ACADEMIC LABORATORY INFORMATION MANAGEMENT SYSTEM:

A TOOL FOR SCIENCE AND COMPUTER SCIENCE STUDENTS

Spencer Lerch

Submitted to the faculty of the School of Informatics
in partial fulfillment of the requirements
for the degree of
Master of Science in Chemical Informatics,
Indiana University

August 2007

i

Accepted by the Faculty of Indiana University,
in partial fulfillment of the requirements for the degree of
Master of Science in Chemical Informatics

**Master's Thesis
Committee**

_____
Mahesh Merchant, Associate Professor, Chair

_____
David Wild, Assistant Professor of Informatics

_____
Thompson N. Doman, Adjunct Associate Professor

This Project is dedicated to everyone that has thought What If….? And made it a reality

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

A sincere wish of gratitude to my parents, fiancé, friends and furry little minions who

have stood by and supported my efforts in this program, even through all the craziness.

ABSTRACT

Spencer Lerch

Proof of Concept - An Academic LIMS application:

The aim of this project is the creation of an open-source, freeware LIMS application that can be used in an academic setting as a teaching tool for both chemistry and computer science students. The LIMS package will combine an application, developed using VB.NET, to manage the data with other open-source or freeware programs such as MySQL and WEKA.

The numerous commercial chemical informatics applications available are useful tools to learn how to manage data from a user's standpoint. However, they are not readily available to the average student, nor do they offer a great understanding into how they were developed from a programmer's frame of mind. There is a great void here that, if filled can greatly help the academic community.

CHAPTER ONE: INTRODUCTION & BACKGROUND

Importance of an Academic LIMS

Douglas Perry puts forward the best argument in an article called:

LIMS in the academic world: This valuable research tool flourishes in the commercial sector, yet in the academic sector it sits untapped.[1]   The article discusses the disconnect between Academics, LIMS Vendors and the private sector and how this disconnect has generated an absence of a LIMS in an academic setting.   He reasons that "The cultural divide hurts both business and academia. Academic scientists are missing out on a potentially valuable research tool, and LIMS companies are missing the opportunity to establish a firm foundation for their products in the larger scientific community."

Certainly, as standard operating procedures shift in labs world wide from paper based data acquisition to a more purely digital environment it is important that these new systems are built and used to the best of their abilities.   The aim of this project is to create a tool that can benefit both the users and developers by starting their education in the academic world.

Introduction to LIMS

There are numerous commercially developed Laboratory Information Management Systems, LIMS, being used in laboratories around the world, including packages from LabVantage, LabWare, PerkinElmer and Tripos, but few to none designed for an academic setting specifically.   A web search for open-source and/or academic LIMS turned up very few examples.   Among them were SIP, Sample Inventory

---

[1] http://pubs.acs.org/subscribe/journals/tcaw/11/i01/html/01comp.html

Program[2], HalX, an "electronic lab book that aims at (i) storage and (ii) easy access and use of all experimental data"[3] from high-throughput experiments of biological data, and MOLE, 'Mining, Organizing and Logging Experiments', which was designed for the "growing data management and target tracking needs of molecular biologists and protein crystallographers"[4]. I will endeavor to go into more detail about these products and their capabilities and why there is still a need for an academic LIMS for chemistry.

NuGensis (www.waters.com) offers a Scientific Data Management System (SDMS), which is quite comprehensive package to manage data within a large laboratory setting. The system can manage files from instrumentation and business applications and capture data directly from lab instruments. It is described as being "Ideal for daily data management and storage strategies", "Necessary for compliance and protection of intellectual property" and "Essential for the collaborative flow of information throughout your organization". The SMDS can exchange "data with electronic laboratory notebooks (ELNs), LIMS, EDMS and other common enterprise technology systems".[5]

One of the components of the SDMS package is called the Vision Publisher[6]. It is the next version of Water's ELN package that allows one to import and combine multiple files from differing sources. The Vision Publisher optimizes utilization of the information collected and cataloged by the SDMS to create reports or summaries of the scientific data. The application is able to manage several kinds of data including:

---

[2] http://www.enfoldsystems.com/Products/Open/SIP/demos/SIP_DEMO
[3] http://scripts.iucr.org/cgi-bin/paper?S0907444905001290
[4] www.cse.clrc.ac.uk/Publications/1431/final_paper_PSFB.doc
[5] http://www.waters.com/WatersDivision/ContentD.asp?watersit=JDRS-5WJQ2W
[6] http://www.waters.com/WatersDivision/ContentD.asp?watersit=JDRS-5WJPK5

- Chemical structures and reactions
- Analytical data such as spectra, chromatograms, and parameters
- Text, authored reports, and comments
- Spreadsheets and tables
- Images and drawings, including pictures
- Scans, movies, and multimedia files

It is also capable of storing the meta-data from the files, not just the files themselves, making that meta-data available to powerful searching tools including ones able to search chemical structures, spectra and chromatograms.

LabWare (www.labware.com) offers a similar product with a wide range of capabilities. A user can enter Project and Sample data as well as Result entry for those samples setup for the Projects. The LIMS has components to connect with and manage lab instrumentation, including the ability to calibrate the instruments. Data from the instruments can be imported by three different means:

- Serial Port communication
- Importation from files generated by the instruments
- Reports stored in the NuGenesis/Vision product mentioned earlier

For the data that cannot be obtained from instrumentation, there is also an E-Lab Notebook to let users "record information that is not easily captured in electronic form".[7] There is an external links and document management system that connects data to external files that are located somewhere on the network. The files can be linked to the LabWare LIMS, copied to a central server or even imported into the database.

Another content management system is the web-based Cerity, now known as OpenLab[8], from Agilent (www.chem.agilent.com). It stores all of the lab information in electronic format where it can be organized and made searchable to users. The web-

---

[7] http://www.labware.com/LWWeb.nsf/lp/en040109
[8] http://www.chem.agilent.com/Scripts/PDS.asp?lPage=16769

3

based interface allows easy "access [or] any file, anytime from anywhere for review, collaboration and reporting".[9] Besides, Cerity/OpenLab, Agilent has other components including the QC Client LIMS that has numerous capabilities including:

- Powerful data extraction tool
- Flexible instrument connectivity
- IT Infrastructure
- Flexible reporting

There are many more products from Agilent which I shall not go into here because they are not as relevant to this project.

The final commercial product that I will describe is by Tripos, which is best known for its 3D software packages for life science studies. Currently Tripos has a set of applications in its Benchware suite including Benchware DataMiner and Benchware Notebook which are designed to gather and analyze laboratory data. The most distinctive component is the Benchware 3D Explore which is a visualization tool for 3D chemical structures giving the user a better understanding of "complex molecular data such as protein-ligand crystal structures, docking results, molecular alignments, or other 3D chemical information."[10] The tool has the means to integrate with 3rd party applications such as ChemDraw and PowerPoint as well as give the user the chance to model new chemicals based on existing data before the actual synthesis is performed.

I've also looked at several free LIMS found through web searches. The first one I found is a java application called Hal-X that is designed for small- to large-scale biochemistry or molecular biology laboratories. The application tracks Samples,

---

[9] http://www.biocompare.com/multimedia/88/Agilent-Cerity-Enterprise-Content-Management-(ECM)-from-Agilent-Technologies.html
[10] http://www.tripos.com/index.php?family=modules,SimplePage,bw3d

4

experiments and Work Flows which it defines as a "succession of experiments".[11]   Also designed for biological labs is the application MOLE, described as a 'preLIMS'. It too tracks experiments and targets for protein analysis.   The limited amount of literature at the website (http://www.mole.ac.uk/demo/help/overview.php) indicates that the application is designed to be a web-based application. Any attempts using IE and Mozilla, though, to get more information from the different components of the application failed to navigate the user to the correct page so more detailed information was not available.

The most comprehensive open-source LIMS found was the Sample Inventory Program from Enfold Systems (http://www.enfoldsystems.com/Products/Open/SIP). The LIMS was built for the Southwest Foundation for Biomedical Research using a combination of the Zope application server, either the PostgreSQL or Sybase Relational Database Servers and the Python 2.3.5 language.   However, the purpose of the LIMS is limited to the management of laboratory samples and the projects with which they are involved.

Research Questions

As far back as ten years, there was little to no usage of LIMS in undergraduate chemistry classrooms or labs and that trend seems to be true today.   Yet more and more graduates are entering jobs where they will be using one.   While there are numerous sources of LIMS available on the market, the commercial LIMS's costs are too prohibitive to be put into use within classrooms or labs on large scales in colleges, universities where there is a more limited budget.   Such a situation might be suitable for

---

[11] http://halx.genomics.eu.org/DOCS/

labs that need to do work which require legal certification for the work being done in them.   However, this does not necessarily need to apply to low level chemistry labs or the standard classes.

These same applications, while providing a user the means to see what can be done, do not show the user how these functions or application components are created or why they were designed as they were.   This is understandable since they are commercial products and competing firms do not want to share their methodology and technology, but that limits developers from learning good or useful techniques to develop the software.   Of the open-source LIMS, they do provide more insight into the code behind the applications, but they are also more limited in their tools and capabilities than the commercial LIMS.   From these factors, I have established several questions which this project is intended to help answer:

Q1:   What is the thought process in designing a LIMS application?
Q2:   How can the data be stored? Accessed?
Q3:   What kind of interfaces will work?
Q4:   What programming language(s) can be used?
Q5:   What 3rd Party tools can be used?
Q6:   Can this be done in such as way to make it inexpensive to develop or use?

The construction of the application will be comprised of two main components: a back-end database to manage the data and front-end GUI allowing the user a means to interface with the data. The back-end storage will be handled using a free database system, the three prime ones being Microsoft Access (www.microsoft.com), Postgresql (http://www.postgresql.org/) and MySQL (www.mysql.org). The results are reported over the next two chapters. The third chapter is devoted to the development of the database and the fourth chapter to the development of the application and the integration 3rd party tools.

Database Design

The MySQL 5.x database (Database Server Community Edition) was determined to be the best choice of the three for several reasons. It is free open-source database platform with solid capabilities to handle small to medium data loads for the environments which this application is intended. It can function on numerous computer operating systems including Windows, Linux, Solaris and Mac OS-X. Microsoft Access is not intended to run on non-Windows environments and does not handle working with multiple users at a time[12]. Postgresql is intended for more multi-user environment, but it is not supported under Windows NT, 9x or ME and even to run on windows requires the cygwin emulation[13].

---

[12] http://www.weberdev.com/ViewArticle/Access-vs.-MySQL
[13] http://www-css.fnal.gov/dsg/external/freeware/pgsql-vs-mysql.html

7

Version 5.0 of MySQL will be installed on the development machine as will a 3rd party application, DBManager Professional Version 3.0.3a Freeware Edition, obtained from http://www.dbtools.com.br.   DBManager is a useful and free application that can be used to manage MySQL databases, tables, relationships and data.   Manipulation of the data and the tables will be done using a combination of DBManager, the LIMS being created for this project, and a tertiary application developed to act as the setup component of the LIMS called LIMSDBMGR.

Application Design

The front-end GUI had several parameters to help determine what language or languages could be used to build it.   The programming language had to be able to connect to the selected backend database, it had to be easy to program and have a lot of functionality to provide the end user.   Based on these parameters, the VB.NET programming language was chosen to be the primary programming language.   It has strong database relation components and the tools to create a robust GUI with numerous capabilities. Later developments can be modeled on the VB.NET frame to transport the application to other operating environments including ASP.NET for a web-based version.

The application was initially developed using Microsoft Visual Studio 2002 with .NET Framework 1.0. This was later changed to Visual Studio 2003 and .NET Framework 1.1 which are more secure and reliable for the development process.   One component of the application and the Help Files were created as HTML files using the Notepad application.

The GUI is intended to give the user working knowledge of several LIMS

8

concepts. The first Phase touches on Instrument management and Data Result Entry. Phase II will expand on the Data Result architecture and add a data analysis tool and a reporting tool. The final phase will incorporate the most new components which will include a Chemical Library, File Management tool, the ability to share data, and a security structure that will limit the access of data to different users. Some of the new Phase III components will also attempt to include several 3rd party applications or tools integrated into the application. These include Crytal Reports for Visual Studio to create reports, the web-based plugin CHIME and the JME Molecular Editor to view the 2D & 3D structures entered into the database, and WEKA (www.cs.waikato.ac.nz/ml/weka/) to create graphical representations of the data.

9

CHAPTER THREE: DATABASE DEVELOPMENT

The first step to creating the database began with the installation of the MySQL

Server.   The free MySQL 5.0 Community Server - Generally Available (GA) Release

was obtained from the MySQL developer website at

http://dev.mysql.com/downloads/mysql/5.0.html#win32.   The "Windows (x86)

ZIP/Setup.EXE" file was chosen, downloaded, and installed on the local development

machine.[14]   Once MySQL server was installed, the DBManager Pro application was then

installed and used to create the database called 'lims'.   All tables referenced in this paper

are solely found in this database.

The final version of the LIMS requires twenty tables to house all of the

information.   The development of the tables came about through three phases of

construction which will be described here after.   The full listing of those tables can be

seen in Table 3.1, Table 3.2, and Table 3.3.

