TECHNICAL COMPUTING USING SYBASE DATABASE FOR BIOMEDICAL SIGNAL ANALYSIS

J. Krupa, A. Procházka, V. Hanta, R. Háva

Institute of Chemical Technology, Prague
Department of Computing and Control Engineering

Abstract

Design a storage for a large amount of biomedical data in a structured way is a hard task. Especially if there is a requirement for realtime access to all of the stored data. For this purpose we decided to use Sybase Adaptive Server Enterprise as a main database engine and MathWorks MATLAB for mathematical data manipulation. Our database consists of approximately 40000 patient EEG anamneses. Each of them is around 120000 records in size.

Many programming languages contain native libraries for Sybase database access. We tried the implementation in *Python* and *PHP* programming languages. For the demonstration there is an example of distributed computing platform based on the database access from all of the application components - Python worker and data dumper, PHP web interface and MATLAB data manipulation script.

1 EEG Database Structure

Because of the large quantity of data which we needed to store into database we had to design the right database architecture to be able to use the database effectively. We worked with two possible database schemas.

One of the schemas consist of three tables. The main table with all of the records from all patients with foreign key for patient identification, the other two for list of patients and measures. Because of the massive amount of rows needed for this solution and also because there wasn't demand to use just part of the data for one patient we decided not to use this schema for production use. We although used this schema for simple comparison on how to add large amount of data into the database.

The second schema consists of two tables - one for list of patients, one for list of patient anamneses and binary blob as a storage for the 120000 * 19 records. It's not possible to select parts of the patients anamneses on database level but it is much faster in terms of adding and later access to the data in the database. All recordings are stored in both filtered variation with 50Hz frequency component removed [6] as well as original data without any filters applied. Because the binary MAT file format has publicly available specification [2] we decided to use it for the binary storage of the data in the database.

2 MATLAB Database Toolbox

MATLAB Database Toolbox [1] supports communication using ODBC or JDBC driver with compatible database including IBM DB2, IBM Informix, Ingres, Microsoft Access, Microsoft Excel, Microsoft SQL Server, MySQL, Oracle, PostgreSQL, Sybase SQL Anywhere and Sybase SQL Server.

A JDBC driver is a software component enabling a Java application to communicate with a database. To connect with each databases type JDBC requires drivers. The JDBC driver provides the connection to the database and implements the protocol for transferring the communication between client and database.

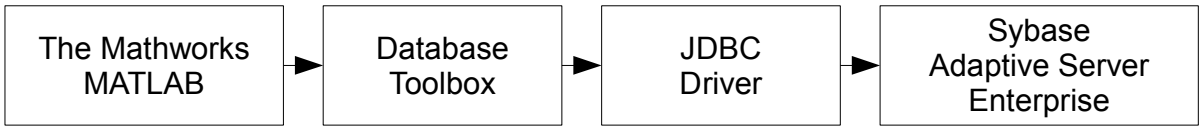


Figure 1: MATLAB - Sybase communication schema

There are two JDBC drivers available for use with Sybase database. jConnect is provided by Sybase for free. jTDS driver is capable of connecting to Sybase database as well as Microsoft SQL Server. It is open-source product available for free.

Listing 1: Database toolbox usage

```
% Database settings
db_host = 'hostname';
db_name = 'eeg';
db_user = 'user';
db_pass = 'password';

% JDBC Parameters
jdbcString = sprintf('jdbc:jtds:sybase://%s/%s', db_host, db_name);
jdbcDriver = 'net.sourceforge.jtds.jdbc.Driver';
%javaaddpath('jtds-1.2.2.jar')

% Create the database connection object
dbConn = database(db_name, db_user , db_pass, jdbcDriver, jdbcString);

% Check to make sure that we successfully connected
if isconnection(dbConn)

        % Create the database query
        db_query = "";

        % Submit database query to the database
        disp(db_query)
        result = get(fetch(exec(dbConn, db_query)), 'Data');

        % If there is some problem with the database query, display it
        % on the screen
        if (result ~= 0)
                disp(result);
        end

% If the connection failed, print the error message
else
    disp(sprintf('Connection failed: %s', dbConn.Message));
end

% Close the connection so we don't run out of MySQL threads
close(dbConn);
```

3 Inserting Data Into Database

3.1 INSERT statement

One of the simplest methods to put data into database using MATLAB and MATLAB Database Toolbox is by INSERT statement as showed on Listing (2). This is probably the easiest and fastest way if there is a need to add small quantity of data. It becomes slow for large amount of records.

