Technical report, June 2010

EVALUATION OF DATABASE MANAGEMENT SYSTEMS

Bachelor's Thesis in Computer Engineering

Xing Liang

Yongyu Lu

School of Information Science, Computer and Electrical Engineering, Halmstad University

School of Information Science, Computer and Electrical Engineering
Halmstad University
Box 823, S-301 18 Halmstad, Sweden

January 2010

DETAILS

First Name, Surname: Xing Liang, Yongyu Lu

University: Halmstad University, Sweden

Degree Program: Computer Engineering

Title of Thesis: Evaluation of Database Management Systems

Academic Supervisor: Wagner Ourique de Morais

ABSTRACT

Qualitative and quantitative analysis of different database management systems (DBMS) have been performed in order to identify and compare those which address requirements such as public domain licensing, free of charge, high product support, ADO .NET Entity Framework compatibility, good performance, referential integrity, among others. More than 20 existing database management systems have been selected as possible candidates. Qualitative analysis reduced that number to 4 candidates DBMSs (PostgreSQL, SQLite, Firebird and MySQL). Quantitative analysis has been used to test the performance of these 4 DBMSs while performing the most common structured query language (SQL) data manipulation statements (INSERT, UPDATE, DELETE and SELECT). Referential integrity and easy to install were also evaluated for these 4 DBMSs. As results, Firebird is the most suitable DBMS which best addressed all desired requirements.

ACKNOWLEDGEMENT

The project, evaluation of database management systems, has taken about four months to finish. It is hard work. During doing project, we want to thank many people for their helps. First we thank two people, Tobias Persson and Michal Lysek, who are from the Medicwave Company. They are always friendly and ready to help us. When we had troubles in programming, they gave us important advice and taught us a lot. We also thank our supervisor, Wagner Ourique De Morais. He helped us find the right direction to finish the project. He also gave us some material of the thesis. He is strict and kind. We can't finish the thesis without him. At last, we want to thank our family and friends. They are always cared about us and give us much support.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

DBMS: Database Management System

MBS: Medicwave Bioinformatics Suite

ODBC: Open Database Connectivity

OLAP: Online Analytical Process

SQL: Structured Query Language

1. INTRODUCTION

1.1. BACKGROUND

The idea for this thesis project comes from a local software company called Medicwave [1]. The company provides bioinformatics tools for high dimensional mass spectrometry data analysis and is one of the leading bioinformatics solution providers in Europe.

The Medicwave Bioinformatics Suite™ (MBS™) is one example of these solutions which contains a set of functionalities for mass spectrometry data analysis. According to the company, the MBS software application provides high quality data analysis together with high flexibility for special requirements, what leads to reduced time for each analysis. Currently, the MBS is a Microsoft Windows-based solution and uses SQLite [2] as database.

The company started to use SQLite in 2005 when they needed a small, simple, easy to use database that didn't cost them anything. In the MBS application, users can run data analysis and generate a result, which is saved in the database. If the user runs another the analysis using the first result as input, they will get another result as output which is also stored in the database. A relational database system such as SQLite allows the system to keep track of the data analysis and achieved results.

Nowadays, the MBS software application has to deal with and store huge amounts of data and the company wants to verify if the SQLite is still a good choice or if another database management system should be adopted in newer versions for the MBS.

Thus, the main goal of this project is to evaluate different database management systems and recommend the most suitable one according to different criteria pointed out by the company.

1.2. PROJECT OBJECTIVE

The MBS application stores large quantities of data, several gigabytes, in the database. As the company is not sure if current DBMS, SQLite, can handle large amounts of data while providing good performance, the company wants to see if alternative DBMSs could be used instead.

The main objective of this project is to compare and evaluate different DBMSs and recommend the best option for the MBS application. In order to do that, distinct company requirements are addressed.

1.3. REQUIREMENTS

The company has specified distinct criteria that must be addressed while evaluating DBMS suitable for the MBS application as follows:

a. Licensing [3].

b. Performance

c. Market Position and support

d. Easy to use

e. Support the .Net Entity Framework

f. Cost

According to the company, due to the system architecture and how it is used, security, replication and "open-source software" [4]are unimportant requirements.

1.4. OUTLINE

The report is organized as follows:

- Chapter 2 presents the methods used to reduce the set of DBMSs to be compared as well as quantitative and qualitative analysis used to evaluate them
- Chapter 3 presents the measurements according to the qualitative criteria
- Chapter 4 presents the DBMSs evaluation according to the measurements and indicated the most suitable database for future versions of the MBS application.
- Chapter 5 presents the project conclusion

1.5. RELATED WORK

The aim of the project is evaluating the DBMS. But some experiments of evaluation from the documentations have been done by other people. These tables show the comparison and features among SQLite, MySQL [5], Firebrid [6], PostgreSQL [7] and other DBMSs.

Test	SQLite 3.3.3 (seconds)	MySQL 5.0.18 (seconds)	Firebird 1.5.2 (seconds)	PostgreSQL 8.1.2 (seconds)
1000 INSERTs	3.823	2.647	0.320	4.922
25000 INSERTs into an index table	1.778	11.524	6.351	19.236

100 SELECTs without an index	3.153	2.718	2.976	5.740
5000 SELECTs with an index	1.872	3.763	5.187	199.823
DELETE without an index	0.528	0.394	0.404	0.336

Table 1: Performance results for distinct test cases in 4 DMBSs [8]

A comparison among different DBMSs is presented in Table 1. According to the results, Firebird and SQLite have better performance while inserting data than others. Considering a small number of INSERT operations, Firebird has better performance than other DBMSs which only takes less than 0.5 seconds. In this test, each SQL statement is a separate transaction. The database files must be opened and closed and the cache must be flushed 1000 times leads. Firebird has advantage under this circumstance. With a huge number of INSERT operations, SQLite is the fastest, which is 19 times than PostgreSQL. SQLite's speed advantage presents for the test.

The performance while performing few SELECT operations is fairly similar among the DBMSs, except for PostgreSQL, which has taken almost twice than others. Considering a huge number of SELECT operations, SQLite is the fastest, being 200 times faster than PostgresSQL, which is the slowest. PostgreSQL performs better than others while deleting data. In summary, SQLite performs better while handling data in indexed tables while Firebird provides better results for non-indexed tables.

2. METHOD

The evaluation has been divided into 2 phases. During the first phase, a preliminary analysis has been done by using the information found at the DBMSs documentation. This first phase reduced the number of DBMSs to be investigated, due to the fact that many of them do not address requirements in terms of licensing, cost and .NET Entity Frame support. During the second phase, the results come out from performance evaluation of the selected DBMSs. The evaluation consists of measuring the performance of the he most common SQL data manipulation statements (INSERT, UPDATE, DELETE and SELECT). The referential integrity is also evaluated. All tests are under the same benchmark condition, i.e., same hardware, amount of data and number of operations.

2.1. QUALITATIVE ANALYSIS

Qualitative analysis is an assessment of analysis which contains some methods, such as theoretical investigating or others. In the project, the qualitative analysis is used to reduce the amount of DBMSs which don't address the requirements according to the documentation. The qualitative analysis focuses on collecting the data and conclusion from the related study and documentation in order to select the DBMSs.

2.1.1. QUALITATIVE CRITERIA

According to the requirements of the company, the quantitative criteria consist of license, cost, market position and support and .net entity frame support.

Database	License	Cost (SEK)	.NET Entity Frame support
Open source			
Apache Derby	Apache License	Free	Yes
CUBRID	BSD, GPL v2	Free	Yes
Firebird	IPL and IDPL	Free	Yes
HSQLDB	BSD	Free	No
H2	EPL and modified MPL	Free	Yes
MonetDB	MonetDB Public License v1.1	Free	Yes
PostgreSQL	PostgreSQL	Free	Yes

	licence		
SmallSQL	LGPL	Free	No
SQLite	Public domain	Free	Yes
MySQL	GPL or Proprietary	4.322	Yes
Proprietary			
DB2	Proprietary	40.000	Yes
FileMaker	Proprietary	7.000	Yes
FrontBase	Proprietary	35,000	Yes
Informix Dynamic Server	Proprietary	2.000	Yes
Ingres	GPL and Proprietary	61.240	Yes
Microsoft SQL Server	Proprietary	18.250	Yes
Oracle	Proprietary	1.530	Yes
ScimoreDB	Proprietary	4.396	Yes
SQLBase	Proprietary	1.253	Yes
Teradata	Proprietary	23.600	Yes
Sybase Advantage Database Server	Proprietary	14.000	Yes

Table 2: The different DBMSs be considered

2.1.2. TESTING CANDIDATES

2.1.2.1. LICENSE

The license protects the patent retaliation and carry requirements and restrictions to the distributors. The company doesn't want to open their application source code if they use new DMBS. The license can't force the company to release the application code. If the license would open their source code, it couldn't be selected.