Database Table	Table Description
tbldevices	List of devices that can be used manually or with the LIMS
tbldata_types	List of Data Types for devices such as pH or Centigrade
tbldevice_types	List of Data Types and the devices to which they are associated
tbldata	The data as entered by the user or pulled into the system from an outside file or device
tbltransactions	A list of sql transactions that have been entered by the LIMS

Table 3.1   Phase I Tables

---

[14] Additional installation instructions can be found at ttp://dev.mysql.com/doc/refman/5.0/en/installing.html

Database Table	Table Description
tblnote_types	List of note types for experiments
tblexperiments	List of user experiments
tblexperiment_notes	Notes relating to the experiments made by the users
tblbatches	List of batches associated to experiments – each experiment can have one or more batches
tbldata_notes	Notes related to the data retrieved

Table 3.2   Phase II Tables

Database Table	Table Description
tblaccess_levels	Access Levels granted to users
tblusers	List of users setup to use the LIMS
tblexperiment_sharing	Denotes which and how experiments are shared between users
tblfile_types	List of file types that can be imported into the system for storage and retrieval
tblfiles	Files are stored here to record version control and manage files in a single location
tblchemicals	List of chemicals that have been used by the users in one or more of the experiments
tbltraits	List of chemical trait types
tblchemical_traits_numeric	Stores the value for numeric trait types
tblchemical_traits_string	Stores the value for string trait types
tblbatch_chemicals	Lists the chemicals, if any used in a particular batch

Table 3.3 Phase III Tables

Database Construction

Creation or modification of the database and its tables was originally done using the DBManager Pro application solely.   Starting in Phase II, alterations to the table structures evolved enough that a new application was created to expedite the process. The LIMSDBMGR application was built to access a set of sql files that could be run to properly create each table and add the corresponding the default records.   Prior to the implementation, I had to reenter the default data after each major change to the database.

This was time consuming and drawing my time away from the development of the LIMS.

A second advantage of developing this application, was that it has laid the groundwork

for a component that could be used for new users to install the LIMS application.   A

screenshot of the application is shown below in Figure 3.1.
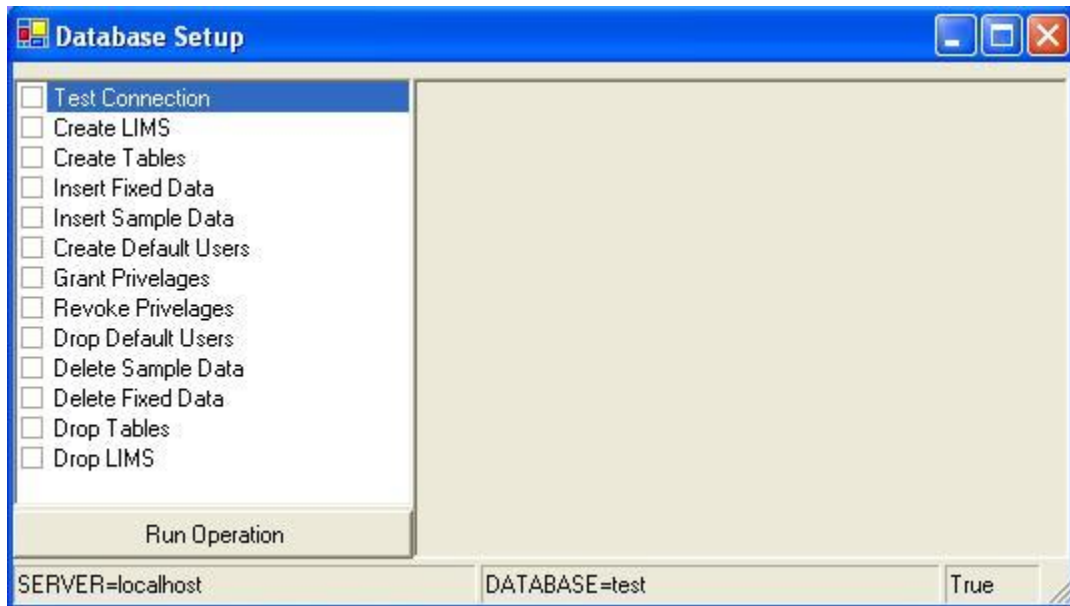
Figure 3.1   LIMSDBMGR Application

Phase I

The first Phase of the LIMS had just five tables developed: *tbldevices*,

tbldata_types, *tbldevice_types*, *tbldata* and *tbltransactions*.   The intention was to create

an application that could link to an outside device and pull data from the device and store

it.   The Device table, *tbldevices*, was created to house the information for the devices

that could be connected to the LIMS.   Descriptions of the fields can be found in Table

A.1.   The name and description fields are used to identify a device within the application

forms.   An example of a device name would be "Thermometer" while the description

would be a lengthier means to identify to the user what the device does.   An example,

12

corresponding to the one above would be "Measures the temperature in Centigrade from -50 to 150".

Each device has at least one, and sometimes more than one, type of chemical trait that it could track, such as pH or the temperature of the solution in Centigrade (C) or Fahrenheit (F). To maintain the atomicity of the devices, the Data Type table, *tbldata_types*, was created and its field schema can be seen in Table A.2. Of special notes is the fourth field in this table which is used to indicate symbol that could be used for a data type such as 'Cº' for Centigrade.

The next step was to combine these two groups of data and associate the devices to one or more data types, thus the Device Type table, *tbldevice_types*, was created; the description of this table can be seen in Table A.3. The intention of the combination is to allow for devices to have one or multiple data types associated with them. With each of these associations, the user also has the ability to set boundaries for the data values including the minimum, maximum and possible variation of the data from the device respectively. These were not included in the Data Type table because these data type values could vary from device to device.

Now that the tables were established to house the information associated to lab instruments, a table had to be constructed to house the most important element: the data. Table A.4 shows the field schema for the Data table, *tbldata*. The table was designed to be grouped using the *batch_id* field with the value field having the user or device entered value.

The final table constructed for Phase I was the transaction table, *tbltransactions*, with six fields described in Table A.5. The purpose of this table is to track all

13

transactions done on the other tables in the database.   The entries are not automatically

generated by the MySQL server, but from code in the LIMS application.   The most

important field is the *sql_text* which stores the SQL syntax used in the transaction.   In the

case that one of the other tables is corrupted, the text in the sql_text field can be used to

restore the data or check for syntactical errors.

The final schema for the first Phase is shown below in Figure 3.2.
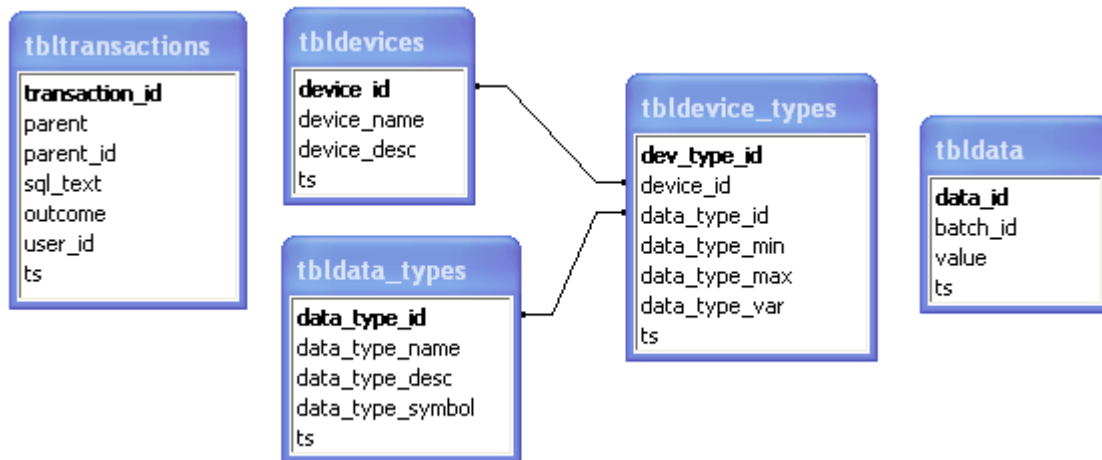
Figure 3.2   Final Phase I Schema

Phase II

Phase II saw the implementation of five more tables: *tblexperiments*, *tblbatches*,

tblexperiment_notes, *tbldata_notes* and *tblnote_types*.   This Phase was designed to create

a more versatile structure to better organize the data, regardless of the source.   The

design was modified to now include groups of batches within experiments.   The

experiments would act as the heart of the data schema, with each experiment having the

potential to have multiple batches, each batch with one or more points of data.   Figure

3.3 shows a graphical representation of the intended schema.

14
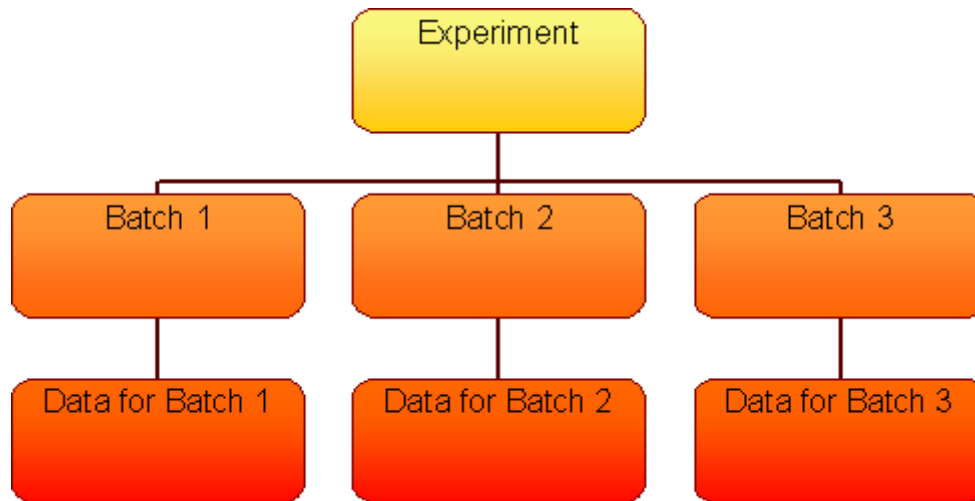
Figure 3.3   Experiment Schema

The Experiment table, *tblexperiments*, was modeled on the tables from the previous Phase.   A more detailed description of the field schema can be seen in Table A.6. At the time that this table was created, the user table still did not exist, but the feature was added with the intention that it would be implemented in Phase III.   The *user_id* would be equivalent to the owner of that particular experiment. That owner could then share out the experiment to other users, but that component is not a feature of this phase and also will be discussed further in Phase III.

With the Experiment table established, the final component to the revised schema shown in Figure 3.2 could be created.   This would be the Batch table, *tblbatches*, comprised of seven fields described in Table A.7.   Originally, the *batch_id* in the Data table was not related to another table and would be incrementally increased using the application component of the LIMS.   Now that the batch table has been created, the batch_id field from the data table is now parented to the batch_id field of the new Batch table.   The Batch table is designed to group data results based on the start and

15

completion of data values being entered either by a user, by an instrument, or by import from an external file. The intention is that a batch can only be started and stopped once. If the user stops a batch and then wishes to continue measuring the same reaction, a new batch will be generated.

In today's academic labs, the standard paper lab notebooks used by the students, are meant to hold more than the quantitative data such as pH or temperature measurements. Sometimes the experiments require detail about the appearance or unusual results that can not be simple described in a numeric fashion.

The notes for an experiment could include a student's questions or conclusions about the experiment data. Two repositories, the Experiment Notes and Data Notes tables, *tblexperiment_notes* and *tbldata_notes* respectively, were created in Phase II to provide a more qualitative structure to the data being assembled. A more detailed description of each table can be found in Table A.8 and Table A.10. The notes could be linked to either an experiment or to an individual data point. Some experiments might simply entail a visual observation taken repeatedly every *x* minutes for a total time of *y*. In this scenario, the time would be the numeric data value while the detailed description would be entered into the data note. Keeping in mind that multiple notes could be applied to either an experiment or a data point, the notes fields were not included in the *tblexperiments* or *tbldata* tables so each of those entities could follow the rules of the Third Normal Form (3NF) for database normalization[15].

Before finalizing the Experiment Notes table, The Note Type table, *tblnote_types* had to be constructed. It only has two fields of concern as described in Table A.9. The *note_type_name* field is used to help better identify experimental notes and to be able to

---

[15] More information on Normalization can be found at http://en.wikipedia.org/wiki/Database_normalization

16

group them for search and reporting purposes. This table was also created to allow for a dynamic source of Experimental Notes. Users will have the ability to add their own Note Types in the future so that they can organize their data according to their own specifications.

The finale table constructed in Phase II was the Experiment Notes table, *tblexperiment_notes*, which uses the *tbldata_notes* table as a model. There are two noticeable differences as can be seen from the field schema in Table A.10. The first is that the foreign key field has changed from *data_id* to *experiment_id* and is related to the primary key of the table *tblexperiments* now. The second difference is the addition of the *note_type_id* field, added as a means to better identify the list of notes for an experiment. This is related to the Note Type table mentioned in the previous paragraph.

