JDBC

JDBC Overview

The set of database manipulation interfaces created in the Java programming language is called JDBC (Java DataBase Connectivity). A set of APIs that provide a consistent interface for a variety of relational databases, defining a set of object-oriented classes of classes that the programmer will use to build SQL requests. That is, if you use a JDBC driver, no matter which database you use, there is an advantage that you can apply it directly without modifying the code.

Standard JDBC Functions

Standard Function Specs 4.0

Extension JDBC Functions

setIPv4

```
void setIpv4(int ind, String ipString)
```

This is a function to input IPv4 address type in PrepareStatement.

Receives column index and IPv4 string as arguments.

setIPv6

```
void setIpv6(int ind, String ipString)
```

This is a function to input IPv6 address type in PrepareStatement.

Receives column index and IPv6 string as arguments.

ExecuteAppendOpen

```
ResultSet executeAppendOpen(String aTableName, int aErrorCheckCount)
```

Opens the protocol to write the Append protocol in the Statement.

The table name and error checking interval are received as arguments. Returns a ResultSet with the result value.

executeAppendData

```
int executeAppendData(ResultSetMetaData rsmd, ArrayList aData)
```

Enters the actual data for the Append protocol in the statement.

Receives the metadata of the ResultSet, which is the result value of executeAppendOpen, and the data to input. When the result value is stored in the transfer buffer, 1 is returned. If the transfer buffer is transferred to Machbase, 2 is returned. Therefore, if 1 or 2 is returned, it is judged as success.

executeAppendDataByTime

```
int executeAppendDataByTime(ResultSetMetaData rsmd, long aTime, ArrayList aData)
```

Enters the actual data for the Append protocol on a time basis in the statement.

Receives the metadata of the ResultSet which is the result value of executeAppendOpen, the time value of the specific time zone to be set, and the data to input as arguments. If the result value is stored in the transmission buffer, 1 is returned.

executeAppendClose

```
int executeAppendClose()
```

Terminates the statement for the Append protocol in the statement.

If the result is a success, it returns 1.

executeSetAppendErrorCallback

```
int executeSetAppendErrorCallback(MachAppendCallback aCallback)
```

Sets a callback function that outputs an error if an error occurs during Append execution.

It takes a callback function that outputs an error log as an argument. If the result is successful, 1 is returned.

getAppendSuccessCount

```
long getAppendSuccessCount()
```

Returns the number of successes for the Append protocol in the Statement.

Returns the number of successful results.

getAppendFailCount

```
long getAppendFailCount()
```

Returns the number of failures for the Append protocol in the Statement.

Returns the number of failures as a result.

Application Development

JDBC Library Installation Check

Verifies that the machbase.jar file exists in the $MACHBASE_HOME/lib directory.

```
[mach@localhost ~]$ cd $MACHBASE_HOME/lib
[mach@localhost lib]$ ls -l machbase.jar
-rw-rw-r-- 1 mach mach 78599 Jun 18 10:00 machbase.jar
[mach@localhost lib]$
```

Makefile Creation Guide

$(MACHBASE_HOME)/lib/machbase.jar must be specified in the classpath. The following is an example of a Makefile.

```
CLASSPATH=".:$(MACHBASE_HOME)/lib/machbase.jar"

SAMPLE_SRC = Sample1Connect.java Sample2Insert.java Sample3PrepareStmt.java Sample4Append.java

all: build

build:
    -@rm -rf *.class
    javac -classpath $(CLASSPATH) -d . $(SAMPLE_SRC)

create_table:
    machsql -s localhost -u sys -p manager -f createTable.sql

select_table:
    machsql -s localhost -u sys -p manager -f selectTable.sql

run_sample1:
    java -classpath $(CLASSPATH) Sample1Connect

run_sample2:
    java -classpath $(CLASSPATH) Sample2Insert

run_sample3:
    java -classpath $(CLASSPATH) Sample3PrepareStmt

run_sample4:
    java -classpath $(CLASSPATH) Sample4Append

clean:
    rm -rf *.class
```

Compile and Link

Run the make command to compile and link as follows:

```
[mach@localhost jdbc]$ make
javac -classpath ".:/home/machbase/machbase_home/lib/machbase.jar" -d . Sample1Connect.java Sample2Insert.
java Sample3PrepareStmt.java Sample4Append.java
[mach@localhost jdbc]$
```

JDBC Sample

Connection Example

Let's write an example program that connects to a Machbase server using a Machbase JDBC driver. Name the source file Sample1Connect.java.