2.1.2.2. Market position and support

The DBMS should have a good market position and support. The DBMS should be fairly common, actively developed or backed by a large company. The DBMS should be supported in terms of technical support and after-sale service. The company

doesn’t want to use a DBMS which won’t be supported or no longer available quickly in the future.

2.1.2.3. .NET ENTITY FRAMEWORK SUPPORT

The company needs to access the database from C++ as well as from C# using the Entity Framework. Mirocsoft .NET Entity Framework is a software framework which supports multiple programming languages and includes library of coded solutions to common programming problems. The application is programmed based on .NET Entity framework. The company is using Entity Framework for the database connection in C#. The connecting string is only the part which the company wants to modify. The company doesn’t want to change their C# code if they switch SQLite to another DBMS. If the DBMS support .NET Entity Framework support, they can switch without changing the code except the connection string.

2.1.2.4. COST

Some DMBSs have price while some are free. The company doesn’t want to pay for the DBMS, including the technical support or after-sale service, if the cost is more than 10000SEK. So the DBMS can’t be too expensive.

2.1.2.5. THE TESTING DBMS CANDIDATES

	License	**Market position and support**	**Cost**	**.NET Entity Frame support**
MySQL	GPL or Proprietary	Google Adwords, NASA companies etc. use	4322	Yes
Firebird	IPL and IDPL	Popular with middle and small companies	Free	Yes
SQLite	Public domain	Firefox, Apple companies etc. use	Free	Yes
PostgreSQL	PostgreSQL licence	Afilias, Royal companies etc. Use	Free	Yes

Table 3: The DBMSs be selected to test

2.2. QUANTITATIVE ANALISYS

Quantitative analysis is a method of gathering the data which researchers need using some experiments or survey. In the project, quantitative analysis provides the results from a numerical perspective. The quantitative analysis measures the

performances which contains INSERT, SELECT, DELETE and referential integrity.

2.2.1. BENCHMARKS

The benchmark is a set of programs to assess related performance of testing. It provides related standard tests and trials. In the project, the goal of the benchmark is to test the performance of DBMSs and get the results in order to show which DBMS is most suitable for the application. The MBS Application needs a single user database which doesn't need to be tested advanced features of modern DBMS, such as: OLAP, cluster and so on. So the tests put the main focus on the speed of reading from and writing data from/into the database. The tests make insert and update queries with different data size to compare the operating time of several DBMSs. The benchmark includes hardware conditions and program conditions.

Do the benchmark test base on the following conditions:

a. Hardware configuration
    a) Processor: Intel(R) Core(TM)2 Duo CPU T7100 @ 1.80GHz
    b) Installed memory (RAM): 2.00GB
b. Software configuration
    a) Operating System: Windows XP
    b) DBMSs:
        i. SQLite 3.6.22
        ii. Mysql 5.1.43
        iii. PostgreSQL 8.4.2
        iv. Firebird V2.1.3
c. Create the databases
    a) A database has been created in each DBMS according to the database model PROVIDED BY THE COMPANY.
d. Populate tables
    a) Ten Tables were been populated with default data.
    b) Insert 3GB data into ten tables. Two tables, ResultData and Analysis, contain most data because of the company's requests. The others contain remaining. No need to evaluate the speed because the MBS has 3GB data basically and use them as analysis. The end users only input their data and get the analysis base on the 3GB data and their data.
e. Performance evaluation
    a) Insert operations
        i. 1GB of data
        ii. Insert operations with different size of each row (50K, 500K, 1M) into tables of different databases. Log the time taken. This is the way of inserting data into one table.

Use loop to design the data to ensure the size of each row which is fixed, for

example 50K, 500K or 1M.

Get the time before insert operation.

Use loop to insert the data designed before into table until the size of data is 1G.

Get the time after insert operation.

The difference of the two is the duration that how much it takes to do the insert operation.

b) Select operations

i. 1MB, 100MB, 400MB and 700MB of Data

ii. Select operations data with different size of each row (50K, 500K, 1M) into tables of different databases. Log the time taken. This is the way of selecting data into tables.

Get the time before select operation.

Use loop to select the data from the table until the size of data is the value what is needed, for example 1M, 100M, 400M or 700M.

Get the time after select operation.

The difference of the two is the duration that how much it takes to do the select operation.

c) Delete operations

i. 1MB,100MB,400MB and 700MB of Data

ii. Delete operations data with different size of each row (50K, 500K, 1M) into tables of different databases. Log the time taken. This is the way of selecting data into tables.

Get the time before delete operation.

Use loop to delete the data from the table until the size of data is the value what is needed, for example 1M, 100M, 400M or 700M.

Get the time after delete operation.

The difference of the two is the duration that how much it takes to do the delete operation.

f. Test the Cascade Integrity

One attribute is table A's primary key and it is table B's foreign key. Delete the date in attribute from table A. Do the select operation, 'Select the data from B'. If x cannot be selected from B, it means the DBMS has good cascade integrity in this test.

2.2.1.1. Connect to the Databases into different DBMSs

The databases are connected into the different DBMSs by using the ODBC technique. ODBC is one of ways to connect the database into DBMS. It established a set of standards, and provides a set of criteria for access to database (API). These API use SQL to complete most of task. ODBC itself also provides supports for the SQL. So, the users can directly send the SQL statement to ODBC.

Before using it, the ODBC drivers for each DBMS should be installed. Then use connecting string to connect into the different DBMSs.

2.2.1.2. Creating the databases into the DBMSs

In order to test different database fairly, the data which were put into each database are the same. The database model is showed in *Figure 1*.

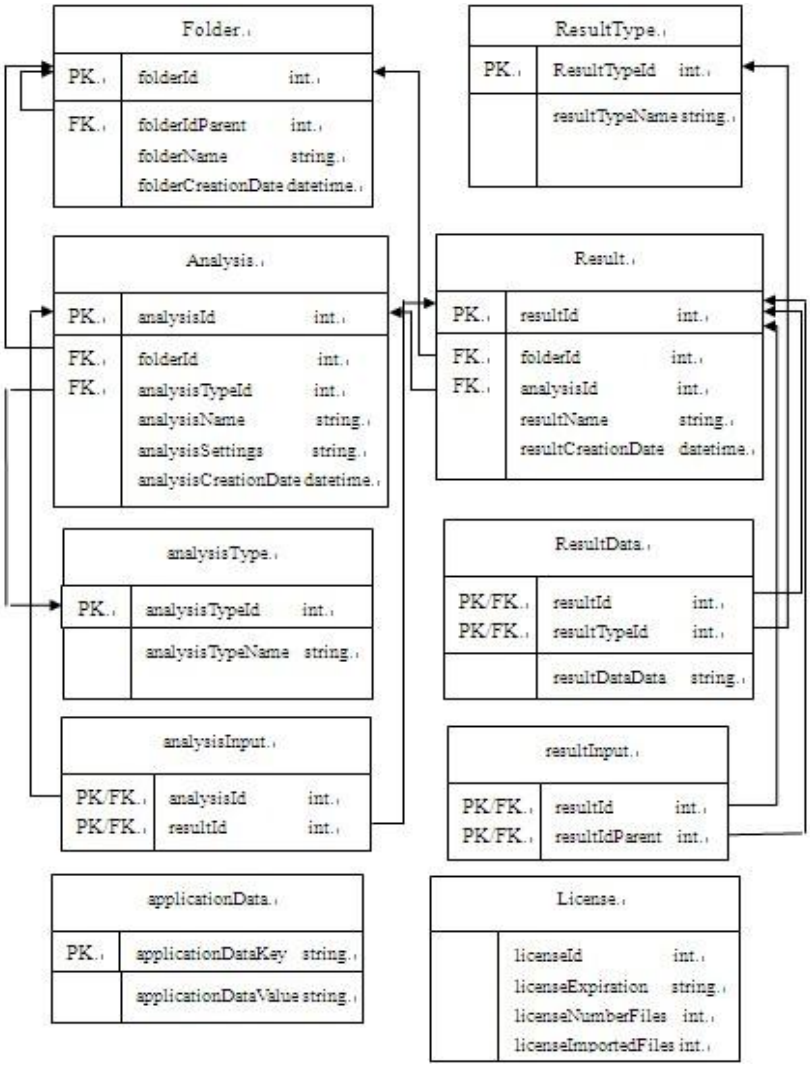

Figure 1: Database model

There are three types used in our tables. Each DBMS has its own schema data type, so, the tables should be created one by one. The information is shown in Table 4.