The original idea had been to use a note's timestamp to identify it in the application, but this was found to be lacking useful information for a user. Instead a combination of the *note_type_name* from the *tblnote_types* table and the timestamp would be a better representation. This can be seen in Figure 4.8.

The final schema for Phase II is shown below in Figure 3.4.

17
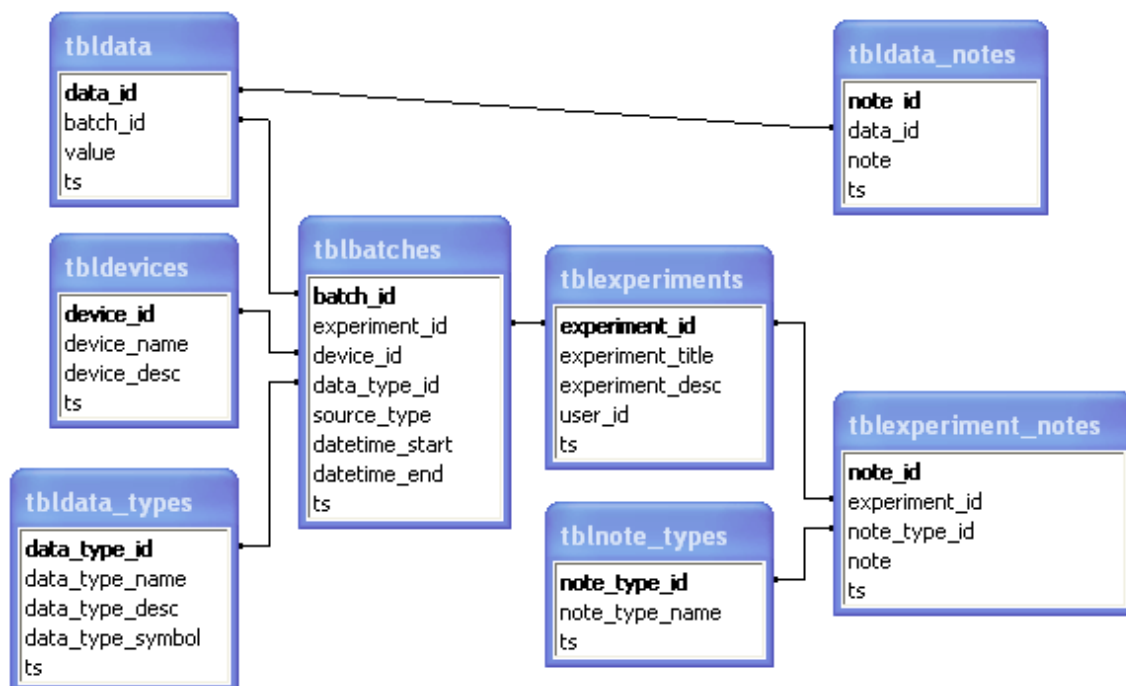
Figure 3.4   Final Phase II Schema

Phase III

The final Phase saw the introduction of the greatest number of new tables, ten in

total, doubling the number of tables overall.   These can be grouped into four categories:

1) Chemicals Tables
2) File Tables
3) Security/User Tables
4) A Sharing Table

These will be discussed in the order listed above.

The first group of tables was designed so that a chemical library could be included

with the LIMS.   Users would be able to indicate what chemicals are used in a batch for

an experiment. They would also be able to add and view traits of these chemicals

including molecular weight, density, the chemical formula, the SMILES formula and

even the 3D model.

18

The first of these tables is the Chemical table, *tblchemicals*, comprised of eight fields as seen in Table A.11. Unlike previous tables, the unique field, *chemical_id* is not auto-incrementing. This is true for all of the tables in the new Phase. I chose to add the new unique id at runtime to test a different model of generating the unique id number. The model has thus far proven successful for this Phase, but it can be converted back to an auto-incrementing type in a future installment if the need arises.

As there are many more traits that a chemical can have than are listed in the Chemical table, and rather than try and fit them into a single table and waste space for chemicals where the information is lacking, a secondary system was designed to handle the remaining traits in a more dynamic means. The first part of this system is the Trait table, *tbltraits*, comprised of four fields described in Table A.12.

Of special note for this table, the *trait_type* field is meant to indicate whether the values associated with the trait should be considered 'NUMERIC' or 'STRING'. The selection of the trait type will determine which of two tables will house the trait's value. An example of a 'NUMERIC' trait would be molecular weight while color would be considered a 'STRING' value. I chose to separate out these two types into two tables, *tblchemical_traits_numeric* and *tblchemical_traits_string* to preserve the values of each type better. A comparison of the fields can be made by comparing the descriptions in Table A.13 and Table A.14. The only difference between the two tables is the type for the *trait_value*: Double for the numeric table and Varchar for the string table.

The final table created for this group, *tblbatch_chemicals*, required only two fields described in Table A.15. Both of these fields are foreign key fields related to the primary keys of the Batch table and Chemical table respectively. No single field was

19

used to uniquely identify a record; rather the two key fields when combined can act as the primary key to uniquely identify a record. The purpose of this table is to relate selected chemicals to a particular batch as chosen by a user.

Taking into consideration that not every aspect of an experiment might be stored locally as data, two tables were designed to store the files (and file types to distinguish the files) created by external applications. These files can include anything from Word documents to PDF or Total Chrom files. The binary stream of the files is copied and stored in the database, identified by a unique HASH value generated by the data in the file at the time it is copied. If that file is changed, the HASH will change as well, allowing multiple versions of a file to be stored at a time. This can allow for a user to observe the progression of a file as well as preserve any and all data in a file even if the external copy is later lost or damaged.

The File Type table, *tblfile_types*, was created first, to be used to distinguish what type of file was being inputted and what application would be used to reopen a file. The field schema is described in Table A.16. The table will allow users to dynamically add file types or update existing file type's extensions for files. The final field in the database, *file_type_image*, is a binary field that stores the image information for the icon associated to the file type, if one exists. This can benefit the user from the GUI side to quickly identify file types by the icon, just as one is used to doing when navigating through windows folders. There is also a default entry made for files with an unknown file type in the rare instance that a type can not be determined. Its unique id is always 0 (zero) and appears to the user as 'Unassociated'.

20

The File table, *tblfiles*, is made up of eight fields, described in Table A17. Of the eight fields, two require some extra explanation. The first is the *file_notes* field which is the only one that does not draw its information directly from the file. It is a field intended to house any information a user wishes to include for a file that can not be gathered from the other fields. It has not been set up like Experiment Notes because there was not an immediate need to differentiate the notes for Files. As well, the volume of files imported, when compared to that of data results, would be minimal enough that a separate table to house the notes would not be needed. The model could be changed in the future if a more dynamic structure is needed or if it is found that fewer users make any notation on the files. The second field of note is the final field of the table, *file_data*. It is a mediumblob[16] field that is capable of storing a file up to the size of 16MB. This type was chosen to both conserve space but still allow for moderately sized files to be included in the database. The next smaller blob field only allows for storage up to 64KB which seems too small for the majority of files today. The largest blob type allows for files up to 4GB, but adding files this size would quickly fill the database or slow the application.

The third group of tables was one of the hardest to design, but the most important set of tables to include. These tables were necessary to allow user functionality and security for the LIMS, an important aspect to any LIMS used today. The first table, *tblaccess_levels*, is concerned with designating the privileges that a user has when working with this LIMS. The field schema for the table is shown in Table A.18, while the default levels are shown in Figure 3.5.

---

[16] Additional Information on the mediumblob type and other MySQL field types can be found at http://dev.mysql.com/doc/refman/4.1/en/storage-requirements.html

21

level_id /	level_title	level_desc	level_options
0	Administrator	Admin - access view and edit most forms - only one with privilege to DELETE records	ALL
1	Manager	Manager - access to view and edit most forms	SELECT, INSERT, UPDATE
2	Lab User	Lab User - limited access to edit some forms and view others	SELECT, INSERT

Figure 3.5   Default Access Levels

At this time, only these three access levels are intended for use with the LIMS.   A user with Administrator level access has the ability to see all of the data and program forms available, but is not allowed to create experiments.   The access level is also intended to be the sole one that can remove records.   An important part of a LIMS is its ability to retain the information it stores even if that information is incorrect and so any removal of the information should be limited.

The next access level is Manager.   This is designed for a single user who is in charge of the local instance of the LIMS in the event that there are multiple users.   A user with the Manager access level can see the experiment and data for all Local Users but can not make changes to it unless authorized by the user.   If there is only one user, then there is no need to create another one with the access level of Local User.   That access level is the lowest and limits the user to seeing only their experiments and data unless another user shares their experiments.   This feature has been mentioned before and will be discussed in more detail in a later section.

The second table included in this group is the User table, *tblusers*.   It is comprised of 6 fields, as shown in Table A.19, used to identify users.   Unlike all other tables so far, the primary key field for this table, *user_id*, does not use an integer value to demark its uniqueness.   Instead the id is generated by a GUID or Globally Unique Identifier. The reason for this is to be able to identify a user when they copy or move their data from one LIMS server to another.   If every machine used an auto-incrementing

22

integer to generate the user ids then a conflict would arise when two users, A & B, both with the same ID but coming from different servers both tried to move any or all of their data to a shared third server. To virtually eliminate this problem, the GUID was chosen instead. "While each generated GUID is not guaranteed to be unique, the total number of unique keys (2^{122} or 5.3×10^{36}) is so large that the probability of the same number being generated twice is very small".[17]

The purpose of the final table created in Phase III, *tblexperiment_sharing*, was to allow users the ability to share their experiments with other users. The field schema, comprised of three fields, can be seen in Table A.20. Like the Batch Chemical table, there is no singular unique key. The uniqueness is generated by combining the two fields: *experiment_id* and *shared_user_id*. These fields are foreign keys related to the primary keys of the Experiment table and User table respectively. The experiment being shared is identified by the experiment_id and the user being allowed access to the experiment is identified by the shared_user_id. The final field, *share_type_id*, describes the level of sharing allowed to the user. There are three levels allowed:

1) 0 – indicates that the user is denied any access to the experiment; this is equivalent to not having a record in the table

2) 1 – indicates that the user is allowed to the view the experiment but not make any edits

3) 2 – indicates that the user is allowed to both view and edit the experiment

The whole purpose of this sharing was to allow more than one user to collaborate on an experiment as can often be the case in an academic laboratory. Allowing a user to also set what privileges another user has with their experiment was important to prevent

---

[17] http://en.wikipedia.org/wiki/Globally_Unique_Identifier

23

unauthorized changes but still allow some latitude with the sharing of data, also an important aspect to any LIMS.

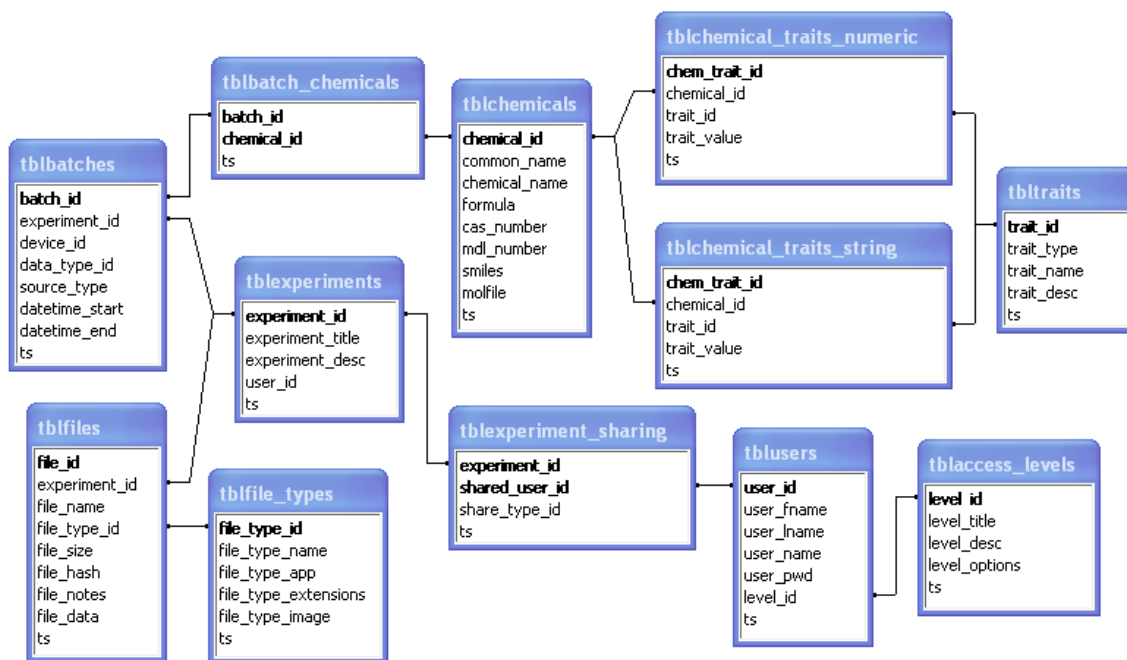The final schema for the first Phase is shown below in Figure 3.6.