> The _arrival_time column is not displayed by default. Therefore, to display the _arrival_time column, add show_hidden_cols = 1 to the connection string.
>
> You can modify the connection string in the following example source as follows:
> String sURL = "jdbc:machbase://localhost:5656/mhdb?show_hidden_cols=1";

```
import java.util.*;
import java.sql.*;
import mach.jdbc.driver.*;

public class Sample1Connect
{
    public static Connection connect()
    {
        Connection conn = null;
        try
        {
            String sURL = "jdbc:machbase://localhost:5656/mhdb";

            Properties sProps = new Properties();
            sProps.put("user", "sys");
            sProps.put("password", "manager");

            Class.forName("com.machbase.jdbc.driver");
            conn = DriverManager.getConnection(sURL, sProps);
        }
        catch ( ClassNotFoundException ex )
        {
            System.err.println("Exception : unable to load mach jdbc driver class");
        }
        catch ( Exception e )
        {
            System.err.println("Exception : " + e.getMessage());
        }
        return conn;
    }

    public static void main(String[] args) throws Exception
    {
        Connection conn = null;

        try
        {
            conn = connect();
            if( conn != null )
            {
                System.out.println("mach JDBC connected.");
            }
        }
        catch( Exception e )
        {
            System.err.println("Exception : " + e.getMessage());
        }
        finally
        {
            if( conn != null )
            {
                conn.close();
                conn = null;
            }
        }
    }
}
```

Now compile and run the source code. Use the Makefile you have already created.

```
[mach@localhost jdbc]$ make
javac -classpath ".:/home/machbase/machbase_home/lib/machbase.jar" -d . Sample1Connect.java Sample2Insert.
java Sample3PrepareStmt.java Sample4Append.java
[mach@localhost jdbc]$ make run_sample1
java -classpath ".:/home/machbase/machbase_home/lib/machbase.jar" Sample1Connect
mach JDBC connected.
```

Data Input and Output Example (1) Direct I/O

Create and display an example that uses the Machbase JDBC driver to input and output data.