	Integer	String	Date/Time
SQLite	INTEGER(64-bit)	TEXT	DATE
Firebird	BIGINT(64-bit)	BLOB	TIMESTAMP
PostgreSQL	BIGINT(64-bit)	TEXT	TIMESTAMP
MySQL	BIGINT(64-bit)	TEXT	TIMESTAMP

Table 4: Different data types for each DBMS

2.2.1.3. Insert data

In order to test the DMBs performance, firstly each table must be populated with some data. As the suggestion of company, 3GB data are put into each database before testing. There are two main tables in the database. One is “ResultData”, and the other one is “Analysis”. When testing the speed, almost all the SQL operations are using these two tables. The method of putting data into database is using loops to put enough size data. There are four similar functions which provide inserting data for four DBMSs. In each function, using loops puts the enough data into the tables. This is a part of the pseudocode [9] about inserting data into one of tables:

Use loop to design the data to ensure the size of each row is fixed, for example 50K, 500K or 1M.
Use loop to insert the data designed before into table until the size of data is 3G.

2.2.2. TEST

2.2.2.1. Test Speed

To test DBMS, the speed consists of 3 parts: the speed of inserting data, reading data and deleting data. Reading data is divided into reading from one table or reading from several tables. Above all, the speed of reading data is the most important factor.

Each kind of SQL operations is tested from the small size of data to the large size. And these SQL operations are executed based on the different size of row. Every executing maybe has its own result to prove which one is the best, at that time we will get the result according the priority level.

When testing the speed of different database, it get the first time before executing the SQL operation. When the SQL operation is over, it get the second time. The second time minus the first time is the time we need, the totally time to execute the

SQL operation. We use this method to get the speed of inserting, reading and deleting data.

2.2.2.2. Test referential integrity

Referential integrity requires every value of one attribute or column of one relational table exist as a value of one attribute or column of another relational table or the same table. In the project we focus on the referential integrity about deleting, Cascades Delete to check the referential integrity. They Cascades Delete can delete the target row. At the same time, all rows that have same value (via foreign keys) are also deleted. Users can run data analysis and generate a result that is saved in the database in the application. If the user runs another the analysis using the first result as input, they will get another result as output which is also stored in the database. These are mentioned in the background. The tables in the database have the relationship according to the foreign keys. The database keeps track of the data analysis and achieved results. If there is something wrong with the referential integrity, Cascades Delete, the results wouldn't be right and lead to the analysis wrong. So the company put much concentration on Cascades Delete to make sure the results and analysis right after updating data.

2.2.2.3. Evaluate easy to use

The application with a DBMS should be transparent to the users. It should be easy to install. Users don't need to know how the application manages data internally. Everything the application needs should be set up automatically either during the installation or the first time the application starts without requiring that the user enters something a normal user wouldn't know, such as the path to the database. That means DBMS can be installed automatically with the installation of MBS or the setup program will install them first if they are not already present when MBS is installed.

3. RESULTS

After running our project, it can get the data about speed and Referential Integrity. They are used to analysis the performance of different DBMS.

3.1. SPEED

The data were got after running the project. We have drawn several figures according to the data about speed.

3.1.1. SPEED OF INSERTING DATA

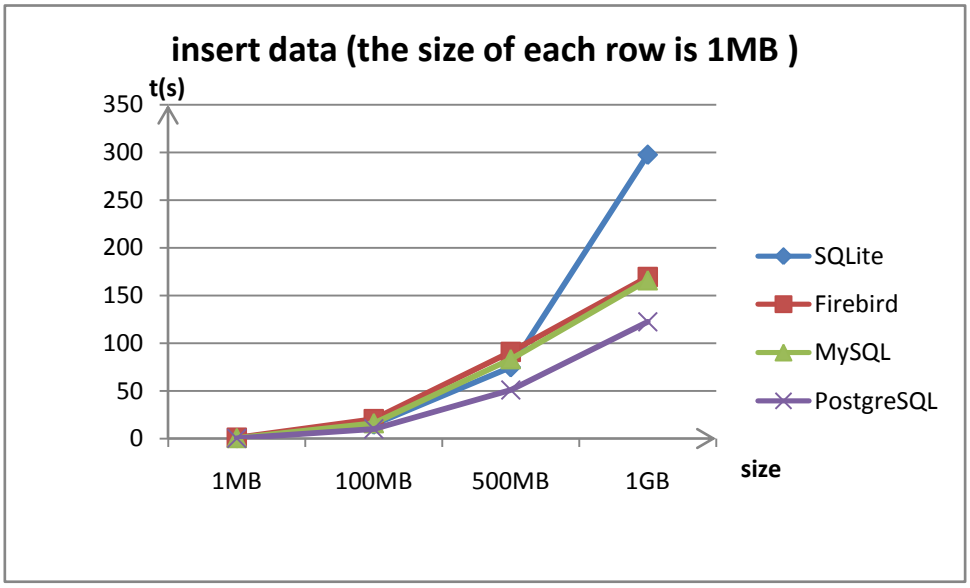

Figure 2: insert data (the size of each row is 1MB)

Test the speed of inserting 1MB, 100MB, 500MB and 1GB data into tables and the size of each row in tables is 1MB. According to the *Figure 2*, PostgreSQL had best performance. Firebird and MySQL have almost same graphic lines. They have little difference. But when inserting 1GB data into tables, SQLite spent more time than others obviously.

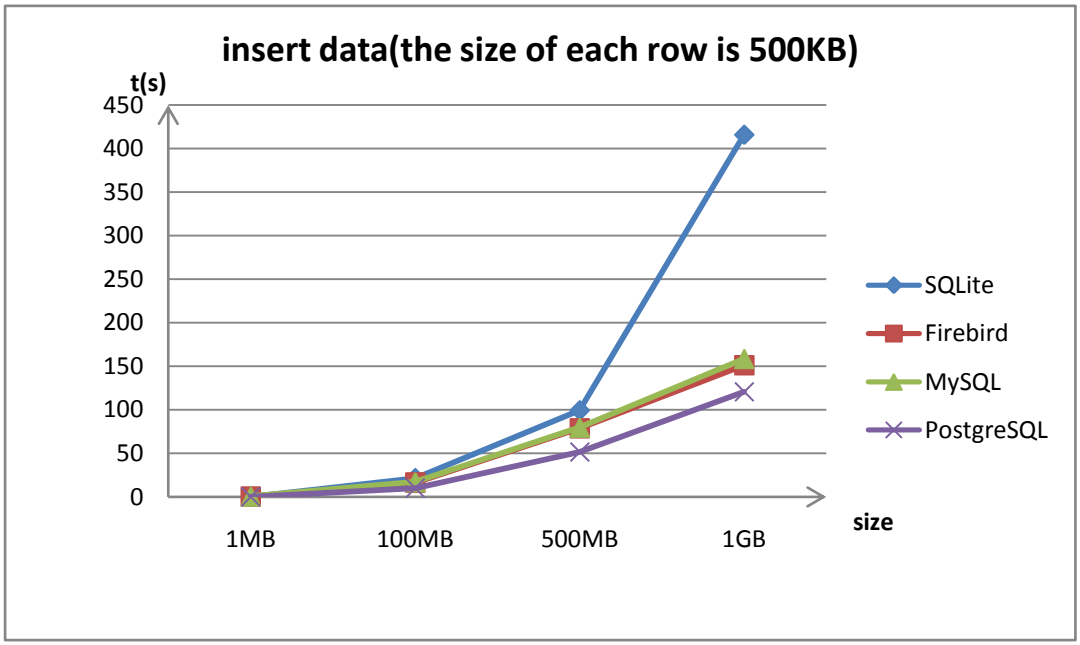

Figure 3: insert data (the size of each row is 500KB)

Test the speed of inserting 1MB, 100MB, 500MB and 1GB data into tables and the size of each row in tables is 500B. According to the 3, PostgreSQL had best performance. Firebird and MySQL have almost same graphic lines. They have little difference. Four DBMSs have small difference of inserting 1MB, 100MB, 500MB data. But when inserting 1GB data into tables, SQLite spent more time than other obviously.

