JDBC

How to use Fail-Over in Altibase JDBC

Overview

When a failure occurs while operating the database system, it provides a fail-over function and overcomes a failure and so that the service can be continued.

This section describes the functions and usage of Altibase JDBC Fail-Over.

Glossary of terms

- CTF (Connection Time Fail-Over): When a failure is detected at the time of accessing the DBMS, this refers to retrying the connection to the DBMS of another available node.
  - If the AlternateServer property is set in the Connection String, CTF operates by default.
  - An application usually should try to connect again.
  - In the form of maintaining a connection pool like WAS, it can be performed automatically through the connection validation setting of WAS.
- STF (Service Time Fail-Over): When a failure is detected while the DBMS is processing a client's request, this refers to reconnecting to the DBMS of another available node.
  - Since the success or failure is returned as an error after executing only the connection, user coding to proceed from the preparation of the statement again must be followed.
- Primary Server: This refers to the server information specified at the first in the Connection URL.
- Alternative Server: This refers to the information of servers specified in AlternateServer connection properties.
- Explicit Connection: This refers to when calling the connect method of the Connection object.
- Implicit Connection: This refers to when the connect method fails or STF occurs, and JDBC internally retries the connection to the DBMS of another available node.

Fail-over related connection properties

Case insensitive.

- AlternateServer: It refers to alternate servers and is described in (IP1:Port1, IP2:Port2,...) format.
  - Alternate servers can be the connection target server in the following cases depending on the settings of LoadBalance and SessionFailOver.
    - When calling the connect method
    - When attempting to reconnect due to a failure of the connect method
    - When attempting to reconnect due to STF occurrence

© Altibase Corp.

- ConnectionRetryCount: Set the number of retry attempts to connect to the same server.
- ConnectionRetryDelay: Waiting time for connection attempts to the same server. The unit is seconds.
  Note) ConnectionRetryCount and ConnectionRetryDelay properties are applied only for internal connection, and if it fails after executing ConnectionRetryCount * ConnectionRetryDelay, it tries to connect to another available server in the same way.
- LoadBalance: Configures the connection attempt order between the primary server and the alternate server when connecting to the DBMS.
  - ON
    - Initial Connection: Randomly attempts to connect to either the primary server or the alternate server.
    - STF: Attempts to connect to the previously connected server first, followed by random connection attempts.
  - OFF
    - Initial Connection: Attempts to connect to the primary server first, followed by the alternate server in sequential order.
    - STF: Attempts to connect to the previously connected server first, followed by the primary and alternate servers in sequential order.
- SessionFailOver: Whether to perform STF or not.
  - When set to ON, it operates as CTF+STF.
  - When set to OFF, only CTF operates.

Verification of Fail-Over Success

- CTF: Success can be determined immediately based on whether the database connection is established.
- STF: Success can be verified through exception handling. Fail-Over is successful if the return value of the getSQLState() method in SQLException is 08F01 (version 6.3.1 or later) or ES_08FO01 (version 6.1.1 or earlier).

Example

When writing a program, refer to the following example.

```
import Altibase.jdbc.driver.*;
import Altibase.jdbc.driver.ex.*;
import java.util.Properties;
import java.sql.*;
import [java.io](http://java.io).*;

class  FailOverSampleSTF
{
    public static void main(String args[])  throws Exception
    {
        //-------------------------------------------------------
        // Initialization
        //-------------------------------------------------------
```

```
        String sURL = "jdbc:Altibase://127.0.0.1:" + args[0]+"/mydb?
AlternateServers=(128.1.3.52:20300,128.1.3.53:20300)&ConnectionRetryCount=3&ConnectionRetryDelay=10&SessionFai
lOver=on&LoadBalance=off";

        try
        {
            Class.forName("Altibase.jdbc.driver.AltibaseDriver");
        }
        catch ( Exception e )
        {
            System.err.println("Can't register Altibase Driver\n");
            return;
        }

        //----------------------------------------------------
        // Test Body
        //----------------------------------------------------

        Properties sProp = new Properties();
        Connection sCon;
        PreparedStatement  sStmt = null;
        ResultSet sRes = null ;
        sProp.put("user", "SYS");
        sProp.put("password", "MANAGER");

        sCon = DriverManager.getConnection(sURL, sProp);

        sStmt = sCon.prepareStatement("SELECT C1 FROM T2   ORDER BY C1");

        while(true)
        {
            try
            {
                sRes = sStmt.executeQuery();
                while( sRes.next() )
                {
                    System.out.println( "VALUE : " + sRes.getString(1)  );
                }
            }
            catch ( SQLException e )
            {
                //Failover 확인.
                //if (e.getSQLState().equals("ES_08FO01"))   // 6.1.1 이하
                if (e.getSQLState().equals("08F01"))           // 6.3.1 이상
                {
                     sStmt = sCon.prepareStatement("SELECT * FROM tb_test1");   // Failover 발생 시 prepare를
다시 해주어야 한다.
                     continue;</span>
                }
                System.out.println( "EXCEPTION : " + e.getMessage() );
                break;
            }
            break;
        }

        sRes.close();

        //----------------------------------------------------
        // Finalize
        //----------------------------------------------------

        sStmt.close();
        sCon.close();
    }
}
```