Listing 2: Simple INSERT statement

```
db_query = "INSERT INTO test (a1, a2) VALUES (1, 2)";
result = get(fetch(exec(dbConn, db_query)), 'Data');
```

3.2 MATLAB fastinsert function

There is fastinsert function available in MATLAB which should be used for insertion of larger amount of data to the database. The example can be seen on Listing (3).

Listing 3: fastinsert function example

```
fastinsert(dbConn, 'eeg', namecols, exdata);
```

3.3 Using Sybase tools

All major databases on the market includes various set of command line administration tools for mass import of data. Sybase Adaptive Server Enterprise [5] comes with bcp tool which can be used for this purpose. It takes comma-separated values (CSV) file as an input and put it into the database. The main advantage of this method is its speed.

In order to export large matrix from MATLAB into CSV file format we can use csvwrite or dlmwrite functions as shown on Listing (4).

Listing 4: MATLAB dlmwrite function example

```
dlmwrite('output1.csv', data, 'precision', 6);
```

There are several database configuration options which can be used to speed-up the whole process. Disabling the transaction log is advised as well as enabling the "select into/bulkcopy-/pllsort" option in database properties.

As we can see from the Table (1) in terms of speed the best way to import data into database is to export them from MATLAB to CSV file and use bcp tool provided with Sybase Adaptive Server Enterprise.

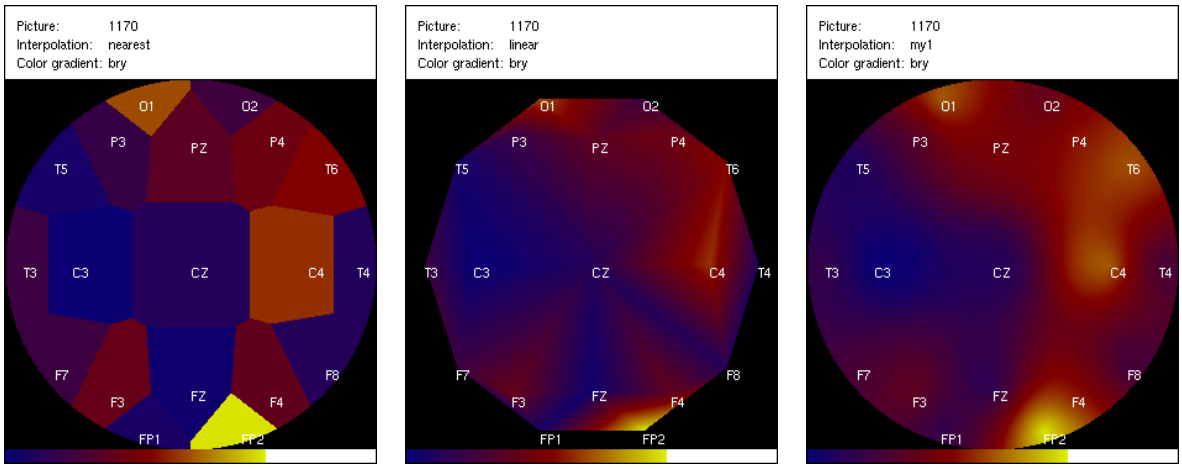
Table 1: Comparison of the time needed to insert records into database

Records	*Time [s]*	
	ASE dbo	*ASE bcp*
1000	13	0.5

4 Mathematical Application

One of the major goals of the database storage deployment was the ability to deliver data to various platforms using different programming languages and operating systems. We achieved that by using standardized communication interfaces provided by JDBC driver as well as language-specific database connectors for Sybase Adaptive Server Enterprise.

Database as a core structure part is used in one of the distributed computing engines we use to process EEG data [3]. It is a hybrid product created on our department using MATLAB, Python [4] and PHP programming languages. One of the applications is distributed computing of interpolated EEG images as we can see on Fig. (2).



(a) Nearest neighbor (b) Linear (c) Weighted distance

Figure 2: Output from distributed computing engine in Python - Different interpolation methods

Acknowledgements

The work has been supported by the research grant of the Faculty of Chemical Engineering of the Institute of Chemical Technology, Prague No. MSM 6046137306.

References

[1] The MathWorks Inc. Matlab documentation. 1984-2009. Available from World Wide Web: `http://www.mathworks.com/access/helpdesk/help/techdoc/`.

[2] The MathWorks Inc. Mat-file format. 1999.

[3] J. Krupa. *Three Dimensional Modelling of EEG Signals.* VŠCHT Praha, 2009.

[4] M. Lutz. *Learning Python, 3rd Edition.* O'Reilly, 2008.

[5] Sybase. *Adaptive Server Enterprise 15.0.3.* 2009.

[6] S. V. Vaseghi. *Advanced Digital Signal Processing and Noise Reduction.* John Wiley & Sons Ltd, West Sussex, 2006.