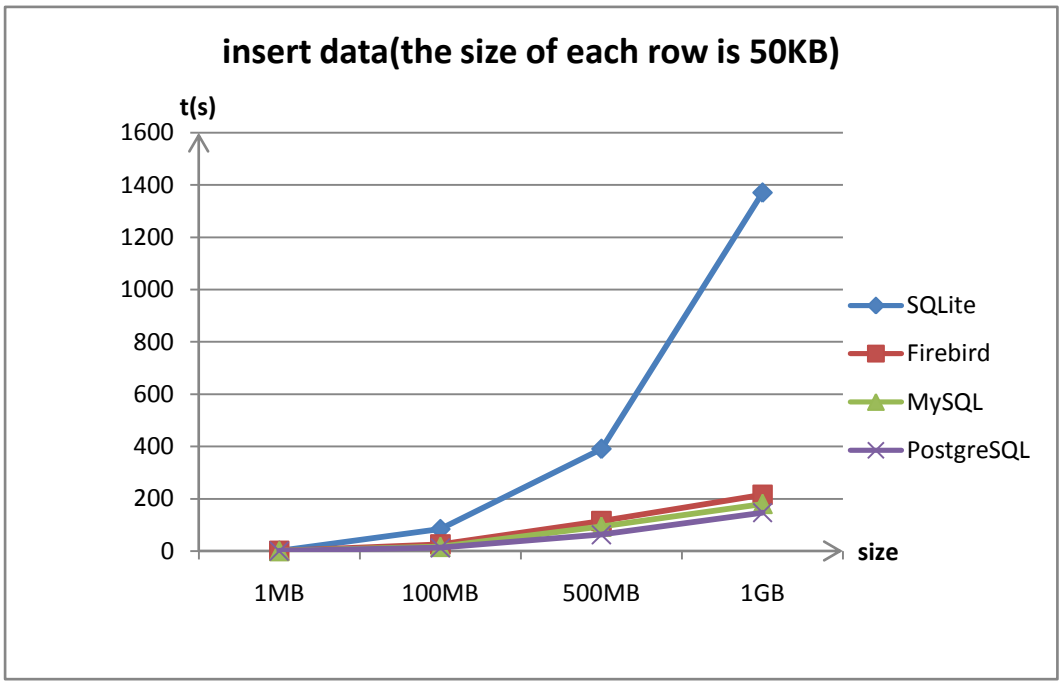

Figure 4: insert data (the size of each row is 50KB)

Test the speed of inserting 1MB, 100MB, 500MB and 1GB data into tables and the size of each row in tables is 500B. According *Figure 4*, PostgreSQL has the best speed. It shows if it inserts the same size of data, the size of each row is smaller, the speed of SQLite is slower. When they insert the data of 1GB, the difference is very obvious. Whatever the size of row is small or big, PostgreSQL is the fastest and SQLite is the slowest.

The data from these three figures shows PostgreSQL is the fastest when inserting the data, whatever the size of data that are inserted and the size of each row are small or big. However, SQLite is the slowest, especially when inserting the big size of data.

3.1.2. SPEED OF SELECTING OF DIFFERENT DBMS

Reading is divided into two parts. One is reading from single table, the other one is reading from several tables.

3.1.2.1. Select from single table

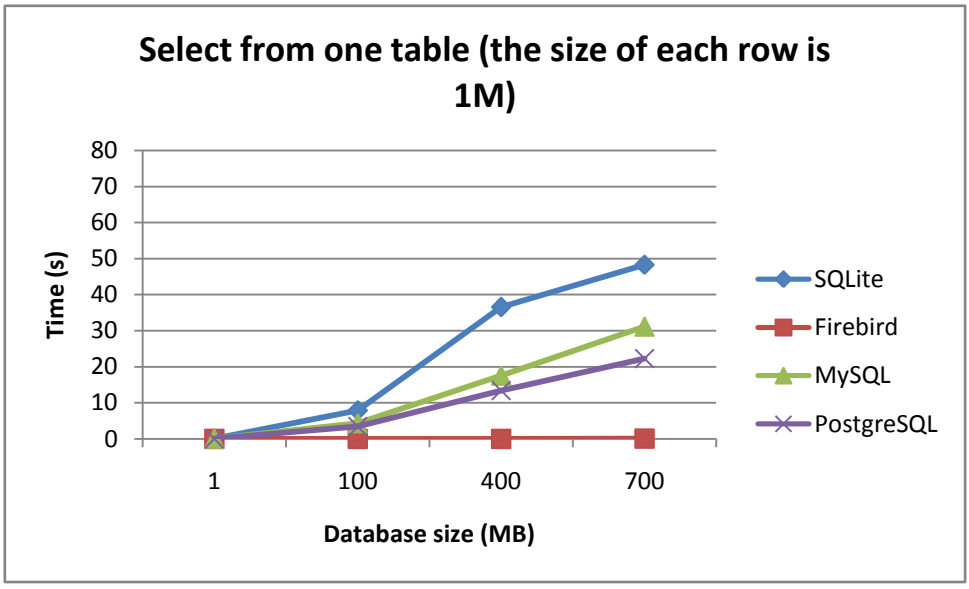

Figure 5: select from single table (the size of each row is 1MB)

Test the speed of selecting 1MB, 100MB, 400MB and 700MB data from the table and the size of each row in the table is 1MB. According to the *Figure 5*, Firebird has best performance. Four DBMSs have small difference when select 1MB and 100MB data. However, when select 400MB data and 700MB from the table, Firebird has best performance obviously. SQLite spends more time than others.

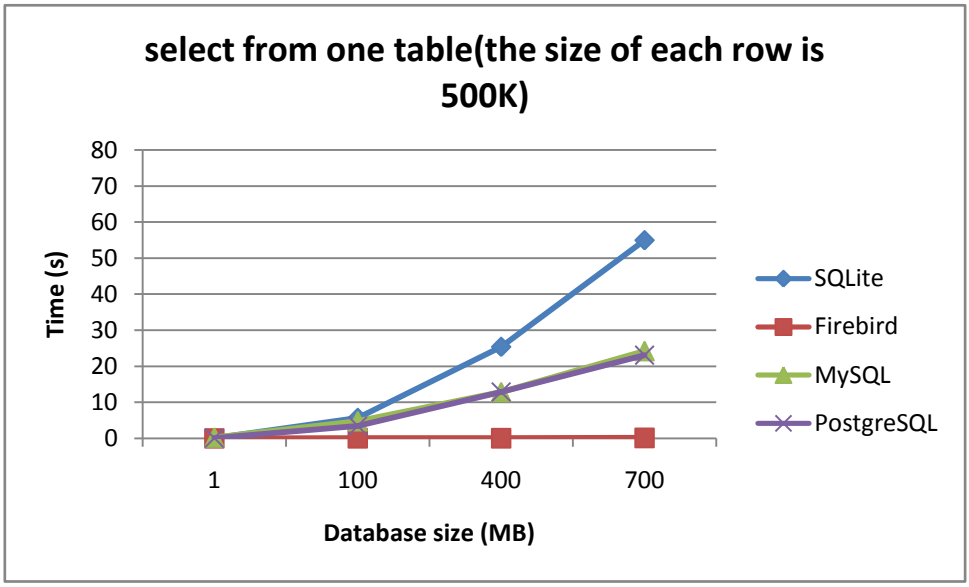

Figure 6: select single table (the size of each row is 500KB)

Test the speed of selecting 1MB, 100MB, 400MB and 700MB data from the table and the size of each row in the table is 500KB. According to the *Figure 6*, Firebird has best performance. Four DBMSs have small difference when select 1MB and 100MB data. MySQL and PostgreSQL have almost same graphic lines. However, when select 400MB data and 700MB from the table, Firebird has best performance obviously. Firebird has the stable line no matter how much data it was. SQLite spends more time than others.

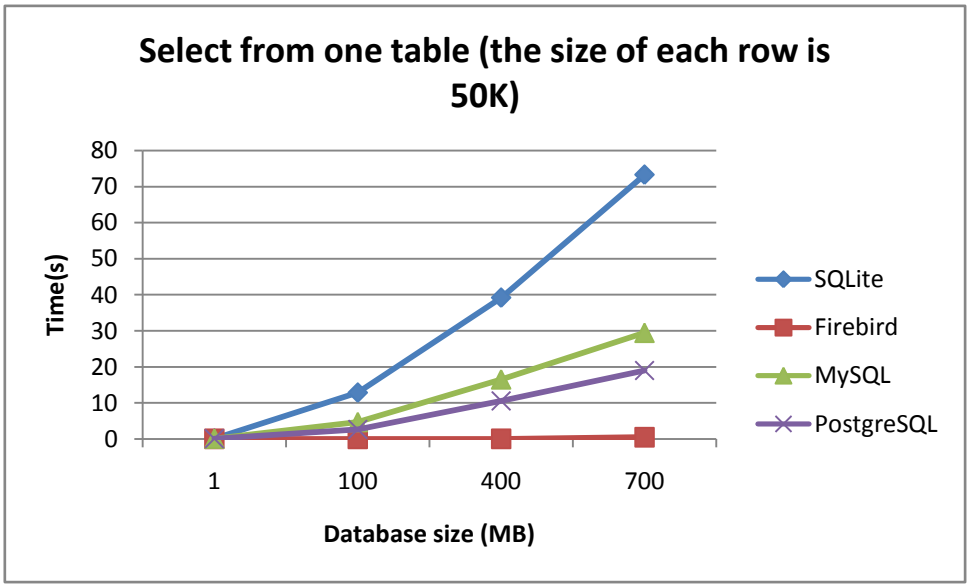

Figure 7: select single table (the size of each row is 50KB)

According to the *Figure 7*, we can find Firebird is fastest. When they select the data of 1MB from the single table, the time they spent is almost the same. They all spent no more than 1 second. When they select the data of 100MB, Firebird uses less than 1 second, but the other three databases all use several seconds, SQLite is the

slowest. When they select the data of 400MB or 700MB, the difference of the databases is obvious. Firebird is the fastest, PostgreSQL is the second, MySQL is the third, and SQLite is the slowest.