Reference

- Replication Manual -> Chapter 4 FailOver
- JDBC User's Manual
  - Chapter 1 Getting Started with JDBC -> Connection Information
  - Chapter 3 Advanced Features -> JDBC and FailOver
- JBoss Integration Guide for Altibase

How to change the jdbc.trc file creation location

Creation time and usage

The jdbc.trc file is a log file created for JDBC-related traces, and is automatically generated when the JDBC driver is initialized.

If fail-over is specified in the connection URL, it is also used for event trace when fail-over occurs.

If fail-over is not specified in the connection url, or fail-over does not occur, it is 0 byte to the default value.

File size varies slightly from platform to platform.

When one Java program is executed at the same time, as many as the number of executions are created,

jdbc.trc
jdbc.trc.lck
jdbc.trc.1
jdbc.trc.1.lck
jdbc.trc.2
jdbc.trc.2.lck

When connecting each connection object separately in a single Java program, the jdbc.trc file is shared and used.

Version

The jdbc.trc file is created for JDBC-related traces from Altibase version 5.5.1 or later.

Creation location

If the ALTIBASE_HOME environment variable is set in the client environment, it is created in **$ALTIBASE_HOME/trc**,
If the ALTIBASE_HOME environment variable is not set, it is created in the location **where the Java program is executed.**

Sometimes, if the user does not have write permission in the directory where the jdbc.trc file is created, the following error may occur in the application due to file creation failure.

© Altibase Corp.

java.io.IOException: Couldn't get lock for /opt/altibase-HDB-server-6.1.1/trc/jdbc.trc

If the above error occurs, give permission to the directory or set the ALTIBASE_HOME environment variable to an appropriate location.

How to change the creation path

Altibase Version	How to change the jdbc.trc log creation location
version 6.1.1.1.3 or earlier	Not available
version 6.1.1.1.4 ~ 6.1.1.1.7	Can be set as an option when running java java -D**ALTIBASE_JDBC_TRCLOG_DISABLE=true**
version 6.1.1.1.8 or later	Set option when running java java -D**ALTIBASE_JDBC_TRCLOG_DISABLE=true** Or Set as an environment variable of the java execution user export **ALTIBASE_JDBC_TRCLOG_DISABLE=true**

Simultaneous connections to different Altibase versions via JDBC

Overview

This document explains how to connect to different versions of Altibase using the Altibase JDBC Driver.

Altibase provides additional JDBC drivers to enable connections to different Altibase versions.
This explanation is based on Altibase 7.3, and the additional driver file provided in version 7.3 is Altibase7_3.jar.

In the application, when loading the driver class using the Class.forName() method, you can connect to different Altibase versions by specifying the class names separately as:

- `Altibase.jdbc.driver.AltibaseDriver`
- `Altibase7_3.jdbc.driver.AltibaseDriver`

This allows simultaneous connections to different Altibase versions.

Driver location

The driver is located in $ALTIBASE_HOME/lib, and for version 7.3, the file name is Altibase7_3.jar.

```
$ ls -al
-rw-rw-rw- 1 altibase altibase  509530 Apr 20 11:10 Altibase7_3.jar
-rw-rw-rw- 1 altibase altibase  506968 Apr 20 11:10 Altibase.jar
```

© Altibase Corp.

Version check

Check the driver for connecting to other versions

```
$ java -jar Altibase7_3.jar
Altibase 7.3.0.1.0 with CMP 7.1.8 for JDBC 4.2 compiled with JDK 8
```

Usage example

The example explains simultaneous connections to versions 7.1 and 7.3.

Use the Altibase.jar file from version 7.1 located at $ALTIBASE_HOME/lib/ and the Altibase7_3.jar file from version 7.3 located at $ALTIBASE_HOME/lib/.

```
// - AltibaseMultiversionConnection.java file
// Altibase 7.3 Driver class - using Altibase7_3.jar file
Class.forName("Altibase7_3.jdbc.driver.AltibaseDriver");
// Altibase 7.1 Driver class - using Altibase.jar file
Class.forName("Altibase.jdbc.driver.AltibaseDriver");
// Altibase 7.3 version URL
String db_url1 = "jdbc:Altibase://192.168.1.111:20300/mydb";
// Altibase 7.1 version URL
String db_url2 = "jdbc:Altibase://192.168.1.222:20300/mydb";
```

© Altibase Corp.