Figure 3.6   Final Phase III Schema

CHAPTER FOUR: APPLICATION DEVELOPMENT

Just as the database tables were developed over three phases, so was the VB.NET application and the forms associated with it.   Rather than separate the results into three Phases again, each form will be discussed individually as it exists in its current state. The progression will follow the order in which a user might come across the forms when working with the LIMS application component of this project.   As there is so much code behind each form, the focus will only be on the functions of the forms and details of the controls viewable to a user.

24

Before the application development was started, the means by which the

application would connect to the database had to be fixed.   The SqlClient that is part of

the System.Data Namespace in VB.NET is intended for use with SQL Server, not

MySQL.   The MySQL connection actually requires a special ODBC driver, mysql-

connector-odbc, which can be downloaded from the MySQL website

(http://dev.mysql.com/downloads/connector/odbc/3.51.html). The most recent version is

3.51.16., but the versions 3.51.12 and 3.51.1 were used for the development of the cycle

of this application.   The drivers were then imported and referenced into the application

as Microsoft.Data.Odbc.   If this Namespace was not imported to a Form, the data

connection would error out.

Login Form

The first screen that any user will see is the Login screen.   This form retrieves a

list of users from the database and populates the User drop down control.   Once a user

selects a name from the list, presumably their own, they enter their password which is

disguised by "*" character.   Clicking the Login button will verify if the password entered

is the same as is stored in the database.   If the entry is incorrect, the password field is

cleared so the user may retype the password.   If the entry is correct then the window

closes and brings the user to the Main navigation window, seen in Figure 4.1

This window has a set of Menus that access all other available forms.   The majority of these forms open within the Main navigation window, just as this document does within Microsoft Word.   A select few open into their own window because they are meant to only be used for a short time period.

Figure 4.1   Main Navigation Form

The list of menu items and what each one does is listed below in Table 4.1.   The primary menus are highlighted in blue in the pictures in the right hand column, with the sub-menu items listed below them.
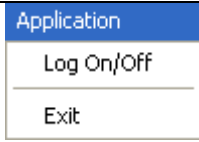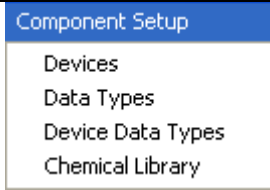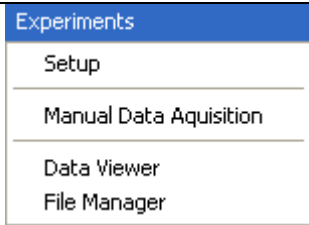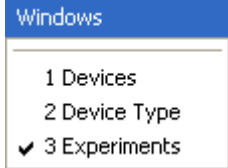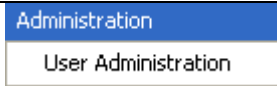
26

Menu Descriptions	Menu Appearance
1) Log On/Off – opens the Login Screen to login a different user 2) Exit – Closes the application	**Application** Log On/Off Exit
1) Devices – Opens the Device management form 2) Data Types – Opens the Data Type management form 3) Device Data Types – Opens the Device/Data Type management form 4) Chemical Library – Opens the Chemical management form	**Component Setup** Devices Data Types Device Data Types Chemical Library
1) Setup – Opens the Experiment management form 2) Manual Data Acquisition – Opens the Data form for entering data manually 3) Data Viewer – Opens the data viewing form 4) Files – Opens the File management form	**Experiments** Setup Manual Data Aquisition Data Viewer File Manager
Experiment Results – Opens the form that can show the Crystal reports created for the application	**Reports** Experiment Results
Shows the windows that have been opened and which one is currently the top level window. A user can change active windows by selecting from these choices or by selecting the original menu that opened the form	**Windows** 1 Devices 2 Device Type ✓ 3 Experiments
Opens the User management form	**Administration** User Administration

Table 4.1   Application Menu Structure and Explanation

As the Login form has already been discussed, those details do not need to be discussed

again so I will move onto the forms that are part of the Component Setup menu.

Component Menu Forms

The first form available under the Component Setup menu is the form to manage

Devices seen in Figure 4.2.   The data for this form is drawn from the *tbldevices* table

discussed earlier.   The layout used in this form became the model for the remainder of

the management forms.   It consists of two Panel controls split by a vertical Splitter

control.   The left panel has a **ListBox** control that lists all available devices, with a count

shown in the Label above the list.   When a user selects one of these devices, the

information is entered into the **TextBox** and **Label** fields in the right panel.   These

values can not be edited unless the user clicks either on the New or Edit buttons.
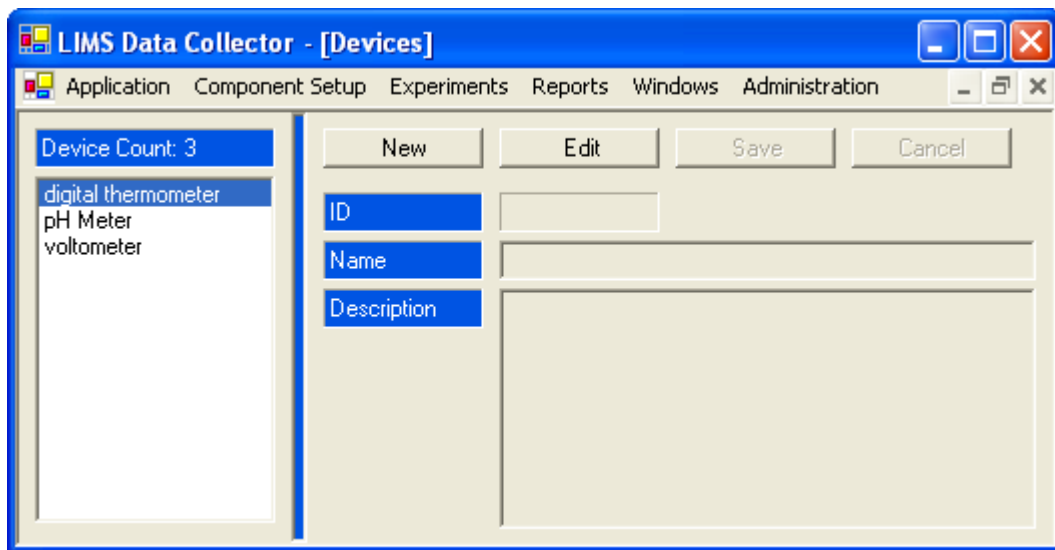
Figure 4.2   Device Management Form

Clicking on the **New** button will clear the fields and allow the user to enter

information for a new device, the type of information denoted by the Label controls.   In

the case of this form that would the Name and Description to give to the device.

Clicking on the **Edit** button will not clear the fields, but give the user access to change

the information currently present in the control fields.   When the information changes

28

have been completed the user can either click the **Save** button to insert or update the data or they can chose to **Cancel** their work, thereby clearing and resetting the information.

Figure 4.3 and Figure 4.4 show the next two available forms: the Data Type management form and the Device/Data Type management form. Their structures and usage are the same as the Device form, but their functions are designed to manage the Data Type and Device/Data Type tables. The only significant difference is on the Device/Data Type form where the list of Device/Data Types is not shown as a list on the left side, but in a TreeNode. This form was chosen since some devices could have more than one data type as seen by the "digital thermometer" entry in Figure 4.4. Rather than duplicate the device names in the list, they were singled out and parented, while all data types related to the device were setup as the child controls. Selecting the data types on the Device/Data Type form will still populate the control fields on the right just as it was done with the first two forms.
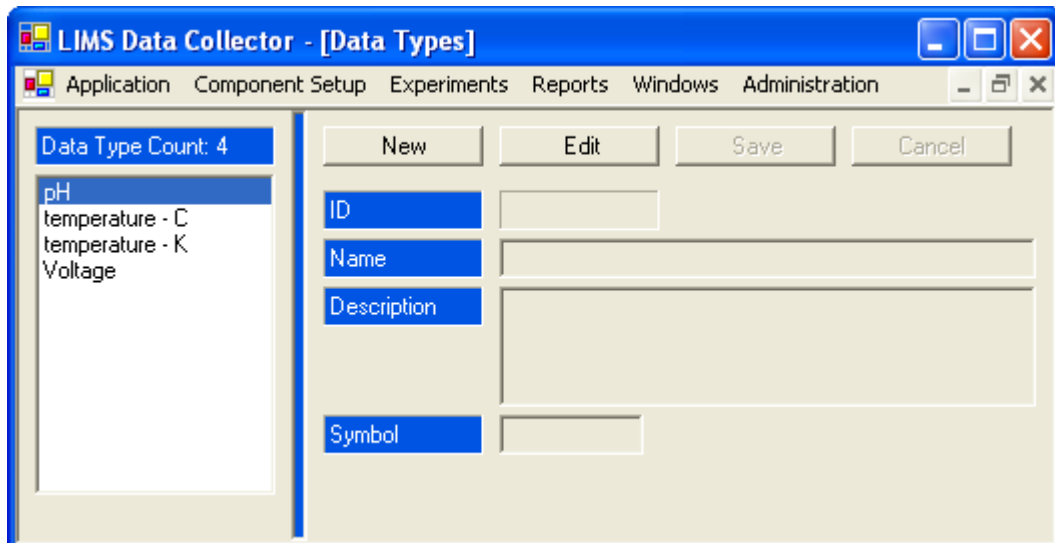

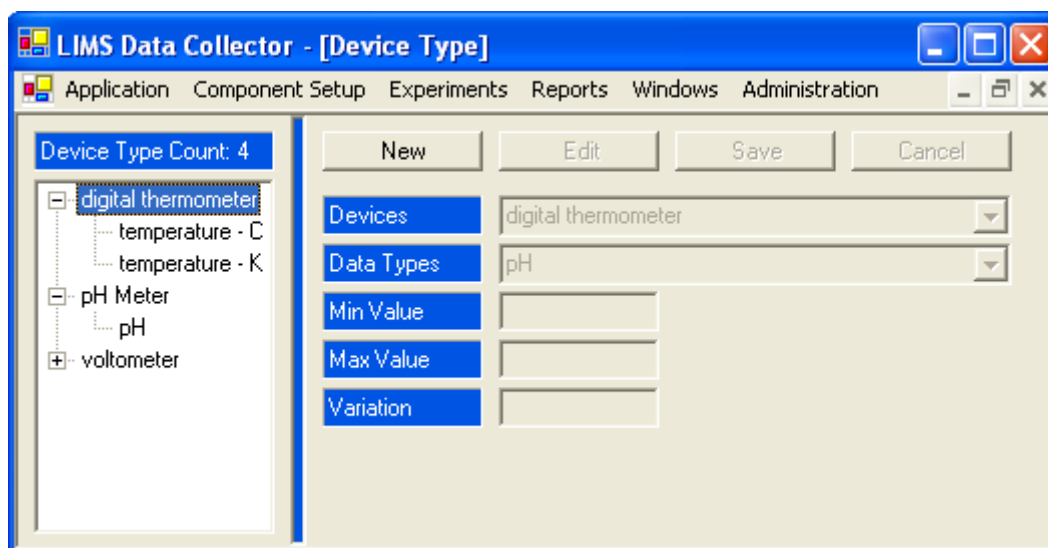Figure 4.3   Data Type Management Form

Figure 4.4   Device/Data Type Management Form

The final form access by the Component Setup menu opens the Chemical Library form which is used to manage the Chemical tables in the database.   Figure 4.5 shows the layout of the form, including a new section in the lower part of the right panel and three new buttons: **View Info**, **View JME** and "**…**".

The new set of controls in the bottom third of the right panel are like a mini version of the other controls.   As described in the table development section, two tables were created to house a dynamic set of chemical traits.   Rather than break out the chemical/trait combinations, as was done in the Device/Data Type form (refer back to Figure 4.4) the new group of controls was designed to show the list of the traits created and their respective values.   The **New**, **Edit**, **Save** and **Cancel** buttons all work under the same guidelines as the primary set at the top of the right panel.   This mini-group was designed to be a more effective means to view and manage the traits for a chemical.

30

Figure 4.5   Chemical Library Form

The **View Info** button opens a stand-alone window shown in Figure 4.6, titled

Chemical Information.   It's designed to quickly show all of the chemical information for

a single chemical, including the dynamic list of trait values as well as the 2D/3D structure

of the molecule.   This latter ability was managed by adding a **WebBrowser** control to

the right panel of the form which linked to a MDL mol file, if one exists, and using the

CHIME plugin available from MDL for free (www.mdl.com/support/developer/chime/).