The data from these three figures shows Firebird is the fastest when selecting the data from a single table. Whatever the size of data and the size of data that are selected are small or big, it always uses less than 1 second to read. SQLite is the slowest.

3.1.2.2. Select from several tables

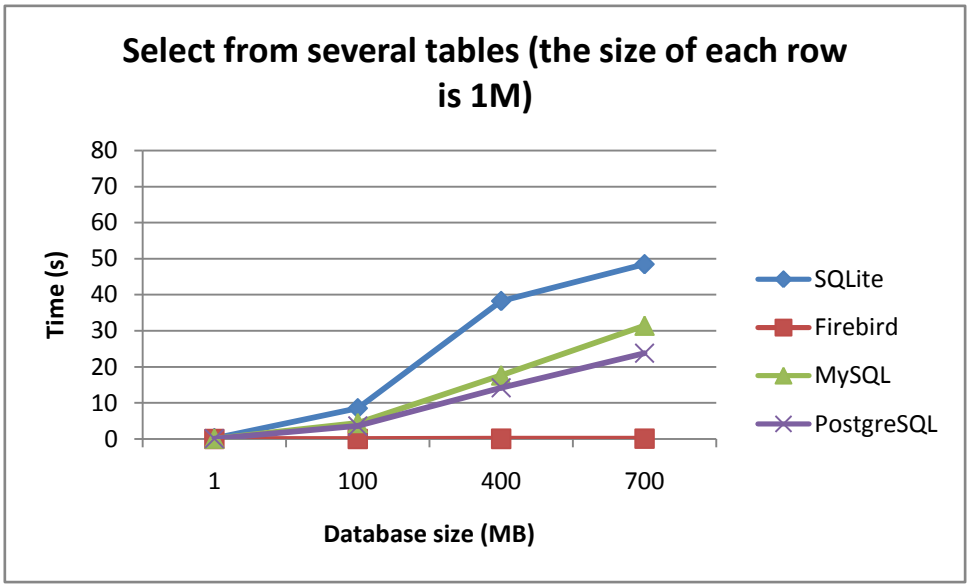

Figure 8: select from several tables (the size of each row is 1MB)

Test the speed of selecting 1MB, 100MB, 400MB and 700MB data from the tables and the size of each row in the table is 1MB. According to the *Figure 8*, Firebird has best performance. Four DBMSs have small difference when select 1MB and 100MB data. MySQL and PostgreSQL have almost same lines. However, when the data which are selected become bigger, Firebird has best performance obviously. SQLite spends more time than others.

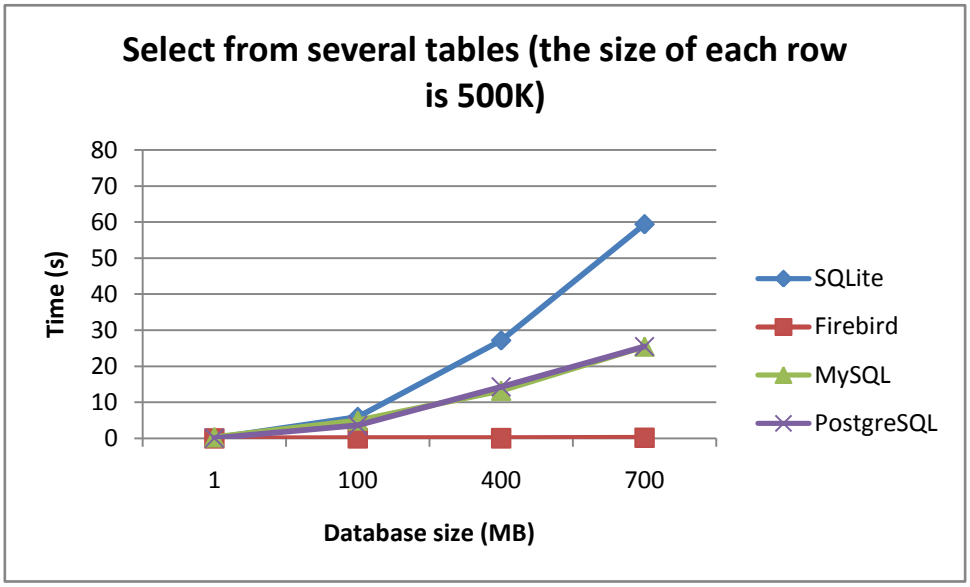

Figure 9: select several tables (the size of each row is 500KB)

Test the speed of selecting 1MB, 100MB, 400MB and 700MB data from the tables and the size of each row in the table is 500KB. According to the *Figure 9*, Firebird has best performance. Four DBMSs have small difference when select 1MB and 100MB data. MySQL and PostgreSQL have almost same lines. However, when the data which are selected become bigger, Firebird has best performance obviously. It uses no more than 0.8 second. SQLite spends more time than others.

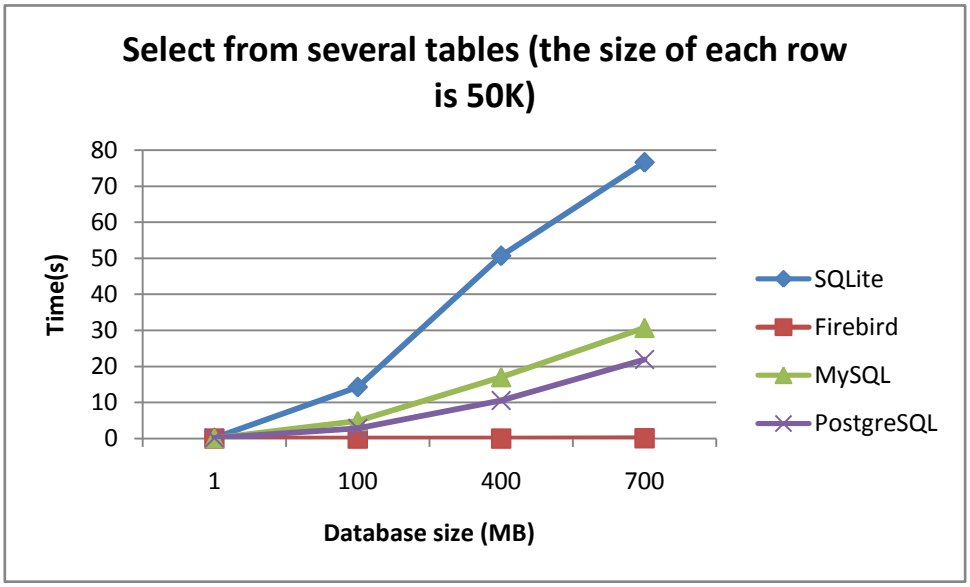

Figure 10: select from several tables (the size of each row is 50KB)

According to *Figure 10*, Firebird is the fastest. When they select the data of 1MB from several tables, they are all very fast. But when the size of data which they select becomes bigger, the difference is obvious. Firebird is the fastest, PostgreSQL is the second, MySQL is the third and SQLite is the slowest. And you if they select the same size of data, the size of each row is bigger, the speed of select is faster.

The data from these three figures shows the speed of selecting the data from several tables, the best one to the worst one is Firebird, PostgreSQL, MySQL and SQLite.

3.1.3. SPEED OF DELETING DATA

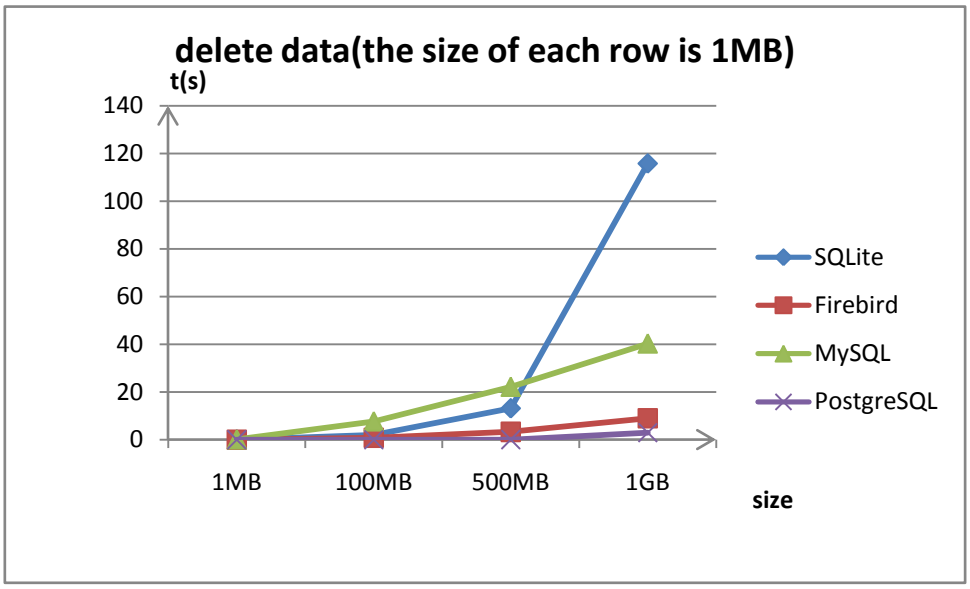

Figure 11: delete data (the size of each row is 1MB)

Test the speed of deleting 1MB, 100MB, 500MB and 1GB data from the tables and the size of each row in the table is 1MB. According to the *Figure 11*, PosrgreSQL has best performance. Four DBMSs have small difference when delete 1MB and 100MB data. But when delete 1GB data, SQLite spends more time than others. The difference is quite big.

