only parameters. In this case you should use a Prepared Statement.

Prepared Statements

Prepared Statements are compiled one time for each connection and stored in the Cloudscape statement cache. Therefore, if you open a connection and run the same query 12 times it will only be compiled once. This can save computing time. We will look at some code from Bubba’s CloudBooks for an example of where the statement cache can be utilized. In the getSaleBooks() method, we need to get the names of the jpg files related to the books that are on sale. We have a list of five Books that are on sale and execute a “select” once for each book to get the name of the associated jpg file.

```
public Vector getSaleBooks(Vector idList)
```

```
    throws SQLException
{
    int numCols = 5;
    int colCnt = 0;
    Integer thId = null;
    Connection conn = null;
    PreparesStatement pstmt = null;

    <… Code deleted for clarity …>

    try {
        //We prepare the statement
        pstmt = conn.prepareStatement("SELECT i_picture "+
                                      "FROM item "+
                                      "WHERE i_id = ?");
        //The query is executed 5 times. 1 time for each picture

        for( int i = 0; i < 5; i++) {
            thId = (Integer) idList.get(i);
            pstmt.setInt(1,thId.intValue());
            pstmt.execute();

            result = pstmt2.getResultSet();
            result.next();

            //A ProductData object is created to store
            //the picture information.
            ProductData newProd =
            new ProductData(thId.intValue(), result.getString
(1));
            saleIdList.add(newProd);
        }
    } //End Try
    finally {
        conn.close();
    }

    <… Code deleted for clarity …>

    return(saleIdList);
}
```

Example 1

In this example, the statement is compiled once and the query is executed five times. If query compilation takes 500 milliseconds, this would save 2000 milliseconds each time this method is called .

Compiling each time:	5 x 500 = 2,500 milliseconds
Compiling once:	1 x 500 = 500 milliseconds
Savings:	2,500 – 500 = 2,000 milliseconds

This works because the single connection executes the query many times and the statement is kept in Cloudscape's statement cache.

Stored Prepared Statements

In most web applications, connections are created and destroyed often because user sessions are short-lived. As you can see in Example 1, the connection is created and destroyed within the method. Some compilation time is saved because you only have to compile the query once each time the method is called. To maintain greater than 100 method calls per second would mean compiling the query each time the method is called (100 x 500 milliseconds = 50,000 milliseconds). With so many connections executing the query it is best to use Cloudscape's Stored Prepared Statements.

Stored Prepared Statements are compiled once when they are created then stored in the database. The compiled statement is called by any connection using the "Execute Statement" SQL syntax. We will implement the query in Example 1 as a Stored Prepared Statement.

First, we create the Stored Prepared Statement using the SQL syntax:

```
Create Statement getItemPictureNameas
SELECT i_picture
FROM item
WHERE i_id = ?;
```

In the getSaleBooks() method only the SQL needs to be changed.

```
Original Syntax:
pstmt = conn.prepareStatement("SELECT i_picture "+
                              "FROM item "+
                              "WHERE i_id = ?");
```

```
New Syntax:
// Comment
// getItemPictureName = SELECT i_picture FROM item WHERE i_id = ?

pstmt =
    conn.prepareStatement("Execute Statement
getItemPictureName");
```

The rest of the method stays the same.

Coding Tip:
When using Stored Prepared Statements, put the SQL text in the comments. Debugging your code can get confusing when the query text is "Execute Statement MyStatement".

Use prepared statements while you are developing your application to simplify development and debugging. Then, before performance testing your application (which we all do before we deploy applications of course), create your Stored Prepared Statements.

The Cloudscape documentation talks in detail about the use of Prepared Statements and Stored Prepared Statements. (See Programming For Performance). After developing the eBusiness test application, "Bubbas CloudBooks ," I did some performance testing comparing Prepared Statements to Stored Prepared Statements.

Performance Measurements

In Bubba's CloudBooks, Cloudscape is running in embedded mode. Connections are opened and closed in each method that accesses the database. In this model, connections don't stay around long, so compiling the Prepared Statements had a significant impact on the performance of the application.

The graphs (Figure 1 & Figure 2) show the response times for each web interaction. Response timing starts when a page is requested and ends when the last character has been received by the client. Each line on the graph represents a different type of interaction (Home, Shopping Cart, Item Detail, etc).

Prepared Statements

The first round of testing simulated 30 web users. The two processor Sun Ultra 80 (2x295MHz) server was at

98% CPU utilization and as you can see (Figure 1), the transaction response times average about 15 seconds per transaction. Most people don't want to wait 15 seconds for each web page to display.

