BTech 450
End of Semester Report

Zhigang Yin

Abstract

BTech 450 DT project is a one-year project. Given some fairly specific problems, we will design and implement a solution. The problem should be challenging enough to be interesting and motivating.

I think the project I've chosen is fairly challenging and interesting. It evolves a lot of research and self-learning processes. There is a general goal for the project but no specific instruction telling how to get the problem solved, so there's a lot of space left for me to explore and find all sorts of alternatives to the final solution. By comparing each alternatives, I will then discover not necessarily the best but at least a comparatively good solution.

This end of year report illustrates and explains the work I have done in year 2003 (March ~ November) for the Project in Information Technology at Auckland University.

Table of Contents

1. Introduction

1.1 Company Introduction

***Information Tools Ltd* (ITL)**

Established in 1989, Information Tools' investigative software and services are now in use in over 1000 sites spanning over 100 countries worldwide.

The company's headquarters are located close to the beach in Milford, on the North Shore of Auckland, New Zealand.

ITL specialise in the development of software tools and database services for marketing. The software can be used with most forms of marketing data and includes market research, sales data, media research including GRPs, advertising expenditure, retail audit, Customer databases, in fact virtually any data needed for marketing.

Client organisations comprise major multinational companies, strategic planning groups, Universities, Financial Institutions, Government organisations, the Media, business consultants and Market Research Agencies.

ITL produces various types of analysing software to provide database services to their clients according to their needs.
For example,

- **"ESPRI"** service involves designing and converting data into a form that makes it simple for organisations to investigate that data using the ESPRI program.

- "**HARMONI**" is for organisations that want to integrate data from different sources, both internal and external.

To learn more about their services and products, please visit their website at http://www.infotools.com/

1.2 Project Introduction

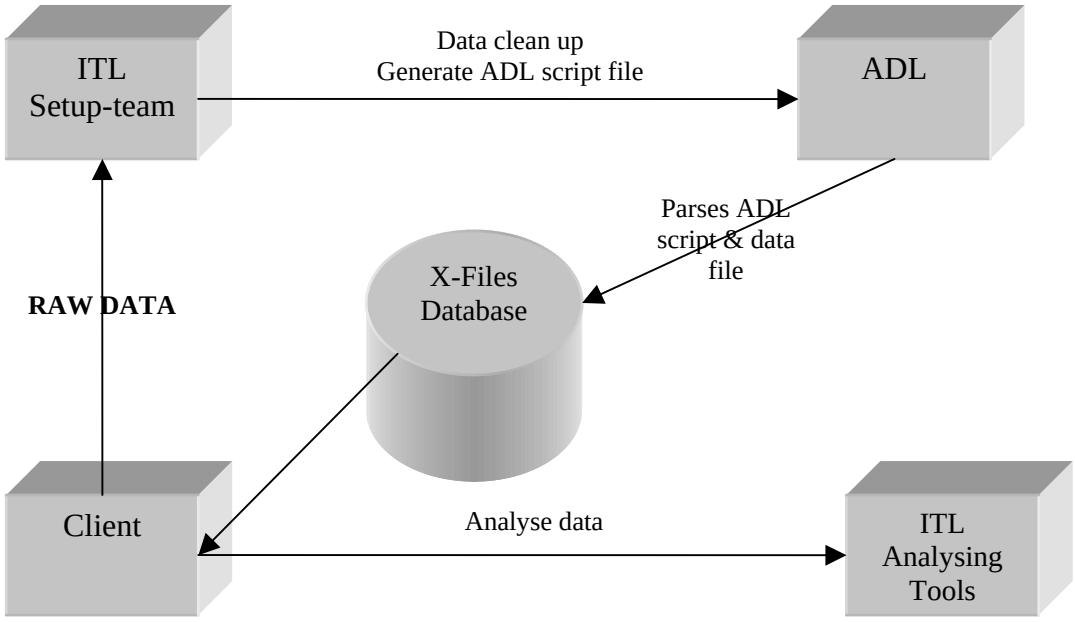
Project title: *Improvements to tools and processes in the creation of Information Tools Ltd (ITL) databases*

Currently around 50% of the company's Auckland office staff (25 or more) is employed to work in a task described as "data setup".

Generally speaking, the setup-team task is that they accept raw data (primarily survey data) in a range of formats and then use *ADL* and supporting programs from the '*iTools*' suite to produce proprietary formatted ITL databases.