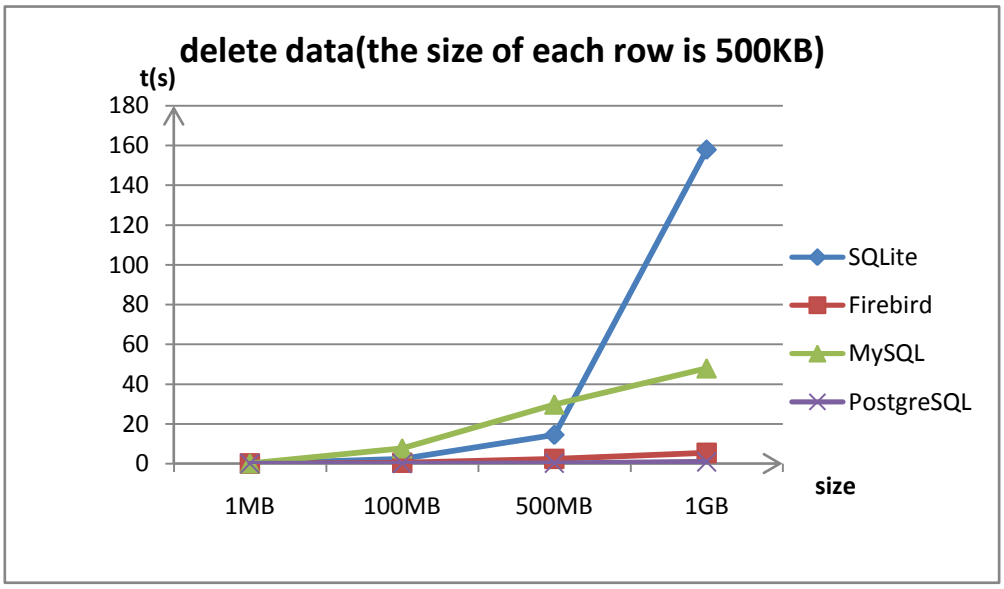

Figure 12: delete data (the size of each row is 500KB)

Test the speed of deleting 1MB, 100MB, 500MB and 1GB data from the tables and the size of each row in the tables is 500KB. According to the *Figure 12*, PosrgreSQL has best performance. Four DBMSs have small difference when delete 1MB, 100MB and 500MB data. But when delete 1GB data, SQLite spends more time than others. The difference is quite big obviously. It takes about 160 seconds.

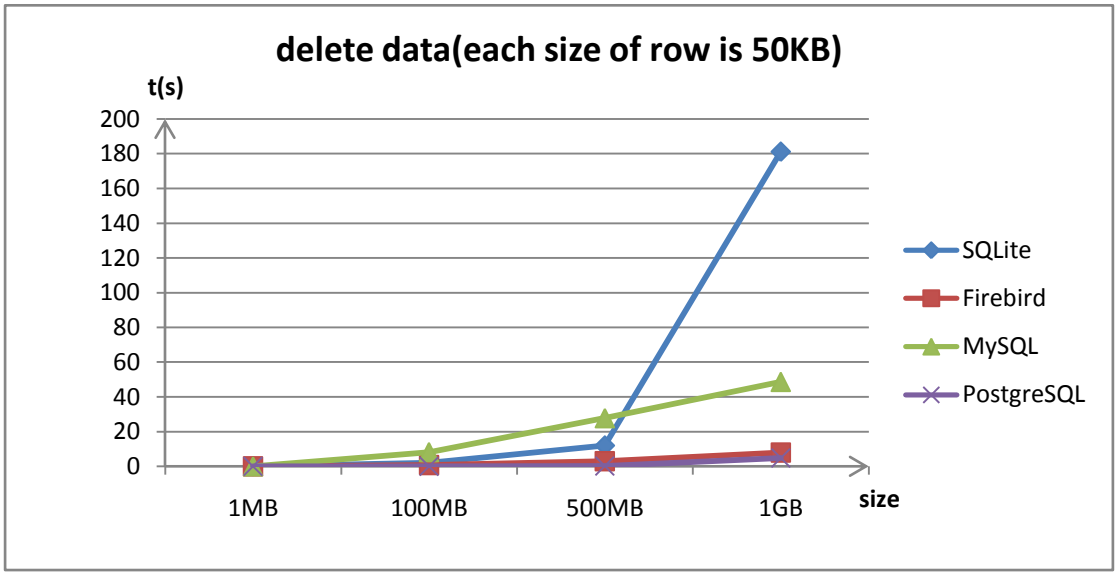

Figure 13: delete data (the size of each row is 50KB)

According to *Figure 13*, PostgreSQL and Firebird are both fast. When they delete the data of 1MB, they use the almost same time. When they delete the data of 100MB, they are also very fast, just MySQL is a bit slower. When they delete the data of 500MB, the speed PostgreSQL and Firebird is very close. SQLite is a bit slower than Firebird and PostgreSQL, and MySQL is the slowest. When they delete the data of 1GB, PostgreSQL and Firebird are much faster and SQLite is much slower than other databases.

The data from these three figures shows PostgreSQL and Firebird are much faster than others. When the size of data they delete is small, MySQL is the slowest. Otherwise, SQLite is the slowest.

3.2. REFERENTIAL INTEGRITY

In the code, there is a function called TestIntegrity(). The results whether the DBMSs have Cascade Delete come out from it. For example, the code executes delete statements to delete the data according to the foreign key among different tables firstly. Then the code executes select statements to check whether the data exist in tables and the relationship is right or not. At last, it prints the results. These are some parts of code of function TestIntegrity().

```
    OdbcCommand cmd1 = new OdbcCommand("Delete from folder where
folderID=1", conn);
    cmd1.ExecuteNonQuery();
    string del1 = "When the data is deleted from table “folder”, ";
    OdbcCommand cmd11 = new OdbcCommand("Select * from folder where
folderID=1", conn);
    OdbcDataReader sdr11 = cmd11.ExecuteReader();
    string inf11 = "the data from table “folder” is deleted successly!";
    using (sdr11)
    {
        while (sdr11.Read())
        {
            inf11 = "the data from table “folder” isn’t deleted! ";
        }
    }
```

```
SQLite:
When the data is deleted from table "folder", the data from table "folder" is deleted successly!
When the data is deleted from table "folder", the data from table "Result" is deleted successly!
When the data is deleted from table "folder", the data from table "Analysis" is deleted successly!
When the data is deleted from table "Result", the data from table "analysisInput" is deleted successly!
When the data is deleted from table "Result", the data from table "resultInput" is deleted successly!
When the data is deleted from table "Result", the data from table "resultData" is deleted successly!
When the data is deleted from table "analysisType", the data from table "Analysis" is deleted successly!
When the data is deleted from table "Result", the data from table "analysisInput" is deleted successly!
When the data is deleted from table "ResultType", the data from table "analysisInput" is deleted successly!
Firebird:
When the data is deleted from table "folder", the data from table "folder" is deleted successly!
When the data is deleted from table "folder", the data from table "Result" is deleted successly!
When the data is deleted from table "folder", the data from table "Analysis" is deleted successly!
When the data is deleted from table "Result", the data from table "analysisInput" is deleted successly!
When the data is deleted from table "Result", the data from table "resultInput" is deleted successly!
When the data is deleted from table "Result", the data from table "resultData" is deleted successly!
When the data is deleted from table "analysisType", the data from table "Analysis" is deleted successly!
When the data is deleted from table "Result", the data from table "analysisInput" is deleted successly!
When the data is deleted from table "ResultType", the data from table "analysisInput" is deleted successly!
MySQL:
When the data is deleted from table "folder", the data from table "folder" is deleted successly!
When the data is deleted from table "folder", the data from table "Result" is deleted successly!
When the data is deleted from table "folder", the data from table "Analysis" is deleted successly!
When the data is deleted from table "Result", the data from table "analysisInput" is deleted successly!
When the data is deleted from table "Result", the data from table "resultInput" is deleted successly!
When the data is deleted from table "Result", the data from table "resultData" is deleted successly!
When the data is deleted from table "analysisType", the data from table "Analysis" is deleted successly!
When the data is deleted from table "Result", the data from table "analysisInput" is deleted successly!
When the data is deleted from table "ResultType", the data from table "analysisInput" is deleted successly!
PostgreSQL:
When the data is deleted from table "folder", the data from table "folder" is deleted successly!
When the data is deleted from table "folder", the data from table "Result" is deleted successly!
When the data is deleted from table "folder", the data from table "Analysis" is deleted successly!
When the data is deleted from table "Result", the data from table "analysisInput" is deleted successly!
When the data is deleted from table "Result", the data from table "resultInput" is deleted successly!
When the data is deleted from table "Result", the data from table "resultData" is deleted successly!
When the data is deleted from table "analysisType", the data from table "Analysis" is deleted successly!
When the data is deleted from table "Result", the data from table "analysisInput" is deleted successly!
When the data is deleted from table "ResultType", the data from table "analysisInput" is deleted successly!
```

Figure 14: referential integrity result

The *Figure 14* shows the result of testing integrity. This is the information we get when we test referential integrity. This figure shows every database has the referential integrity.