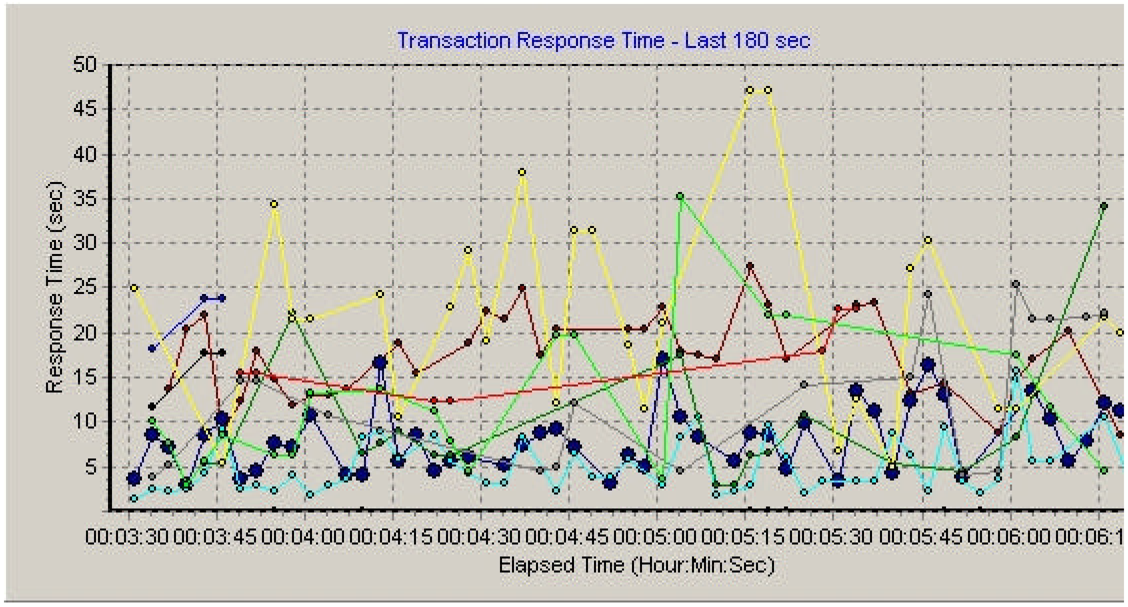

Figure 1

Stored Prepared Statements

The second run utilized Stored Prepared Statements in the same application (see Figure 2).

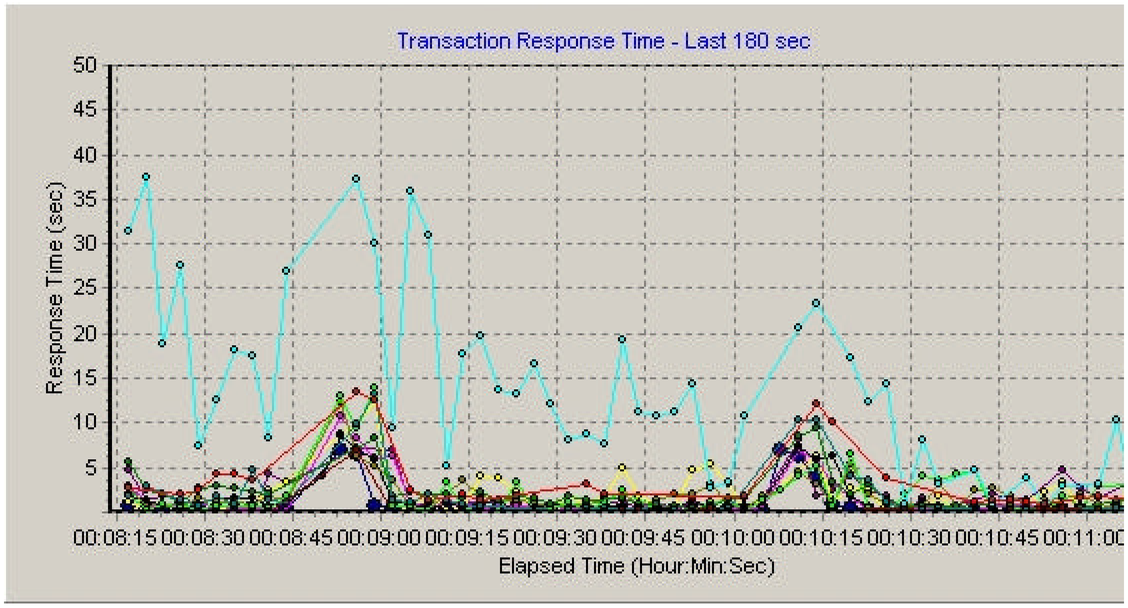

Figure 2

As you can see the average response time is now about 3 seconds instead of the 15 seconds as before. There was another difference, the second run involved 100 clients instead of 30. In this test, the average response time is 400% better with 233% more users. In this case, using Stored Prepared Statements made a significant performance difference.

Note about the graph: The two peaks you see in the graph, as you might guess, were caused by garbage collection.

Summary

Statements, Prepared Statements and Stored Prepared Statements all have uses. I have found through testing that web applications with pre-determined SQL text are much more efficient using Stored Prepared Statements.

Resource Links

Reference

Cloudscape: Programming For Performance:
 http://www.cloudscape.com/support/doc_35/doc/html/tutorial/ps.htm#806040

Bubbas CloudBooks: A Cloudscape ePerformance Test
 i-Cloudscape-Bubbas.htm

Creating A Query Tool:
 i-Cloudscape-QueryApp.html

Author:
 Scott Fadden
 Cloudscape Performance
 Informix Software, Inc.
 scottfa@informix.com