Normally this has been a tool used solely in a commercial browser such as IE or Mozilla.
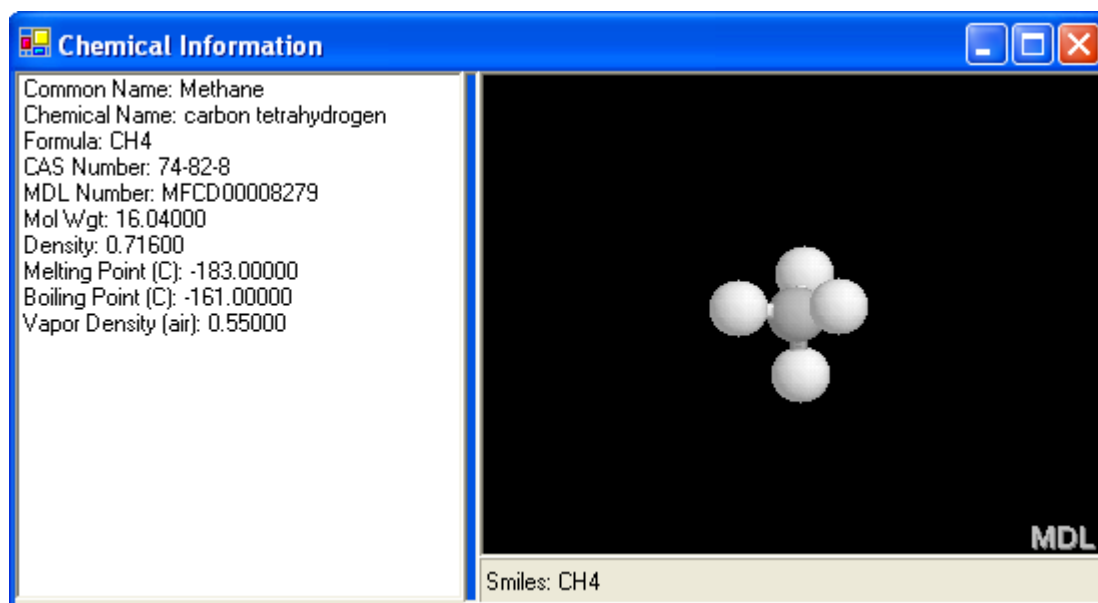
31

Figure 4.6   Chemical Information Form

In the event that a chemical has an existing MDL mol file, the second new button, titled **View JME**, is enabled and opens another stand-alone form that also employs a **WebBrowser** control. This time the control accesses an HTML page that employs the JME Molecular Editor as shown in Figure 4.7.

This page shows the 2D structure of the molecule in the JME molecular editor applet created by Peter Ertl (http://www.molinspiration.com/jme/). The only requirement to get the tool was to email him with the purpose of the project and how his applet would be used. Once the files were emailed back and installed the only requirement was to include the text "JME Editor courtesy of Peter Ertl, Novartis" in the HTML page as has been down below the applet itself. The JME applet is a useful to create and edit molecular structures as well as generate a chemical's SMILE formula and the MDL mol file. This form is currently limited to only viewing the files.

32
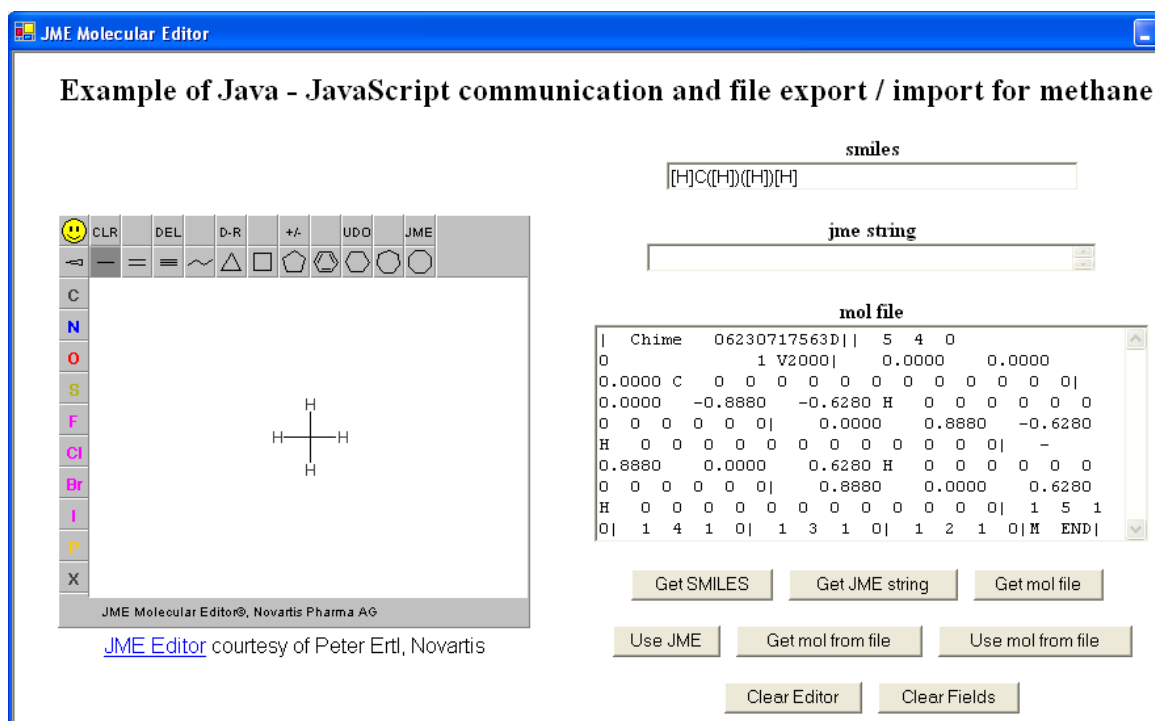
Figure 4.7   Java Molecular Editor

Experiment Menu Forms

The first and foremost form available through this menu is the form to manage the user's experiments as shown in Figure 4.8.   This form is designed to manage the experiment table and is modeled like the previous management forms, including the mini-group to handle the experimental notes that a user can create.   In addition, there are two new buttons: **Files** and **Share**.

33

Figure 4.8   Experiment Management Form

The **Files** button opens the File Manager window, a stand-alone form shown in Figure 4.9.   This form works with the Files tables to add files chosen by the user to the database.   If the user has the privilege to 'Edit' an Experiment then the **Add File** button will be enabled for them. If, however, they only have the 'View' privilege, then they will only be able to see the file list and view the files.

The process to add files to an experiment begins by clicking on the **Add File** button.   If the **Preview File First** Checkbox control is checked then the file will be opened in its native application first to make sure the correct file is being saved. The file can then be accepted (or this process skipped by unchecking the **CheckBox** control) and the Saving File form, Figure 4.10, will open, showing the progress of the file being saved. Upon completion of a successful save to the database the list of files in the File Manager form will be update to reflect the additional file.
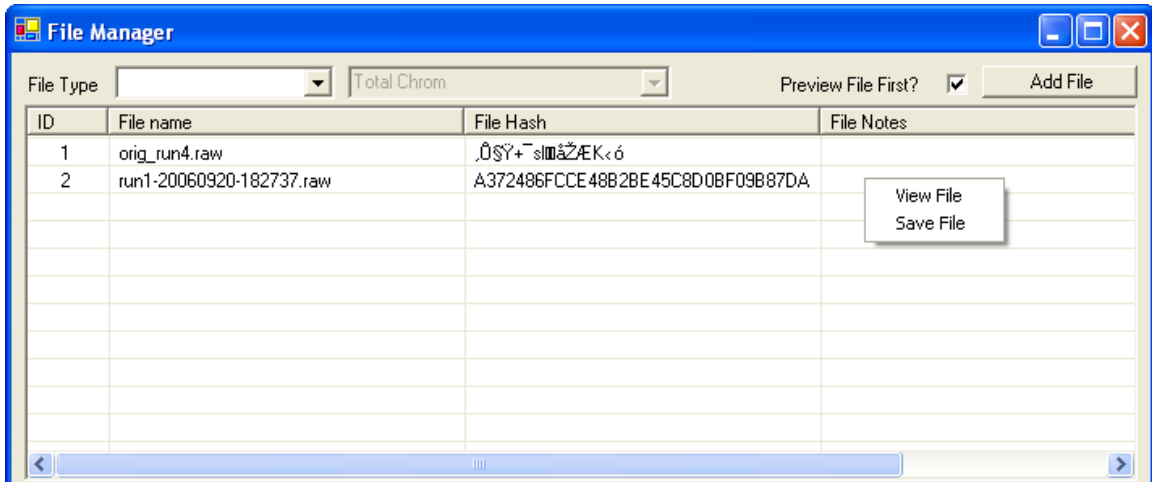
Figure 4.9   File Manager

.
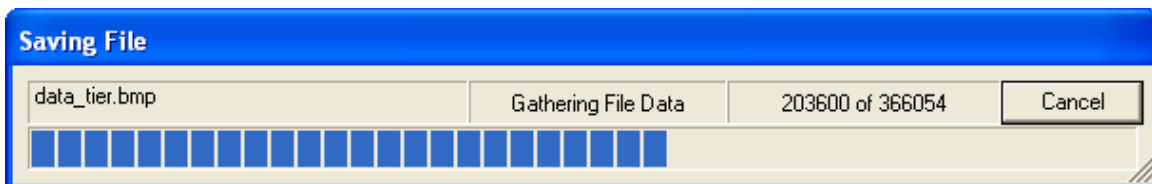
Figure 4.10   Save File Form

Examining Figure 4.9 will also show a **PopUp** menu with two options on it: **View File** and **Save File**.   The **View File**, when selected will load the data from the *file_data* field of the *tblfiles* table into a new external file and proceed to open the file in its native application.   The **Save File** option will allow the user to save any changes made to the File Note field that is part of the **ListView** control.

Returning to the Experiment management form, Figure 4.8, there is also the **Share** button.   Clicking this button will take the user to the Experiment Sharing form shown in Figure 4.11.   This form provides a powerful option to the user to share his/her

35

experiments with other users.   They have the option to choose the user as well as the
level of the sharing.
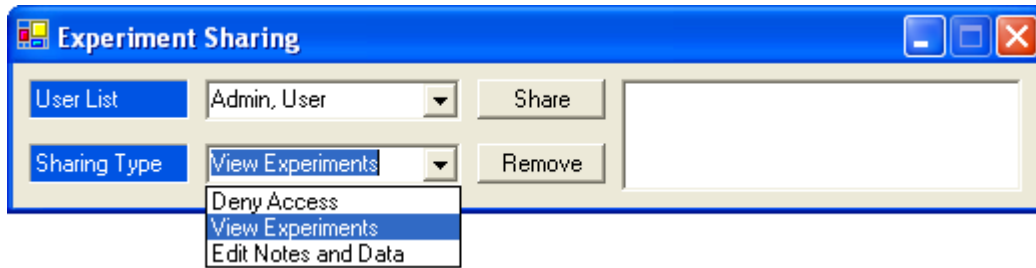
Figure 4.11   Experiment Sharing Form

The second form available in the Experiment Menu is the form to collect data
manually.   As one can see from Figure 4.12, the design of this form differs greatly from
the other management forms seen thus far.
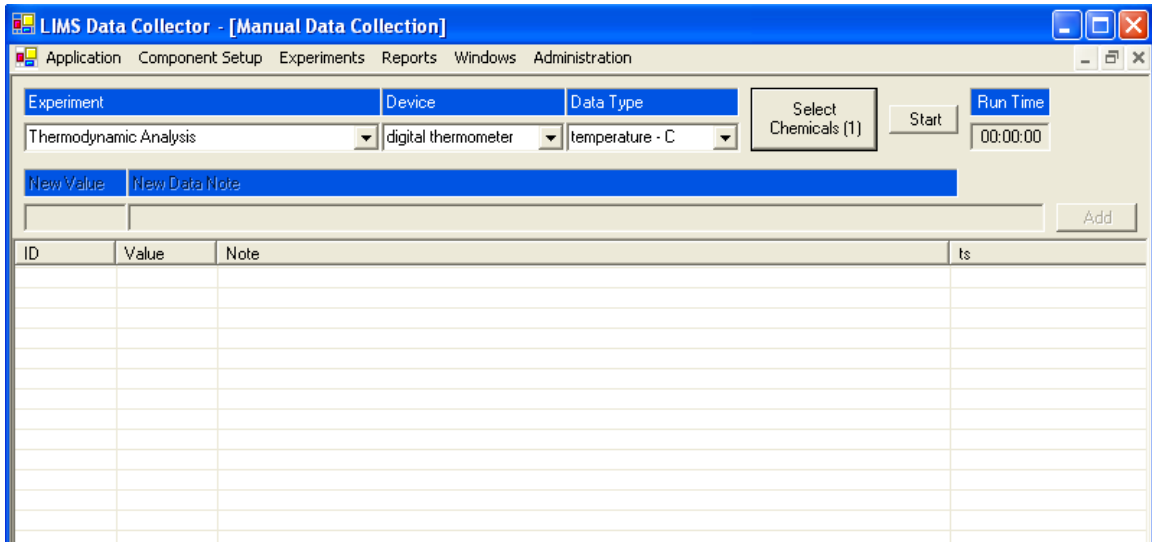
Figure 4.12   Data Management Form Example 1

A user must select one of his/her experiments, a device for the experiment and a
single data type for the experiment. If the user has a device that can gather more than one

data type at a time and both of these data types are desired by the user, then an additional Data management window will be opened for each additional data type thereby creating a batch for each data type.