3.3. EASY TO USE

The company provides application execution files for users who want to use MBS. When users install MBS, the files must provide everything which will be used. Users don’t need to install the database after they install MBS.

3.3.1. FIREBIRD

Firebird is embedded DBMS. If the company uses it, they just need to put the dynamic link library of each DBMS into MBS. After users install MBS, they can use DBMS without installing it.

3.3.2. SQLITE

SQLite is embedded DBMS. The company is using it. They put the dynamic link library of each DBMS into MBS. After users install MBS, they can use DBMS without installing it.

3.3.3. MYSQL

MySQL is not embedded DBMS. If the company uses it, they have to create application execute files include MBS and DBMS. When users click application execution files, they can install MBS and DBMS together. The figures show the steps of installing MBS and MySQL.

Figure 15: Welcome to the Medicwave Bioinformatics Suite Setup Wizard

When you want to install the software, you can see the “welcome” interface. It looks like *Figure 15*. It doesn’t do anything and click “Next” to continue.

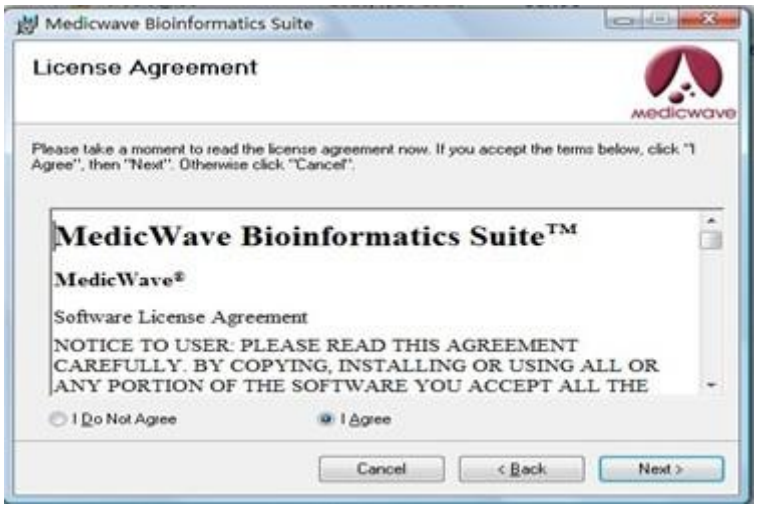

Figure 16: License Agreement

Figure 16 is the license of MedicWave Bioinformatics Suite. If you don’t agree with the license, you cannot continue to install. Otherwise you choose “I Agree” and click “Next” to continue.

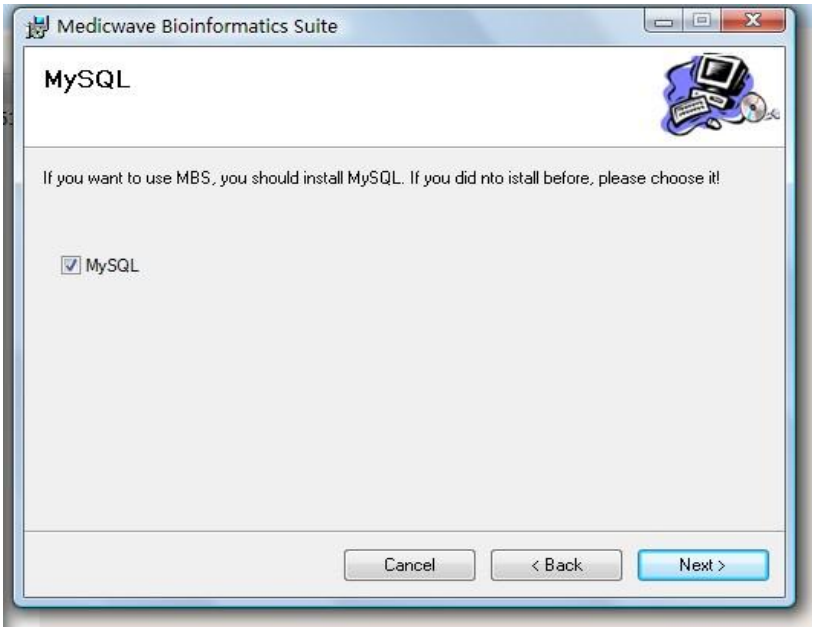

Figure 17: MySQL

The aim of *Figure 17* is to ask you whether you want to install MySQL together or not. If you have installed MySQL in your computer before, you should not choose “MySQL”. Otherwise, you should choose it to install, because the application can only work with MySQL together. No matter you choose “MySQL” or not, you can click “Next” to continue.

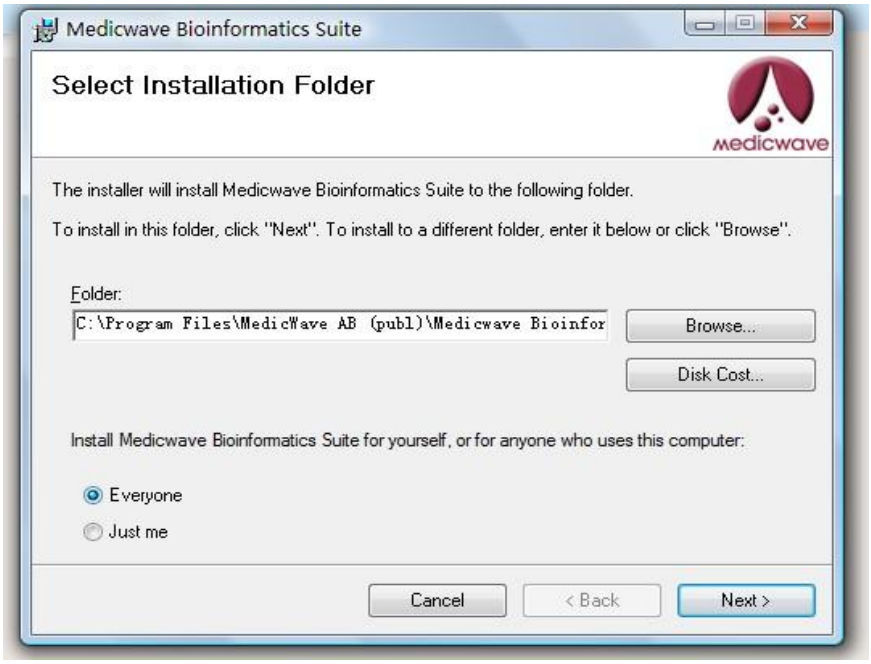

Figure 18: Select Installation Folder

Figure 18 is to ask you which folder you want to install. There is a default folder in *Figure 18*, if you don't change it, Medicwave Bioinformatics Suite will be installed in the default folder. Click "Next" to continue.

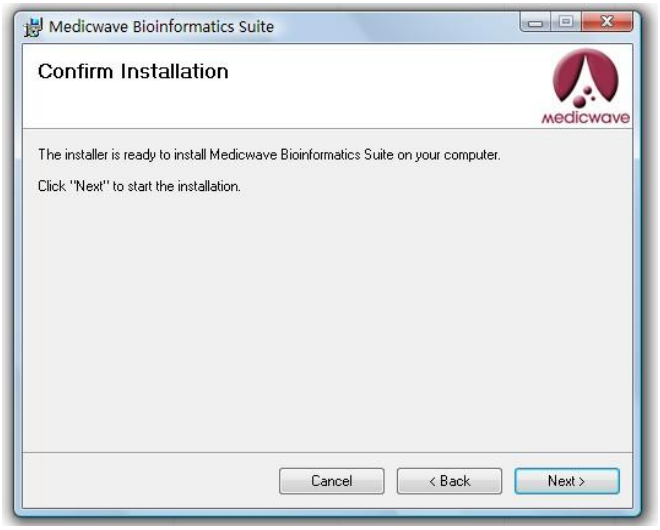

Figure 19: Confirm Installation

Figure 19 is the confirm interface. It is to confirm whether you want to install the software or not. Click "Next" to continue.