The name of the source file is called Sample2Insert.java.
First, you need to create the necessary tables using the machsql program.
In the example, we used the sample code to create a table called sample_table in advance.

```
[mach@localhost jdbc]$ machsql
================================================================
     Machbase Client Query Utility
     Release Version 3.5.0.826b8f2.official
     Copyright 2014, Machbase Inc. or its subsidiaries.
     All Rights Reserved.
================================================================
Machbase server address (Default:127.0.0.1):
Machbase rser ID  (Default:SYS)
Machbase user password: MANAGER
MACHBASE_CONNECT_MODE=INET, PORT=5656
mach> create table sample_table(d1 short, d2 integer, d3 long, f1 float, f2 double, name varchar(20), text
text, bin binary, v4 ipv4, v6 ipv6, dt datetime);
Created successfully.
mach> exit
[mach@localhost jdbc]$
```

```
import java.util.*;
import java.sql.*;
import mach.jdbc.driver.*;

public class Sample2Insert
{
    public static Connection connect()
    {
        Connection conn = null;
        try
        {

            String sURL = "jdbc:machbase://localhost:5656/mhdb";

            Properties sProps = new Properties();
            sProps.put("user", "sys");
            sProps.put("password", "manager");

            Class.forName("com.machbase.jdbc.driver");

            conn = DriverManager.getConnection(sURL, sProps);

        }
        catch ( ClassNotFoundException ex )
        {
            System.err.println("Exception : unable to load mach jdbc driver class");
        }
        catch ( Exception e )
        {
            System.err.println("Exception : " + e.getMessage());
        }
```

```
        return conn;
    }


    public static void main(String[] args) throws Exception
    {
        Connection conn = null;
        Statement stmt = null;
        String sql;

        try
        {
            conn = connect();
            if( conn != null )
            {
                System.out.println("mach JDBC connected.");

                stmt = conn.createStatement();

                for(int i=1; i<10; i++)
                {
                    sql = "INSERT INTO SAMPLE_TABLE VALUES (";
                    sql += (i - 5) * 6552;//short
                    sql += ","+ ((i - 5) * 429496728);//integer
                    sql += ","+ ((i - 5) * 922337203685477580L);//long
                    sql += ","+ 1.23456789+"e"+((i<=5)?"":"+")+((i-5)*7);//float
                    sql += ","+ 1.23456789+"e"+((i<=5)?"":"+")+((i-5)*61);//double
                    sql += ",'id-"+i+"'";//varchar
                    sql += ",'name-"+i+"'";//text
                    sql += ",'aabbccddeeff'";//binary
                    sql += ",'192.168.0."+i+"'";//ipv4
                    sql += ",'::192.168.0."+i+"'";
                    sql += ",TO_DATE('2014-08-0"+i+"','YYYY-MM-DD')";//dt
                    sql += ")";

                    stmt.execute(sql);

                    System.out.println( i+" record inserted.");
                }

                String query = "SELECT d1, d2, d3, f1, f2, name, text, bin, to_hex(bin), v4, v6, to_char
(dt,'YYYY-MM-DD') as dt from SAMPLE_TABLE";
                ResultSet rs = stmt.executeQuery(query);
                while( rs.next () )
                {
                    short d1 = rs.getShort("d1");
                    int d2 = rs.getInt("d2");
                    long d3 = rs.getLong("d3");
                    float f1 = rs.getFloat("f1");
                    double f2 = rs.getDouble("f2");
                    String name = rs.getString("name");
                    String text = rs.getString("text");
                    String bin = rs.getString("bin");
                    String hexbin = rs.getString("to_hex(bin)");
                    String v4 = rs.getString("v4");
                    String v6 = rs.getString("v6");
                    String dt = rs.getString("dt");

                    System.out.print("d1: " + d1);
                    System.out.print(", d2: " + d2);
                    System.out.print(", d3: " + d3);
                    System.out.print(", f1: " + f1);
                    System.out.print(", f2: " + f2);
                    System.out.print(", name: " + name);
                    System.out.print(", text: " + text);
                    System.out.print(", bin: " + bin);
                    System.out.print(", hexbin: "+hexbin);
                    System.out.print(", v4: " + v4);
                    System.out.print(", v6: " + v6);
                    System.out.println(", dt: " + dt);
```

```
                }
                rs.close();
            }
        }
        catch( SQLException se )
        {
            System.err.println("SQLException : " + se.getMessage());
        }
        catch( Exception e )
        {
            System.err.println("Exception : " + e.getMessage());
        }
        finally
        {
            if( stmt != null )
            {
                stmt.close();
                stmt = null;
            }
            if( conn != null )
            {
                conn.close();
                conn = null;
            }
        }
    }
}
```