The last step before starting the run involves selecting all of the chemicals being used in this part of the experiment.   This is done using the Batch Chemical Selection form seen in Figure 4.13. Addition or subtraction of chemicals to the "Selected Chemicals" list is done by double clicking on a chemical in either list.   The **Clear** button can also be used to quickly clear the Select Chemicals list of any entries.   When the user has chosen all of the chemicals, they simply click on the **Close** button.
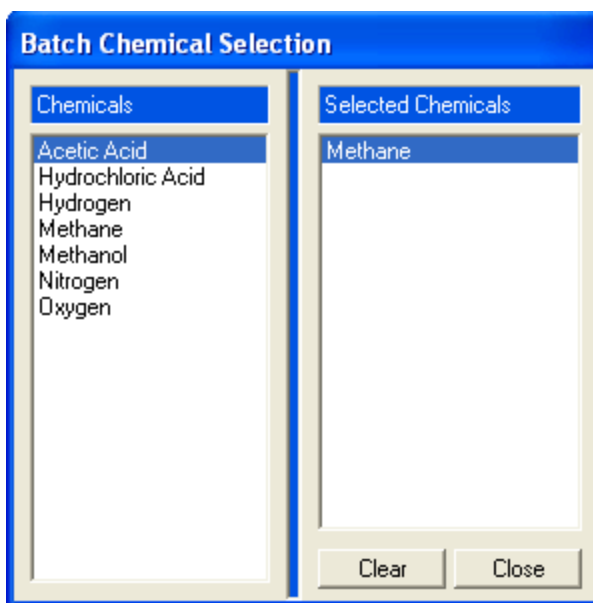
Figure 4.13   Batch Chemical Selection

The final step to instantiate the batch is to click on the **Start** button.   This will start a timer, seen to the right of button and enable the row of controls below for the user to enter a **New Value** and/or **New Data Note**.   To expedite the entry, the **Enter** key has been linked to the **Add** button so clicking the key will be equivalent to clicking on the

37

button.   The focus is then automatically shifted back to the **New Value** field.   This can save time for the user so they don't need to fuss with a mouse to move back and forth between the fields.   As the data is entered, it is displayed in the list controls below the data entry controls.   The data are ordered by the ID that is automatically generated and an example of this can be seen in Figure 4.14.

When the user no longer needs to enter data, they can click on the **Stop** button and end the batch sequence.   Clicking the **Start** button again without making any other changes will simply create a new batch group.

Figure 4.14   Data Management Form Example 2

Now that the data has been gathered, the user needs a means to view that data at a later time.   That is the purpose of the Data Viewer form illustrated in Figure 4.15. A user is able to look at all of his/her experiments and those that have been shared with them by other users.   A user can select more than one experiment and more than one batch at a time and look at the data together.

Figure 4.15   Data Viewing Form

The final form available in the Experiment Menu is for the File Manager form. This form was discussed earlier (Figure 4.9) when the Experiment Management form was being detailed.   This form instance only differs in one way from the earlier discussion. The File Manager form started from the Experiment management form was limited to only the Experiment chosen at the time that the form was open.   The File Manager, when opened form the Experiment menu now allows the user to select from any of the reports to which he/she has access.

Report Menu Forms

The Data Viewer was useful for looking at past data from within the application, but sometimes it is necessary to present that data externally.   One of the means of doing this is with reports.   This is the purpose of the Experimental Results Form which has access to a Crystal Report designed to go with this application.   Figure 4.16 shows an

39

example of the report being displayed after a user has selected an experiment and batch to be linked to the report. Changing either will refresh the reports underlying data. The **Crystal Report** also has the benefit of being able to print the report or export the data in the report to PDF or excel formats. There will be a more in depth explanation of the crystal reports used in the application in the Third Party Integration section.
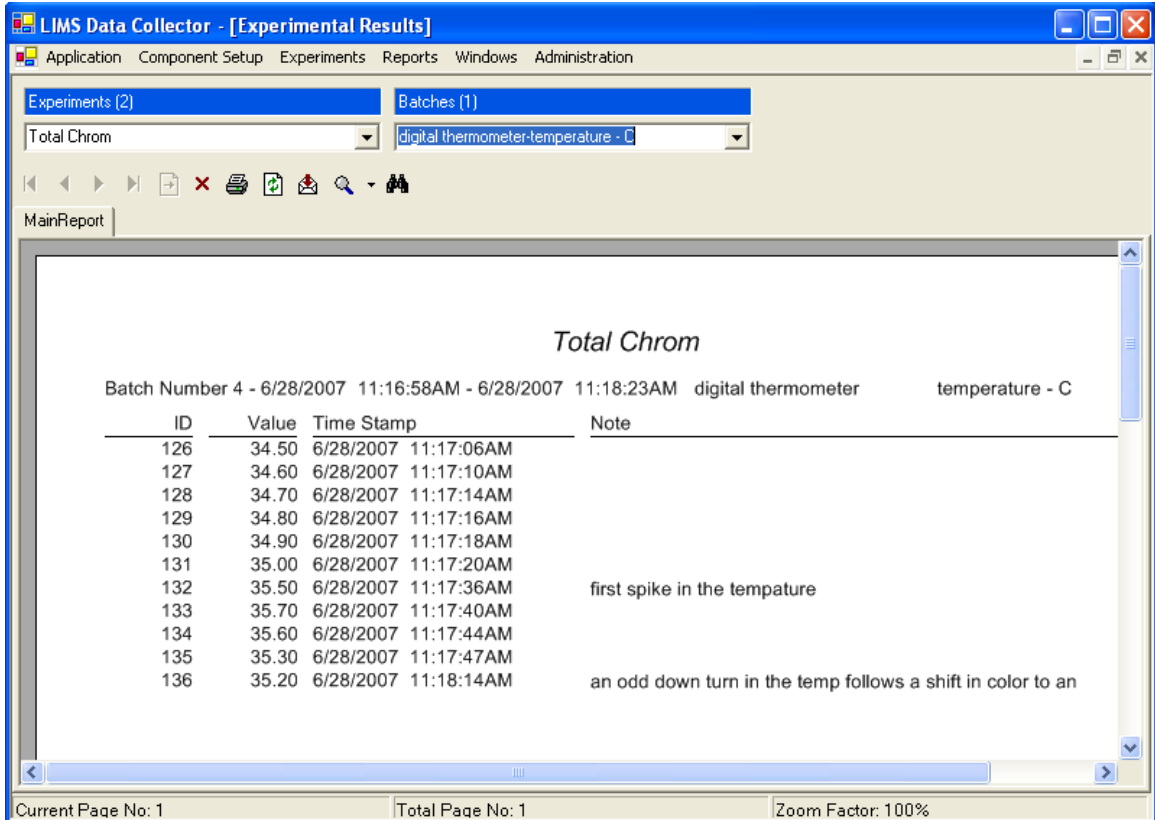
Figure 4.16 Experimental Results Form

Before the Crystal Reports component was added, the reporting was done by converting the data into a Rich Text Format, RTF, which was displayed in a **RichTextBox** control. The layout of a report was far more difficult and time consuming than with a Crystal Report. It would also have to be done in some sort of modular form so that other reports could be added in the future. With Crystal Reports, the creation of

reports is far easier and much more powerful and dynamic than the RTF. A user could now add in graphics such as images or graphs which was not possible at all in RTF.

<div align="center">Administration Menu Forms</div>

The finale form constructed and available deals with the users and security for the application. The User Administration form, illustrated in Figure 4.17 returns to the original model for the design of the layout. The list on the left is restricted, depending on the user's current access level. If they are only a **User** then they will only see their own record and they will only be able to change their name and password. **Managers** can see all of the other users, including the **Administrator**, but a **Manager** can not edit the **Administrator's** record, nor can he change his or any other user's access level. A **Manager** can create new users, but they are limited to the role of **User**, and a **Manager** can edit the name and password for a user. Only an **Administrator** can create a user with the access level of **Manager**, which is role limited to one user per machine. The majority of the time the **Manager** and the **Administrator** are likely to be one in the same, but this scenario allows some versatility in the rare case that they are different persons.

41
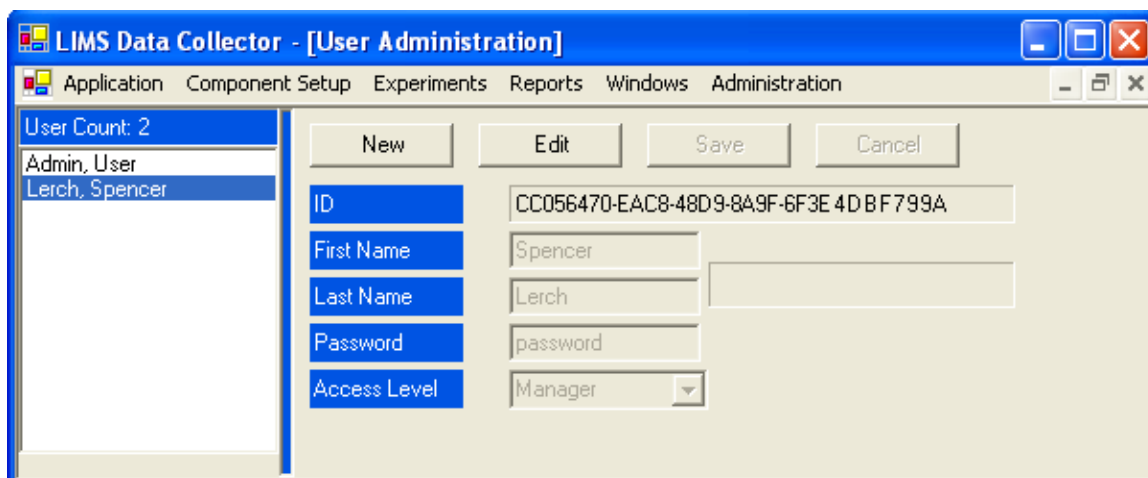
Figure 4.17   User Administration Form

Third Party Integration

In the previous section, the integration of two 3rd party applications was discussed, these being the JME Molecular editor and the CHIME plugin.   Both of these were easily integrated into the LIMS and can be modified by future users readily.   The crystal reports are not quite as easy to work with, but they are a very powerful tool for the application's reporting needs.   To create any crystal report, a developer must either have access to one of the Business Solutions development kits, such as Crystal Reports 9, 10 or XI, or have Visual Studio with a .NET control used to create the reports.

Since the application was developed using VS 2003, the choice was made to use the .NET crystal report development tools.   A separate project was created and a new crystal report object was added by going to the **Project Menu** and selecting the sub menu 'Add New Item…' as shown in Figure 4.18.   This opened the window shown in Figure 4.19 where a developer can select the Crystal Report Object.

42
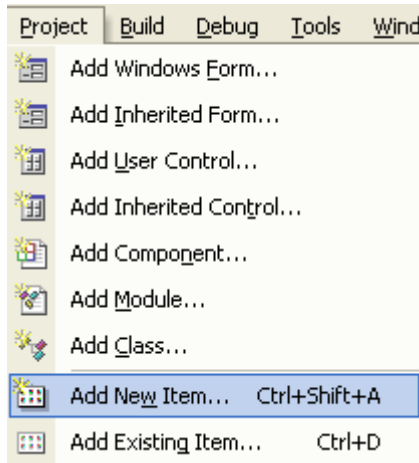
Figure 4.18   Add New Item Menu
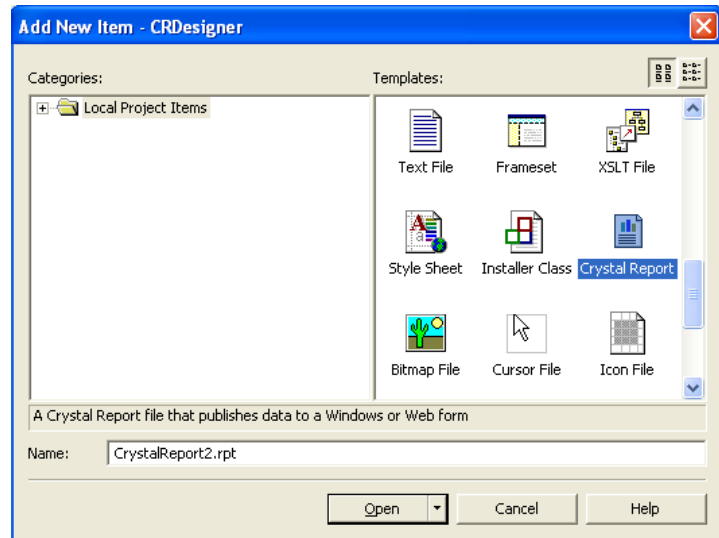
Figure 4.19   Adding a Crystal Report Object

Before the crystal report object could connect to the database, a System DSN had

to be created.   The details of how to make one are not necessary here, but the values

entered for the DNS are shown in Figure 4.20.
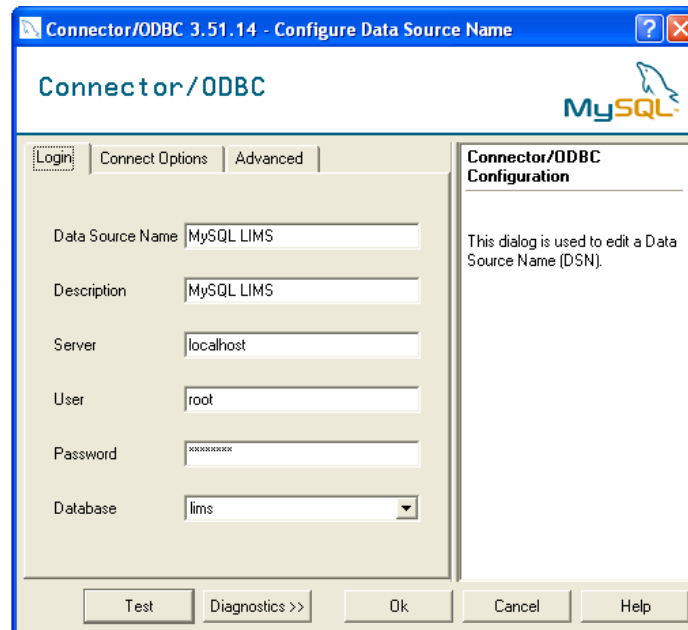
Figure 4.20   System DNS Entry for MySQL Database