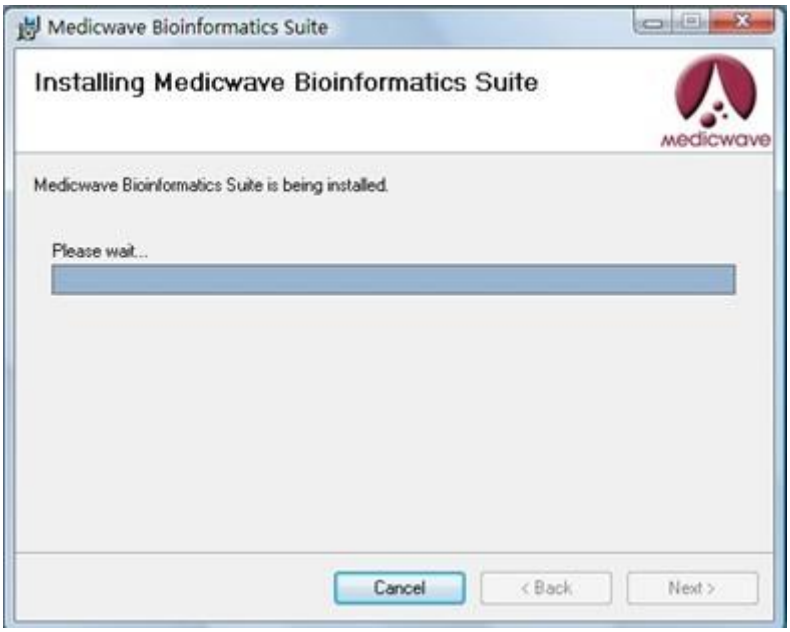

Figure 20: Installing Medicwave Bioinformatics Suite

Figure 20 shows the process of installing.

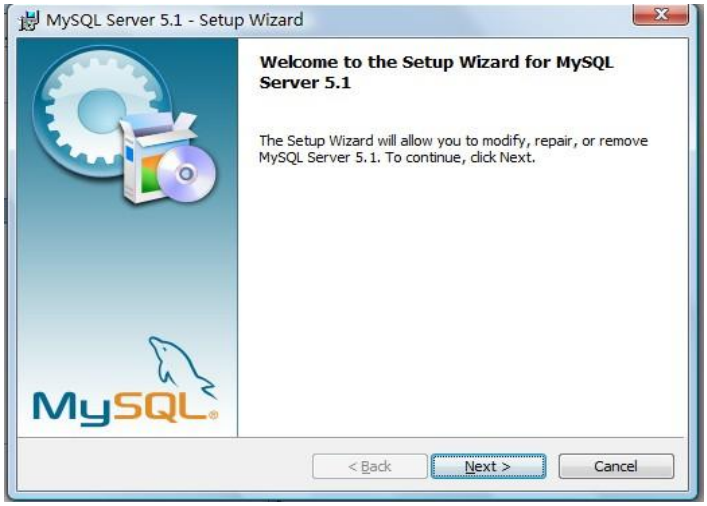

Figure 21: Welcome to the Setup Wizard for MySQL Server 5.1

If you chose to install MySQL together, from now on, the installing of MySQL is beginning. *Figure 21* is the welcome interface. Click “Next” to continue.

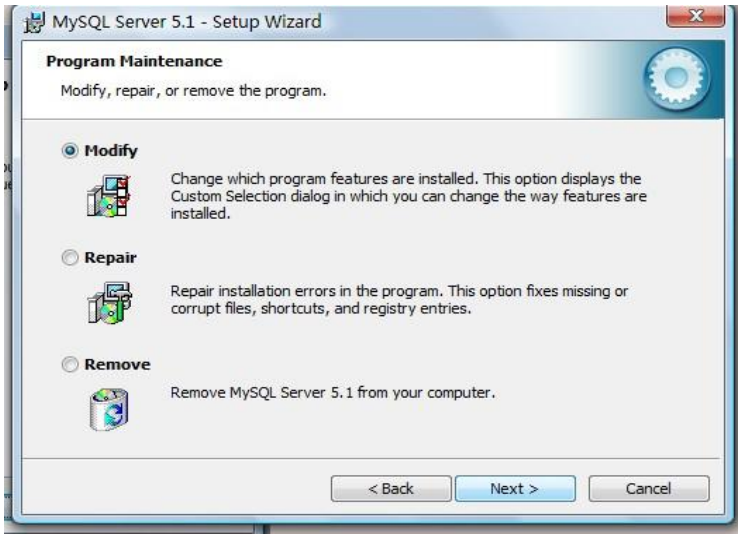

Figure 22: Program Maintenance

Figure 22 is to ask you that you want to modify, repair or remove it. You can choose "Modify" and click "Next" to continue.

Figure 23: Ready to Modify the Program

Figure 23 is the ready interface. You can click "Install" to continue. When the install is over, the whole software is finished.

3.3.4. POSTGRESQL

PostgreSQL is not embedded DBMS. If the company uses it, they have to create application execute files include MBS and DBMS. When users click application execution files, they can install MBS and DBMS together. It is similar to the process of install the application with MySQL. The only difference is that finishing installing MBS, it starts to install PostgreSQL instead of MySQL. The figures don't be shown here, you can consult the process of install MBS with MySQL.

3.4. SUMMARY

According to the tests before, each database has its own superiority.

In terms of speed, Firebird and PostgreSQL has the great advantage in the operation of inserting data. In some case, the speed of PostgreSQL is a little faster than Firebird. In the operation of selecting data, the performance of Firebird impress us, it is much faster than other databases. Firebird is also the fastest in the operation of deleting data.

In terms of referential integrity, all the DBMSs have the good referential integrity.

In terms of easy to use, each DBMS can install while installing the application. However, embedded DBMS only need to copy some dll files while making the setup program, so embedded DBMS is easier than the disembedded DBMS.

At last, let's compare with the related work. We test the performance according to the requirement from the company. They don't need to do the operation with index, so when we testing, we only test without index. Form our result of testing, firebird is the best choice for the company. To review the related work, the data got by our testing is not exactly the same with the related work. The reasons are the difference of the table structures, the difference of the DBMS versions, the difference of operated data, the difference of test environment, the difference of test method, and the difference of test conditions and so on. These reasons led to the different testing result between ours and the related work. But the conclusion is the same. From the related work, Firebird is the best while doing the operation without index, what is the same conclusion with our testing.

4. CONCLUSION AND FEATURE WORK

The key point of the project is to find the most suitable DBMS for the application, MBS. All the tests are under the requirements and on the expectation of the company. According to the results from quantitative and qualitative assessments, Firebird was the DBMS with better results, such as

a. Firebird has the license, InterBase Public License (IPL) and Initial Developer's Public License (IDPL), which meet with the requirement. For IPL and IDPL, if modifications are created in files independently, the contents of the files can be distributed under a license which is different from the original license. The two licenses allow firebird to be used in any projects including commercial, without paying license fees. They don't request users open their source code. The company doesn't need to open their source code when they choose Firebird as their DBMS for MBS.
b. The company treats the performance of reading data is very important requirement. Firebird spends the least time selecting data. The advantage is obvious. For the speed of inserting and deleting, Firebird spends least time among testing DBMSs.
c. The Firebird has referential integrity. The Cascades Delete in Firebird works.
d. Users don't need to install it when they install MBS. It is easy to use.
e. It is free for commerce use.

Firebird also has other advantages features. It originally started its life as the Borland InterBase database, so it has long history and widespread use, which makes it high stable and conforms to entry-level SQL-92 requirements. For embedded edition, Firebird has very small footprint, no more than 3MB and support unlimited database(limited by file system).

In summary, Firebird is the best choice for the MBS application.

Because of the limited time, not all the features of the DBMS have been tested, for example, memory share, transaction, triggers. Future work could include tests using variant .Net database providers, more complex queries, performance under ADO.NET Entity Framework, and so on.

New experience and knowledge about database system have been gained during the project. The project improves the students' research skills in how to evaluate and test DBMS, in theory and practice.

5. REFERENCE

[1] Medicwave AB. Available at: http://www.medicwave.com
[2] SQLite. Available at: http://www.sqlite.org
[3] http://www.ifross.org/ifross_html/lizenzcenter-en.html
[4] http://developer.kde.org/documentation/licensing/licenses_summary.html
[5] MySQL. Available at: http://www.mysql.com/?bydis_dis_index=1
[6] Firebird. Available at: http://www.firebirdsql.org/
[7] PostgreSQL. Available at: http://www.postgresql.org/
[8] Comparison of SQLite, MySQL, PostgreSQL and Firebird. Available at: http://www.sqlite.org/cvstrac/wiki?p=SpeedComparison
[9] http://users.csc.calpoly.edu/~jdalbey/SWE/pdl_std.html