Now compile and run the source code. Use the Makefile you have already created.

```
[mach@localhost jdbc]$ make
javac -classpath ".:/home/machbase/machbase_home/lib/machbase.jar" -d . Sample1Connect.java Sample2Insert.
java Sample3PrepareStmt.java Sample4Append.java
[mach@localhost jdbc]$ make run_sample2
make run_sample2
java -classpath ".:/home/machbase/machbase_home/lib/machbase.jar" Sample2Insert
mach JDBC connected.
1 record inserted.
2 record inserted.
3 record inserted.
4 record inserted.
5 record inserted.
6 record inserted.
7 record inserted.
8 record inserted.
9 record inserted.
d1: 26208, d2: 1717986912, d3: 3689348814741910320, f1: 1.2345679E28, f2: 1.23456789E244, name: id-9, text:
name-9, bin: aabbccddeeff, hexbin: 616162626363646465656666, v4: 192.168.0.9, v6: 0:0:0:0:0:0:c0a8:9, dt:
2014-08-09
d1: 19656, d2: 1288490184, d3: 2767011611056432740, f1: 1.2345678E21, f2: 1.23456789E183, name: id-8, text:
name-8, bin: aabbccddeeff, hexbin: 616162626363646465656666, v4: 192.168.0.8, v6: 0:0:0:0:0:0:c0a8:8, dt:
2014-08-08
d1: 13104, d2: 858993456, d3: 1844674407370955160, f1: 1.23456788E14, f2: 1.23456789E122, name: id-7, text:
name-7, bin: aabbccddeeff, hexbin: 616162626363646465656666, v4: 192.168.0.7, v6: 0:0:0:0:0:0:c0a8:7, dt:
2014-08-07
d1: 6552, d2: 429496728, d3: 922337203685477580, f1: 1.2345679E7, f2: 1.23456789E61, name: id-6, text: name-
6, bin: aabbccddeeff, hexbin: 616162626363646465656666, v4: 192.168.0.6, v6: 0:0:0:0:0:0:c0a8:6, dt: 2014-08-
06
d1: 0, d2: 0, d3: 0, f1: 1.2345679, f2: 1.23456789, name: id-5, text: name-5, bin: aabbccddeeff, hexbin:
616162626363646465656666, v4: 192.168.0.5, v6: 0:0:0:0:0:0:c0a8:5, dt: 2014-08-05
d1: -6552, d2: -429496728, d3: -922337203685477580, f1: 1.2345679E-7, f2: 1.23456789E-61, name: id-4, text:
name-4, bin: aabbccddeeff, hexbin: 616162626363646465656666, v4: 192.168.0.4, v6: 0:0:0:0:0:0:c0a8:4, dt:
2014-08-04
d1: -13104, d2: -858993456, d3: -1844674407370955160, f1: 1.2345679E-14, f2: 1.23456789E-122, name: id-3,
text: name-3, bin: aabbccddeeff, hexbin: 616162626363646465656666, v4: 192.168.0.3, v6: 0:0:0:0:0:0:c0a8:3,
dt: 2014-08-03
d1: -19656, d2: -1288490184, d3: -2767011611056432740, f1: 1.2345679E-21, f2: 1.23456789E-183, name: id-2,
text: name-2, bin: aabbccddeeff, hexbin: 616162626363646465656666, v4: 192.168.0.2, v6: 0:0:0:0:0:0:c0a8:2,
dt: 2014-08-02
d1: -26208, d2: -1717986912, d3: -3689348814741910320, f1: 1.2345679E-28, f2: 1.23456789E-244, name: id-1,
text: name-1, bin: aabbccddeeff, hexbin: 616162626363646465656666, v4: 192.168.0.1, v6: 0:0:0:0:0:0:c0a8:1,
dt: 2014-08-01
```

Data Input and Output Example (2) PreparedStatement Input Used

Create and view an example that uses a PreparedStatement to input and output data.