43

With this DNS established, a user could now connect to the database through the

ODBC object as illustrated in Figure 4.21.
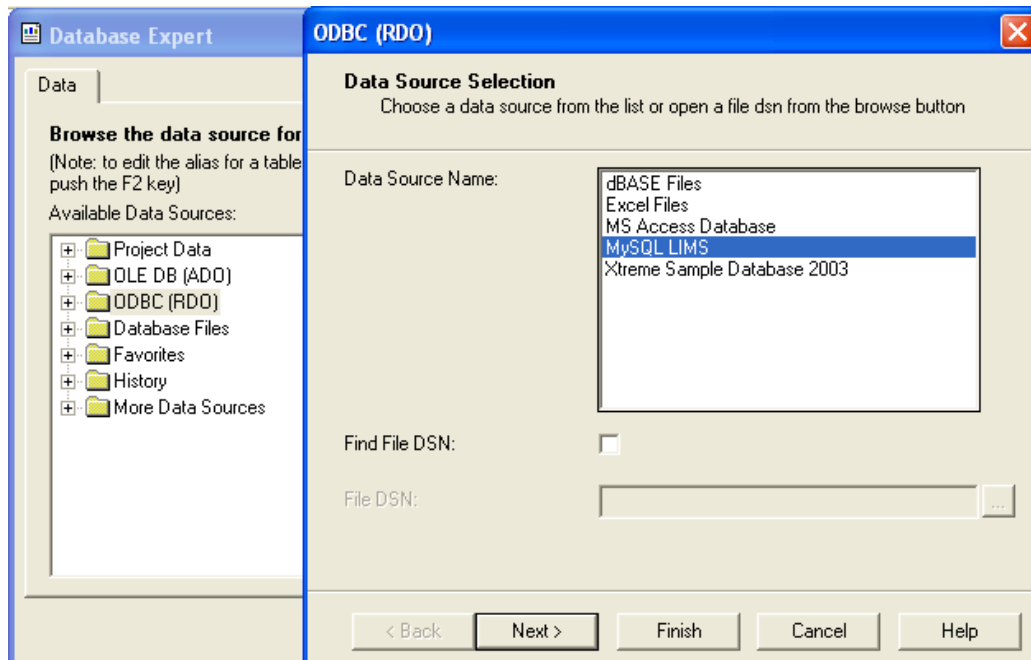
Figure 4.21   ODBC DNS to the MySQL DNS
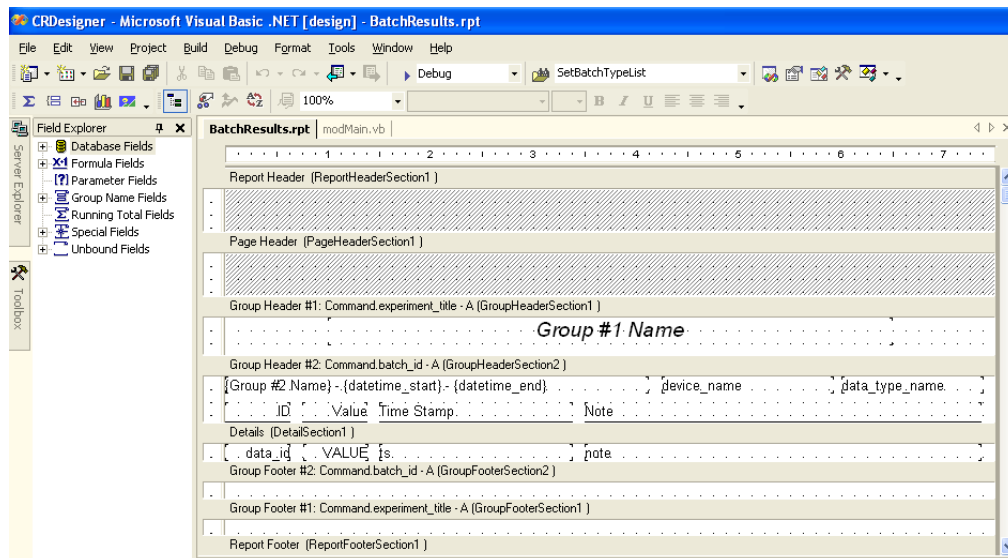
Figure 4.22   Crystal Report Designer in Visual Studio 2003

44

The final step is for the user to create a Command query and layout the fields from the query. The model of the report created for this application is illustrated in Figure 4.22. The current report in the application is hard coded into the Experimental Results form, but this can be modified in the future for the user to have a more dynamic access to the reports created for the LIMS.

With the ability to enter, store and retrieve data from the LIMS database, the next step was to be able to connect it to an external program that could analyze the data. Spotfire would normally be a logical choice, but since not every lab will have access to that application, I wanted to go with one that would be free available. I chose WEKA because it is free to use and provides some of the same tools and functionality.

WEKA took some work to get everything working. The application was easily downloaded and installed, but trouble occurred when trying to get it to connect to the MySQL database. It was determined that an adapter had to be downloaded which would allow WEKA to work with the database. After some searching on the internet, the most recent version was found, mysql-connector-java-5.0.4-bin.jar on the MySQL website. It was installed and a modification was made to the CLASSPATH of the computer to include this file with Java applications. During the internet search for the file he format for the database connection string was also located: jdbc:mysql://localhost:3306/database. Rewording it to be specific to the database, jdbc:mysql://localhost:3306/lims, a connection was established.

However, more trouble occurred with some unrecognized data format errors. The first error involved the unsigned integer format that was used for many of the unique/primary key fields. Adjustments to the sql fields used, changing the unsigned

integers to normal signed integer fields solved the problem.   A second type format error

occurred, this time for the datetime fields used for the timestamp fields in every table.   A

further literature search indicated how this could be fixed by modifying the

Databaseutils.proc file included with WEKA.   The file was initially compressed into the

weka.jar file so it first had to be extracted.   Once extracted, the unknown data types for

the datetime were added and a separate field was changed to default the database

connection string to jdbc:mysql://localhost:3306/database.   With all of these

modifications, WEKA was now able to login to the database and extract data from the

table, tbldata.   Figure 4.23 shows the visualized plot comparing my data values to their

data_ids.
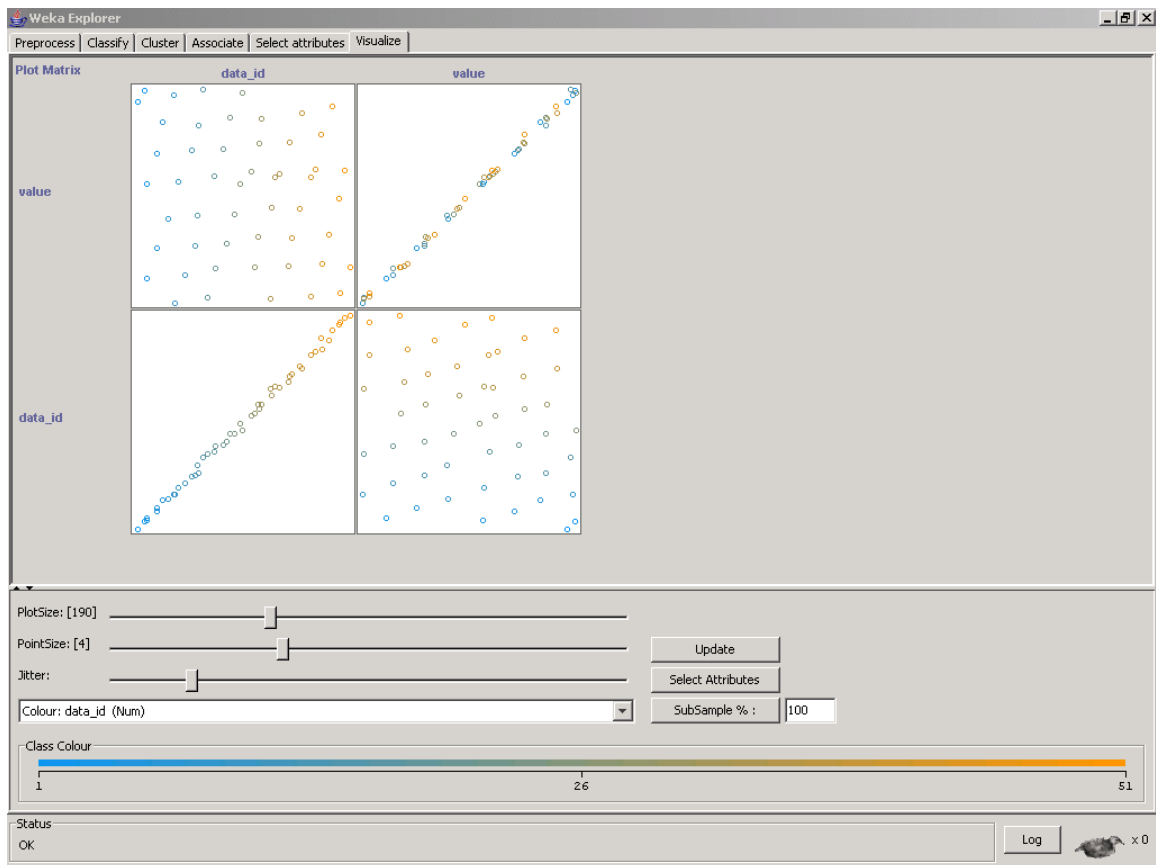
Figure 4.23   WEKA Plots

46

CHAPTER FIVE: CONCLUSIONS

Discussion

This project addressed two problems really: the feasibility of an academic LIMS and what the thought process behind designing such an application would be. Regarding the feasibility, there were questions about how the data could be stored and accessed, what kind of interfaces will work and what 3rd Party tools could be used and could it be done at a low cost.

The culmination of all the work has produced a working LIMS that has numerous capabilities but some shortfalls too. What started as a small little application, to add data to a database from a virtual device, has grown into a more robust application that includes reporting and file management tools. The VB.NET programming language and the MySQL database have proven to be a useful combination to create the initial platform for the LIMS. It gives the users the ability to store and view chemical data in a more informative fashion with the help of the 3rd party programs such as CHIME and JME Molecular Editor. They were easily implemented into the application, either directly or via HTML documents that are also easy to create and edit.

The project has also provided me with a great deal of knowledge from a programming perspective on how much time is required to put together even a single component. Like other non-LIMS projects, as a developer I needed to first determine the needs of the user. These needs include what the user needs to accomplish and how the user would normally get to the end results. Then the foundation could be built starting with the database tables where the data is to be stored. Once that has been finalized, the

47

front end GUI is constructed around the data structure. Then the component could be reevaluated to determine what functions could be changed or what other functions could be added to make the component more productive.

Finally, other than time and sweat, it cost nothing to produce this LIMS. Everything that I used was freely downloadable from the internet.   The Visual Studio development kit from the IUWARE website, MySQL from www.mysql.org, the CHIME plugin is available at from www.mdl.com, the JME can be obtained by contacting Perter Ertl, and the WEKA application is available at www.cs.waikato.ac.nz/ml/weka.   Because all of these components are free, there is no reason why the application, and how it was developed, can not be shared freely to other academic users.

Limitations

The biggest limitation for this LIMS is its ability to import data from external sources.   There is no SerialPort object in .NET 1.x which puts a huge limitation on its ability to connect directly to external devices.   There is a SerialPort object class in .NET 2.0 so future modifications could be made to transition the application to that framework.

During the course of the different Phases of this project, more than one attempt was made to read and translate external data files, generated by the TotalChrom software. The ability to read the files was easy, but not the ability to distinguish between matching meta-data values or pick out the true data values.   The first attempts to modify the files involved substituting single characters in the ASCII readable portions.   This was done using notepad to read, write and resave the file.   However, this only proved to corrupt the files when they were reread by the TotalChrom applications.   Analysis of the converted

48

files, led to the discovery that Notepad had converted particular byte values to a value unreadable by the TotalChrom applications.

An alternative method was found that resulted in the successful modification of a .raw file to one that was still readable by the TotalChrom applications, and the matching ASCII fields were different enough to identify them individually now. There are, however, too many unknown variables within the files to determine the full order and to what the byte data between the fields corresponds.

Another limitation is the ablitity for the application to pull the mol value, generated by the JME component, into the database. The limitation resides in the ability to save the value generated using the JME form to a file. Javascript can save the information to a file, but this can error out depending on the restrictions set on the windows machine. If a technique can be established to get around the restrictions, this limitation can be easily overcome.