The primary tool of the data set-up team is considered to be the ADL program, as at present that is the only program that can create the ITL inverse formatted databases for end-users. Other companies also use the ADL program worldwide, to produce databases for client analysis.

Following diagram shows the work flow of the setup-team task and the role of ITL's main products, (ADL, X-Files Database, and Analysing Tools) in the workflow.



Since the ADL program is a fundamental part of the Information Tools business model, ITL seeks to develop better, more industry standard ways in which the setup task could be done. Through this ITL hopes to improve not only the ADL program itself, but also the production task in which they process data and build databases for customer analysis.

See more detailed description on ADL in Appendix A: What is ADL?

2. Project Description

2.1 Limitations of Existing ADL

There are several issues with the existing ADL language and program:

- ADL scripts are difficult to automate. As ADL scripts are very flexible and the language not well defined, human generated scripts can be very difficult to machine parse. ADL scripts are designed to be read and written by setup *people* and not automated tools. This makes it difficult to automate production processes.
- The ADL language is unusual – the closest equivalent language maybe the AWK scripting language (which is also relatively unclear), but it is sufficiently different from common programming/scripting languages that finding new employees with ADL or directly related skills is extremely difficult. This means that all new setup staff require extensive training in order to be productive.
- The language (syntax and semantics) is not consistent, thus ADL has a steep learning curve. For instance, there are multiple ways of creating an Axis.

The ADL language and program was largely designed for performing ad-hoc jobs, allowing great flexibility in turning textual data into ITL databases. It was not designed for dealing with (the increasingly common) 'trackers' - large regular data streams that remain largely unchanged in format between waves. The setup team currently works around many of the limitations of ADL by utilizing a range of ad-hoc pre-processing tools such as Grid, MSM2ADL or 'CokeV', which manipulate data into a form that ADL can handle. Post-processing tools such as Shrink/Merge/AddAxes also often modify the databases output by ADL, in order to build complete databases for clients.

2.2 Project Requirements

The requirements are from two main sources: the programmers and the Users.

The programmers are standing on the conceptual level, aiming to improve ADL based on their programming experience. Each of the programmers is experienced and can program in more than one programming language, so they can clearly see the advantages of ADL in processing raw data files and in generating databases. At the mean time, by comparing the algorithm and features with other languages, they can see exactly the weakness of current version of ADL.

--Conceptual requirements:

1) **Training**
2) **Speed**
3) **Correctness**
4) **Easy to maintain and extend to meet new challenges.**
   - Flexibility
   - Integration
5) **Retain backwards and forwards compatibility**.
6) **Extra features:**

On the other hand, current ADL users are not necessarily experienced in programming. They might be hired and trained to use only ADL language, but have no idea of what programming languages are like, for example, JAVA, C++, VB. However, they can still find inconvenient bits while using ADL syntax to generate databases. These are reflected in the feedbacks from users to the programmers.

--practical requirements from ADL users:

- **Error handling**
- **Calculation**
- **Security**
- **Variable types support**
- **Other features**

In another document "*ENHANCED ADL DESIGN DOCUMEN*", all the requirements, both conceptual and practical requirements, are listed and discussed in detail. (See in MS word file "DesignDoc.doc")

2.3 Project Objectives

There are several objectives with the project, these objectives are general instructions which will help me, guide me through the stages and finally achieve the goal of this project.

	Objective	**Tasks**
1	Training	ADL and database setup training
2	ADL language documentation	Understanding the current structure of language, and the typical use of the language. Code walk-through. Producing a comprehensive document on ADL.

3	Designing an enhanced ADL language	Analysing the requirements for the design, and exploring alternatives. Documenting the impact of the possible designs on other existing tools that the designs need to co-operate with. Producing a final design document and a language user-guide.
4	Implementation plan for the enhanced language	Formulating the implications of implementing the design choice(s). Producing a document describing the implementation proposal.
5	Program development	Choosing the appropriate programming environment. Program development. Testing and debugging. Code documentation.
6	Final report and presentation	Prepare presentation and report

2.4 Status of completion

Along with each objectives of the project, there is certain form of measurement of completion. These measurements/ milestones are listed in the following table. Some of them are finished in the first semester, and the rest are to be done during the semester break and the second semester of this year.