The name of the source file is Sample3PrepareStmt.java.

```
import java.util.*;
import java.sql.*;
import java.text.SimpleDateFormat;
import mach.jdbc.driver.*;

public class Sample3PrepareStmt
{
    public static Connection connect()
    {
        Connection conn = null;
        try
        {
            String sURL = "jdbc:machbase://localhost:5656/mhdb";

            Properties sProps = new Properties();
            sProps.put("user", "sys");
            sProps.put("password", "manager");
```

```
            Class.forName("com.machbase.jdbc.driver");

            conn = DriverManager.getConnection(sURL, sProps);

        }
        catch ( ClassNotFoundException ex )
        {
            System.err.println("Exception : unable to load mach jdbc driver class");
        }
        catch ( Exception e )
        {
            System.err.println("Exception : " + e.getMessage());
        }
        return conn;
    }

    public static void main(String[] args) throws Exception
    {
        Connection conn = null;
        Statement stmt = null;
        machPreparedStatement preStmt = null;
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss SSS");

        try
        {
            conn = connect();
            if( conn != null )
            {
                System.out.println("mach JDBC connected.");

                stmt = conn.createStatement();
                preStmt = (machPreparedStatement)conn.prepareStatement("INSERT INTO SAMPLE_TABLE VALUES(?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?)");

                String ipStr = null;
                String dateStr = null;
                for(int i=1; i<10; i++)
                {
                    ipStr = String.format("172.16.0.%d",i);
                    dateStr = String.format("2014-08-09 12:23:34 %03d", i);
                    byte[] bin = new byte[20];
                    for(int j=0;j<20;j++){
                        bin[j]=(byte)(Math.random()*255);
                    }
                    java.util.Date day = sdf.parse(dateStr);
                    java.sql.Date sqlDate = new java.sql.Date(day.getTime());

                    preStmt.setShort(1, (i-5) * 3276 );
                    preStmt.setInt(2, (i-5) * 214748364 );
                    preStmt.setLong(3, (i-5) * 922337203685477580L );
                    preStmt.setFloat(4, 1.23456789101112131415*Math.pow(10,i));
                    preStmt.setDouble(5, 1.23456789101112131415*Math.pow(10,i*10));
                    preStmt.setString(6, String.format("varchar-%d",i));
                    preStmt.setString(7, String.format("text-%d",i));
                    preStmt.setBytes(8, bin);
                    preStmt.setIpv4(9, ipStr);
                    preStmt.setIpv6(10, "::"+ipStr);
                    preStmt.setDate(11, sqlDate);
                    preStmt.executeUpdate();

                    System.out.println( i+" record inserted.");
                }

                //date type format : YYYY-MM-DD HH24:MI:SS mmm:uuu:nnnn
                String query = "SELECT d1, d2, d3, f1, f2, name, text, bin, to_hex(bin), v4, v6, to_char
(dt,'YYYY-MM-DD HH24:MI:SS mmm:uuu:nnn') as dt from SAMPLE_TABLE";
                ResultSet rs = stmt.executeQuery(query);
                while( rs.next () )
                {
                    short d1 = rs.getShort("d1");
```

```
                    int d2 = rs.getInt("d2");
                    long d3 = rs.getLong("d3");
                    float f1 = rs.getFloat("f1");
                    double f2 = rs.getDouble("f2");
                    String name = rs.getString("name");
                    String text = rs.getString("text");
                    String bin = rs.getString("bin");
                    String hexbin = rs.getString("to_hex(bin)");
                    String v4 = rs.getString("v4");
                    String v6 = rs.getString("v6");
                    String dt = rs.getString("dt");

                    System.out.print("d1: " + d1);
                    System.out.print(", d2: " + d2);
                    System.out.print(", d3: " + d3);
                    System.out.print(", f1: " + f1);
                    System.out.print(", f2: " + f2);
                    System.out.print(", name: " + name);
                    System.out.print(", text: " + text);
                    System.out.print(", bin: " + bin);
                    System.out.print(", hexbin: "+hexbin);
                    System.out.print(", v4: " + v4);
                    System.out.print(", v6: " + v6);
                    System.out.println(", dt: " + dt);
                }
                rs.close();
            }
        }
        catch( SQLException se )
        {
            System.err.println("SQLException : " + se.getMessage());
        }
        catch( Exception e )
        {
            System.err.println("Exception : " + e.getMessage());
        }
        finally
        {
            if( stmt != null )
            {
                stmt.close();
                stmt = null;
            }
            if( conn != null )
            {
                conn.close();
                conn = null;
            }
        }
    }
}
```

Now compile and run the source code. Use the Makefile you have already created.