A minor limitation with the file management component is the ability to automatically resave a file that has been opened from the File Manager form, modified, and saved. A user currently would have to import the newly saved file just as was done with the original copy. By automatically saving the file, fewer steps will have to be taken by the user and it will ensure a more accurate recording of the file versions.

Future Research

New ideas were constantly being generated on how to present the data or manage all of the work behind the scenes. Components that were in a "finalized" stage could suddenly seem obsolete and would have to be reworked. During the development

49

process, numerous ideas for modifications evolved from the challenges of the application development and what could be done with the data. One of the easiest challenges to meet will be the need to develop more reports based on the data in the database. The tool has been created to make the reports, so a means to dynamically get to the reports, a report manager of sorts, could be developed to do this.

As mentioned in the Limitations section, there is no SerialPort object in the .NET framework used to build this application. There are couple of work arounds to this including migrating everything to the .NET 2.0 framework where there is a SerialPort object. This could greatly expand the capabilities of the LIMS. Furthermore, redesigning the application to make it capable of accepting 3rd party plugins or modules, such as ones for data importation, could potentially create a more powerful and dynamic application.

The next biggest step in the LIMS' evolution, though, would be to establish a multi system environment linked to a central storage system. Data could be passed back and forth between users within a lab environment or between labs even. The user_id is currently designed to ensure that two users do not have the same id regardless of when or where the user record is created. The intention is to use a combination of the user_id and the primary data value of a record to create a new dual primary key in the shared data repository. That way the data can still be linked to a single user while simultaneously allowing for multiple user's data to be compared.

Before any of this can be used on a new user's computer, the installation procedures and the LIMSDBMGR will have to be finalized. The installation procedures will have to include parameters for the LIMSDBMGR to detect if the LIMS is already

50

installed, that ways, existing data is not accidentally overwritten.   The user will also have to have the option of selecting the address of the MySQL database, whether that is localhost or one that is hosted on another machine.   That information will then have to be saved and accessible to the new LIMS installation.   The stored information can be done using the registry or some sort of file store in a static location.   The latter method would be more useful for when a future version is ported to another operating environment such as Max OSX or Linux.

Implications of Results

So why then are there not more academic LIMS out there?   Perhaps there are and they are just not being shared and perhaps there really is just a void.   I used just one computer programming language to produce the LIMS, but that can be converted to other languages and platforms such as ASP.NET or php.   There certainly is no lack of tools that can be used to develop a LIMS, so perhaps there is a lack of persons able to do it or a lack of necessity.

APPENDIX A: ACADEMIC LIMS TABLE SCHEMA

tbldevices			
Column	**Type**	**Characteristics**	
device_id	smallint(5) (Auto)	Not Null	Primary key for table
device_name	varchar(50)	Not Null	Name of device
device_desc	varchar(500)	Not Null	Description of device
ts	datetime		timestamp

Table A.1   Device Table

tbldata_types			
Column	**Type**	**Characteristics**	
data_type_id	smallint(5) (Auto)	Not Null	Primary key for table
data_type_name	varchar(50)	Not Null	Name of data type
data_type_desc	varchar(100)	Not Null	Description of data type
data_type_symbol	varchar(5)	Not Null	Symbol for data type
ts	datetime		timestamp

Table A.2   Data Type Table

tbldevice_types			
Column	**Type**	**Characteristics**	
dev_type_id	smallint(5) (Auto)	Not Null	Primary key for table
device_id	smallint(5)	Not Null	Foreign key to device table
data_type_id	smallint(5)	Not Null	Foreign key to data type table
data_type_min	decimal(10, 4)		Minimum value for device data
data_type_max	decimal(10, 4)		Maximum value for device data
data_type_var	decimal(10, 4)		Variation for device data
ts	datetime		timestamp

Table A.3   Device Data Type Table

tbldata			
Column	**Type**	**Characteristics**	
data_id	int(10) (Auto)	Not Null	Primary key for table
batch_id	int(10)	Not Null	Unique id for batches
value	decimal(10, 4)	Not Null	Numeric data value
ts	datetime		timestamp

Table A.4   Data Table

52

tbltransactions

Column	Type	Characteristics	
transaction_id	int(20) (Auto)	Not Null	Primary key for table
parent	varchar(50)	Not Null	Parent table
parent_id	int(11)	Not Null	Id of primary key in parent table
sql_text	varchar(5000)	Not Null	Sql string executed
outcome	smallint(5)	Not Null	Records effected by transaction
user_id	char(36)	Not Null	Foreign key to user table
ts	datetime		timestamp

Table A.5   Transaction Table

tblexperiments

Column	Type	Characteristics	
experiment_id	smallint(5) (Auto)	Not Null	Primary key for table
experiment_title	varchar(50)	Not Null	Title of experiment
experiment_desc	varchar(500)	Not Null	Description of experiment
user_id	char(36)	Not Null	Foreign key to user table
ts	datetime		timestamp

Table A.6   Experiment Table

tblbatches

Column	Type	Characteristics	
batch_id	int(10) (Auto)	Not Null	Primary key for table
experiment_id	smallint(5)	Not Null	Foreign key to experiment table
device_id	smallint(5)	Not Null	Foreign key to device table
data_type_id	smallint(5)	Not Null	Foreign key to data type table
source_type	tinyint(1)	Not Null	Origin of batch
datetime_start	datetime	Not Null	Start date & time of batch
datetime_end	datetime		End date & time of batch
ts	datetime		timestamp

Table A.7   Batch Table

tbldata_notes

Column	Type	Characteristics	
note_id	int(10) (Auto)	Not Null	Primary key for table
data_id	int(10)	Not Null	Foreign key to data table
note	int(11)	Not Null	Id of primary key in parent table
ts	datetime		timestamp

Table A.8   Data Note Table

tblnote_types			
Column	**Type**	**Characteristics**	
note_type_id	smallint(5) (Auto)	Not Null	Primary key for table
note_type_name	varchar(50)	Not Null	Name of note type
ts	datetime		timestamp

Table A.9   Note Type Table

tblexperiment_notes			
Column	**Type**	**Characteristics**	
note_id	int(10) (Auto)	Not Null	Primary key for table
experiment_id	smallint(5)	Not Null	Foreign key to experiment table
note_type_id	smallint(5)	Not Null	Foreign key to note type table
note	varchar(8000)		Experiment table
ts	datetime		timestamp

Table A.10   Experiment Note Table

tblchemicals			
Column	**Type**	**Characteristics**	
chemical_id	int(11)	Not Null	Primary key for table
common_name	varchar(100)		Common name of chemical
chemical_name	varchar(200)		Chemical name of chemical
formula	varchar(100)		Chemical formula of chemical
cas_number	varchar(11)		CAS registry number
mdl_number	varchar(12)		MDL id number
smiles	varchar(500)		SMILES for the chemical
molfile	varchar(100)		Link to mol file
ts	datetime		timestamp

Table A.11   Chemical Table

tbltraits			
Column	**Type**	**Characteristics**	
trait_id	smallint(6)	Not Null	Primary key for table
trait_type	char(7)		Type of trait: Numeric or String
trait_name	varchar(25)		Name of trait
trait_desc	varchar(50)		Description of trait
ts	datetime		timestamp

Table A.12   Trait Table

54

tblchemical_traits_numeric

Column	Type	Characteristics	
chem._trait_id	int(11)	Not Null	Primary key for table
chemical_id	int(11)	Not Null	Foreign key to chemical table
trait_id	smallint(6)	Not Null	Foreign key to trait table
trait_value	double(13,5)		Numeric value of trait
ts	datetime		timestamp

Table A.13   Chemical Numeric Traits Table

tblchemical_traits_string

Column	Type	Characteristics	
trait_id	int(11)	Not Null	Primary key for table
chemical_id	int(11)	Not Null	Foreign key to chemical table
trait_id	smallint(6)	Not Null	Foreign key to trait table
trait_value	Varchar(255)		String value of trait
ts	datetime		timestamp

Table A.14   Chemical String Traits Table

tblbatch_chemicals

Column	Type	Characteristics	
batch_id	int(10)	Not Null	Co-Primary key for table; Foreign key to batch table
chemical_id	int(11)	Not Null	Co-Primary key for table; Foreign key to chemical table
ts	datetime		timestamp

Table A.15   Batch Chemical Table

tblfile_types

Column	Type	Characteristics	
file_type_id	smallint(2)	Not Null	Primary key for table
file_type_name	varchar(50)	Not Null	Name of file type such as Excel or PDF
file_type_app	varchar(200)		Application associated to file type
file_type_extensions	varchar(200)		Extensions associated to file type
file_type_image	blob		Icon associated to file type
ts	datetime		timestamp

Table A.16   File Type Table

55

tblfiles			
Column	**Type**	**Characteristics**	
file_id	int(11)	Not Null	Primary key for table
experiment_id	smallint(5)	Not Null	Foreign key to experiment table
file_name	varchar(50)	Not Null	Name of file
file_type_id	smallint(2)		Foreign key to file type table
file_size	int(11)	Not Null	Size of file
file_hash	char(32)	Not Null	Unique hash code for file
file_notes	text		Notes related to the file
file_data	mediumblob		Binary data of file up to 16MB
ts	datetime		timestamp

Table A.17   File Table

tblaccess_levels			
Column	**Type**	**Characteristics**	
level_id	tinyint(4)	Not Null	Primary key for table
level_title	varchar(50)	Not Null	Title of level
level_desc	varchar(500)	Not Null	Description of level
level_options	varchar(25)	Not Null	Options for level
ts	datetime		timestamp

Table A.18   Access Level Table

tblusers			
Column	**Type**	**Characteristics**	
user_id	char(36)	Not Null	Primary key for table
user_fname	varchar(25)		First name of user
user_lname	varchar(25)		Last name of user
user_name	varchar(50)		Full name of user
user_pwd	varchar(25)		User password
level_id	tinyint(4)		Foreign key to access level table
ts	datetime		timestamp

Table A.19   User Table

tblexperiment_sharing			
Column	**Type**	**Characteristics**	
experiment_id	smallint(5)	Not Null	Co-Primary key for table; Foreign key to experiment table
shared_user_id	char(36)	Not Null	Co-Primary key for table; Foreign key to user table
share_type_id	tinyint(4)	Not Null	Type of sharing granted to the user by the experiment's owner
ts	datetime		timestamp

Table A.20   Experiment Sharing Table

56

REFERENCES

J. Prilusky, E. Oueillet, N. Ulryck, A. Pajon, J. Bernauer, I. Krimm, S. Quevillon-
Cheruel, N. Leulliot, M. Graille, D. Liger, L. Trésaugues, J. L. Sussman, J. Janin,
H. van Tilbeurgh and A. Poupon (2005). HalX: an open-source LIMS (Laboratory
Information Management System) for small- to large-scale laboratories
Acta Crystallographica Section D, Biological Crystallography
Volume 61, Part 6 (June 2005)
http://scripts.iucr.org/cgi-bin/paper?S0907444905001290

Chris Morris, Peter Wood1, Susanne L. Griffiths, Keith S. Wilson, Alun W. Ashton.
MOLE: A data management application based on a protein production data model
www.cse.clrc.ac.uk/Publications/1431/final_paper_PSFB.doc

dnaTools... Bioinformatic Tools for DNA Research
http://www.dnatools.com/

Perry, D. (2002). Email LIMS in the academic world: This valuable research tool
flourishes in the commercial sector, yet in the academic sector it sits untapped. In
A*bstract of the January 2002 Today's Chemist* (pp. 15-16, 19): ACS Publications.

Spencer Lerch

sller@juno.com
 (317) 502-0321
5442 N College Ave.
Indianaplis, IN, 46220 USA

Education

Master of Science in Chemical Informatics, Expected May 2007
School of Informatics, Indiana University Purdue University at Indianapolis (IUPUI)
Thesis: An Academic oriented LIMS
Advisor: Mahesh Merchand
- A Proof of Concept Project to create an open-source, freeware LIMS designed for academic labs and classrooms.

Bachelor of Arts in ACS Chemistry, May 1998
Connecticut College, New London, CT

Professional Experience

Data Analyst and Database Administrator
Noble of Indiana; January 2003 - Present
- Generate and maintain organizational reports using Crystal Reports 9 and 11.
- Develop software programs using VB.NET and XML.
  - o Includes several in-house software packages including the Report Viewer and Report Manager to view and manage the 400+ reports utilized by the staff.
- Act as primary developer, tester, and support of the software.
- Analyze, troubleshoot and fix issues arising with the databases, including problems with data integrity and transaction verification, using SQL Query Analyzer and Crystal reporting.

Data Analyst and Project Developer
Diversified Mail Services (DMS); April 2001 – August 2002
- Developed and maintained several databases using Access 2000 and VB 6.0, including a multi-user database to generate and track internal jobs.
- Programmed, tested, and debugged a database application used to generate and store employee evaluations, absences, and violations.
- Automated the processing of a bi-weekly group of address database file.
- Performed data analysis and quality control on pre-existing, replicated databases.
- Assisted the Data Manager in setting up and processing client databases used to produce the addresses for mailings done by DMS, and process incoming mail as necessary for a separate division of the company.