It should be noted that the data entered in Sample2Insert.java is output together.

```
[mach@localhost jdbc]$ make
javac -classpath ".:/home/machbase/machbase_home/lib/machbase.jar" -d . Sample1Connect.java
Sample2Insert.java Sample3PrepareStmt.java Sample4Append.java
[mach@localhost jdbc]$ make run_sample3
make run_sample3
java -classpath ".:/home/machbase/machbase_home/lib/machbase.jar" Sample3PrepareStmt
Mach JDBC connected.
1 record inserted.
2 record inserted.
3 record inserted.
4 record inserted.
5 record inserted.
6 record inserted.
7 record inserted.
```

```
8 record inserted.
9 record inserted.
d1: 13104, d2: 858993456, d3: 3689348814741910320, f1: 754454.6, f2: 453821.380752063, name:
varchar-9, text: text-9, bin: ?+,??r?J?????S)n?, hexbin:
A4C9A8D491D6728B4AACB39EE5FC5300296EFA9F, v4: 172.16.0.9, v6: 0:0:0:0:0:0:ac10:9, dt:
2014-08-09 12:23:34 009:000:000
?h???a?, hexbin: 6C20F09329ABBA3E7DE501C30DA368D6EFC961EF, v4: 172.16.0.8, v6:
0:0:0:0:0:0:ac10:8, dt: 2014-08-09 12:23:34 008:000:000
d1: 6552, d2: 429496728, d3: 1844674407370955160, f1: 2664182.0, f2: 1357910.1926900472, name:
varchar-7, text: text-7, bin: ????Uls?q?H?I?&(?, hexbin:
B5A0A2EFA185556C73BF719448BD49C92628F8C6, v4: 172.16.0.7, v6: 0:0:0:0:0:0:ac10:7, dt:
2014-08-09 12:23:34 007:000:000
d1: 3276, d2: 214748364, d3: 922337203685477580, f1: 443847.1, f2: 9342855.256576871, name:
varchar-6, text: text-6, bin: ??>x??Eu?? ?Iw??+n, hexbin:
BC973E78F5B44575D6CC15F94977DAE62B6E1D0E, v4: 172.16.0.6, v6: 0:0:0:0:0:0:ac10:6, dt:
2014-08-09 12:23:34 006:000:000
d1: 0, d2: 0, d3: 0, f1: 1283723.1, f2: 1771261.2019240903, name: varchar-5, text: text-5,
bin: &== j?j3?? T??y?
??, hexbin: 263D3D1C6AF56A33F79D0C54A5C479A4030AFE8B, v4: 172.16.0.5, v6: 0:0:0:0:0:0:ac10:5,
dt: 2014-08-09 12:23:34 005:000:000
d1: -3276, d2: -214748364, d3: -922337203685477580, f1: 9447498.0, f2: 7529392.937964935,
name: varchar-4, text: text-4, bin: ?Sw ??)? ?h2?E??/?, hexbin:
C653771DD2DF29CDB30ED96832E745D3D7A52FD2, v4: 172.16.0.4, v6: 0:0:0:0:0:0:ac10:4, dt:
2014-08-09 12:23:34 004:000:000
d1: -6552, d2: -429496728, d3: -1844674407370955160, f1: 9589634.0, f2: 5994172.201347323,
name: varchar-3, text: text-3, bin: 9aB,.????L/?=3,?`?f, hexbin:
3961422C2EA39BE6F2964C2FCD3D332C8960A466, v4: 172.16.0.3, v6: 0:0:0:0:0:0:ac10:3, dt:
2014-08-09 12:23:34 003:000:000
d1: -9828, d2: -644245092, d3: -2767011611056432740, f1: 7409537.5, f2: 2313739.6613546023,
name: varchar-2, text: text-2, bin: _? N?3 ?? ??~H ??= 8, hexbin:
5F84144EF63320F3C718B0FD7E4809A4CB3D1838, v4: 172.16.0.2, v6: 0:0:0:0:0:0:ac10:2, dt:
2014-08-09 12:23:34 002:000:000
d1: -13104, d2: -858993456, d3: -3689348814741910320, f1: 596626.75, f2: 2649492.1936065694,
name: varchar-1, text: text-1, bin: ???d??Wu$v? 7m?-, hexbin:
E8D0C564B4EB57E59B08752476FC07376DBF2D14, v4: 172.16.0.1, v6: 0:0:0:0:0:0:ac10:1, dt:
2014-08-09 12:23:34 001:000:000
d1: 26208, d2: 1717986912, d3: 3689348814741910320, f1: 1.2345679E28, f2: 1.23456789E244,
name: id-9, text: name-9, bin: aabbccddeeff, hexbin: 616162626363646465656666, v4:
192.168.0.9, v6: 0:0:0:0:0:0:c0a8:9, dt: 2014-08-09 00:00:00 000:000:000
d1: 19656, d2: 1288490184, d3: 2767011611056432740, f1: 1.2345678E21, f2: 1.23456789E183,
name: id-8, text: name-8, bin: aabbccddeeff, hexbin: 616162626363646465656666, v4:
192.168.0.8, v6: 0:0:0:0:0:0:c0a8:8, dt: 2014-08-08 00:00:00 000:000:000
d1: 13104, d2: 858993456, d3: 1844674407370955160, f1: 1.23456788E14, f2: 1.23456789E122,
name: id-7, text: name-7, bin: aabbccddeeff, hexbin: 616162626363646465656666, v4:
192.168.0.7, v6: 0:0:0:0:0:0:c0a8:7, dt: 2014-08-07 00:00:00 000:000:000
d1: 6552, d2: 429496728, d3: 922337203685477580, f1: 1.2345679E7, f2: 1.23456789E61, name:
id-6, text: name-6, bin: aabbccddeeff, hexbin: 616162626363646465656666, v4: 192.168.0.6, v6:
0:0:0:0:0:0:c0a8:6, dt: 2014-08-06 00:00:00 000:000:000
d1: 0, d2: 0, d3: 0, f1: 1.2345679, f2: 1.23456789, name: id-5, text: name-5, bin:
aabbccddeeff, hexbin: 616162626363646465656666, v4: 192.168.0.5, v6: 0:0:0:0:0:0:c0a8:5, dt:
2014-08-05 00:00:00 000:000:000
d1: -6552, d2: -429496728, d3: -922337203685477580, f1: 1.2345679E-7, f2: 1.23456789E-61,
name: id-4, text: name-4, bin: aabbccddeeff, hexbin: 616162626363646465656666, v4:
192.168.0.4, v6: 0:0:0:0:0:0:c0a8:4, dt: 2014-08-04 00:00:00 000:000:000
d1: -13104, d2: -858993456, d3: -1844674407370955160, f1: 1.2345679E-14, f2: 1.23456789E-122,
name: id-3, text: name-3, bin: aabbccddeeff, hexbin: 616162626363646465656666, v4:
192.168.0.3, v6: 0:0:0:0:0:0:c0a8:3, dt: 2014-08-03 00:00:00 000:000:000
d1: -19656, d2: -1288490184, d3: -2767011611056432740, f1: 1.2345679E-21, f2: 1.23456789E-183,
name: id-2, text: name-2, bin: aabbccddeeff, hexbin: 616162626363646465656666, v4:
192.168.0.2, v6: 0:0:0:0:0:0:c0a8:2, dt: 2014-08-02 00:00:00 000:000:000
d1: -26208, d2: -1717986912, d3: -3689348814741910320, f1: 1.2345679E-28, f2: 1.23456789E-244,
name: id-1, text: name-1, bin: aabbccddeeff, hexbin: 616162626363646465656666, v4:
192.168.0.1, v6: 0:0:0:0:0:0:c0a8:1, dt: 2014-08-01 00:00:00 000:000:000
```

Extension Function Append Example

The Machbase JDBC driver supports the Append protocol to quickly upload large numbers of data.

The following is an example of using the Append protocol.
Use the sample_table used in the previous example.
The name of the source file is called Sample4Append.java.
Enter the contents of data.txt into sample_table.
Copy the data.txt file used in the CLI append example.

```
import java.util.*;
import java.sql.*;
import java.io.*;
import java.text.SimpleDateFormat;
import java.math.BigDecimal;
import mach.jdbc.driver.*;


public class Sample4Append
{
    protected static final String sTableName = "sample_table";
    protected static final int sErrorCheckCount = 100;

    public static Connection connect()
    {
        Connection conn = null;
        try
        {
            String sURL = "jdbc:machbase://localhost:5656/mhdb";

            Properties sProps = new Properties();
            sProps.put("user", "sys");
            sProps.put("password", "manager");

            Class.forName("com.machbase.jdbc.driver");

            conn = DriverManager.getConnection(sURL, sProps);

        }
        catch ( ClassNotFoundException ex )
        {
            System.err.println("Exception : unable to load mach jdbc driver class");
        }
        catch ( Exception e )
        {
            System.err.println("Exception : " + e.getMessage());
        }
        return conn;
    }

    public static void main(String[] args) throws Exception
    {
        Connection conn = null;
        MachStatement stmt = null;
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
        Calendar cal = Calendar.getInstance();
        String filename = "data.txt";

        try
        {
            conn = connect();
            if( conn != null )
            {
                System.out.println("Mach JDBC connected.");

                stmt = (MachStatement)conn.createStatement();

                ResultSet rs = stmt.executeAppendOpen(sTableName, sErrorCheckCount);
                ResultSetMetaData rsmd = rs.getMetaData();

                System.out.println("append open ok");

                MachAppendCallback cb = new MachAppendCallback() {
                        @Override
```

```
                    public void onAppendError(long aErrNo, String aErrMsg, String aRowMsg) {
                         System.out.format("Append Error : [%05d - %s]\n%s\n", aErrNo, aErrMsg, aRowMsg);
                    }
                };

            stmt.executeSetAppendErrorCallback(cb);

            System.out.println("append data start");
            BufferedReader in = new BufferedReader(new FileReader(filename));
            String buf = null;
            int cnt = 0;
            long dt;

            long startTime = System.nanoTime();

            while( (buf = in.readLine()) != null )
            {
                ArrayList<Object> sBuf = new ArrayList<Object>();
                StringTokenizer st = new StringTokenizer(buf,",");
                for(int i=0; st.hasMoreTokens() ;i++ )
                {
                    switch(i){
                        case 7://binary case
                            sBuf.add(new ByteArrayInputStream(st.nextToken().getBytes())); break;
                        case 10://date case
                            java.util.Date day = sdf.parse(st.nextToken());
                            cal.setTime(day);
                            dt = cal.getTimeInMillis()*1000000; //make nanotime
                            sBuf.add(dt);
                            break;
                        default:
                            sBuf.add(st.nextToken()); break;
                    }
                }

                if( stmt.executeAppendData(rsmd, sBuf) != 1 )
                {
                    System.err.println("Error : AppendData error");
                    break;
                }

                if( (cnt++%10000) == 0 )
                {
                    System.out.print(".");
                }
                sBuf = null;

            }
            System.out.println("\nappend data end");

            long endTime = System.nanoTime();
            stmt.executeAppendClose();
            System.out.println("append close ok");
            System.out.println("Append Result : success = "+stmt.getAppendSuccessCount()+", failure =
"+stmt.getAppendFailureCount());
            System.out.println("timegap " + ((endTime - startTime)/1000) + " in microseconds, " + cnt +
" records" );

            try {
                BigDecimal records = new BigDecimal( cnt );
                BigDecimal gap = new BigDecimal( (double)(endTime - startTime)/1000000000 );
                BigDecimal rps = records.divide(gap, 2, BigDecimal.ROUND_UP );

                System.out.println( rps + " records/second" );
            } catch(ArithmeticException ae) {
                System.out.println( cnt + " records/second");
            }

            rs.close();
        }
    }
```

```
        catch( SQLException se )
        {
            System.err.println("SQLException : " + se.getMessage());
        }
        catch( Exception e )
        {
            System.err.println("Exception : " + e.getMessage());
        }
        finally
        {
            if( stmt != null )
            {
                stmt.close();
                stmt = null;
            }
            if( conn != null )
            {
                conn.close();
                conn = null;
            }
        }
    }
}
```

When appending, date type data must be converted to long type nanosecond time.

```
[mach@localhost jdbc]$ make run_sample4
make run_sample4
java -classpath ".:/home/machbase/machbase_home/lib/machbase.jar" Sample4Append;
Mach JDBC connected.
append open ok
append data start
......
append data end
append close ok
Append Result : success = 60000, failure = 0
timegap 6905594 in microseconds, 60000 records
8688.61 records/second
```

Displays the dot (.) every 10,000, and can know the input time.

```
# Use machsql to check number actually entered.
# Confirm that 60018 are entered including those from Sample2Insert and Sample3PrepareStmt.


[mach@localhost jdbc]$ machsql
=================================================================
     Machbase Client Query Utility
     Release Version 3.0.0
     Copyright 2014, Machbase Inc. or its subsidiaries.
     All Rights Reserved.
=================================================================
Machbase server address (Default:127.0.0.1):
Machbase user ID  (Default:SYS)
Machbase user password: MANAGER
MACH_CONNECT_MODE=INET, PORT=5656
mach> select count(*) from sample_table;
count(*)
----------------------
60018
[1] row(s) selected.
```