	Objective	**Milestone**	**Period**	**Status**
1	Training	Ability to read and write ADL scripts	March	Completed
2	ADL language documentation	A comprehensive document describing ADL	April	Completed
3	Designing an enhanced ADL language	A design document and a user-guide for the enhanced language	May	Completed
4	Implementation plan for the enhanced language	An implementation specification	June	To be completed in semester break
5	Program development	Application program and code documentation	July ~ September	To be completed in 2nd semester
6	Final report and presentation	Report and presentation	October	To be completed in 2nd semester

3. Research & Learning

3.1 ADL & Analysis Applications

3.1.1 ADL

Before start working on the project, it is necessary to familiarize myself with the ITL working environment, including the workflow, the analysing software, the ITL Database format and most importantly, ADL.

I was given 40 hours or more on training of how to write ADL script files and "compiling" the script file with ADL programme, also on how to read and extract useful information by using the analysing tools.

I have been provided a CD containing all the information I possibly need for the project: Analysis Application, latest version of ADL programme, ADL manual (softcopy and hardcopy), sample data files, and the source code of the ADL program.

Even though the training session was finished, I must still spend at least 2 hours a week on writing ADL script files and using analysis application, as I've noticed the importance of continuous practice for programming language. There are two components required for running ADL: data file and the ADL script file.

The ADL script files have ".ADL" extension, and must have the same file name as the data file. For example, if the above matrix is stored in a file called "test.dat", my ADL script must be named "test.adl" in order to read in data from the data file. Theoretically, the data file can have any extension, as long as its content is in form of data matrix.

Data files are basically test data made up by myself. They usually contain digit matrix look like:

1, 1, 2, 3, 4
2, 0, 1, 5, 3
2, 1, 5, 1, 6
0, 0, 0, 1, 4

For the above DAT file my ADL script might be like:

```
Filetype delimited ","
["Gender" where = 2
"Female"     =1
"Male"       =0
]
```

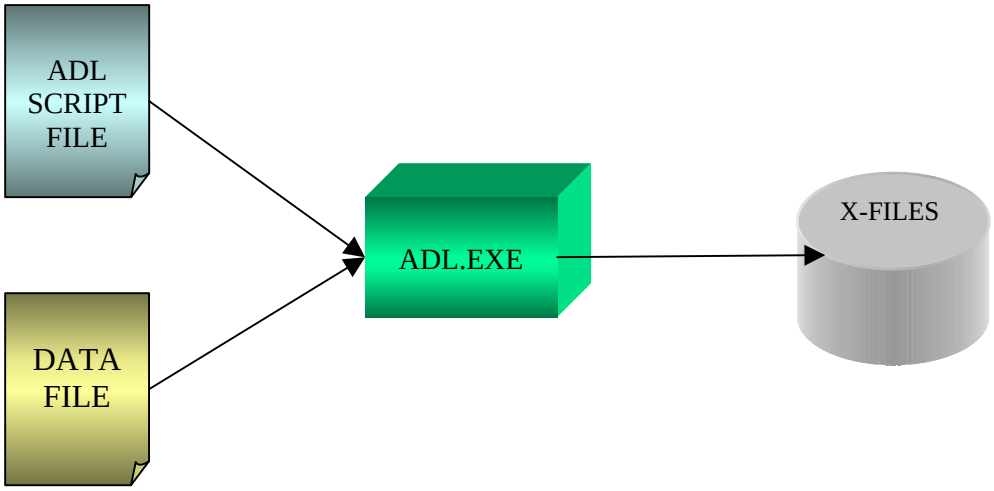
This script file will then looks at the data matrix, and detect that this file is "," delimited. Then it will generate an axis named "Gender" with 2 ELEMENTS namely "Female" and "Male".
"Where = 2" tells ADL to look at the second column of the matrix and count the number of "1"s and "0"s. So the result will be 2 counts on each sex.

The data matrix can be delimited by anything as long as the delimiter does not exist in the actual data. For instance,

1, 1, *, 3, 4
2, 0, /, 5, 3
2, 1, ,, 1, 6
0, 0, @, 1, 4

The above data matrix has a column (column 3) which contains a list of symbols rather than digits. The data is fine, but as "," exists as a data value, the delimiter of the data file can no longer be comma. It has to be changed to some other symbol, or otherwise completely removed, i.e. NO delimiter for this particular file, ADL will then treat each character as a column.

The result of running ADL against the data file is illustrated below:



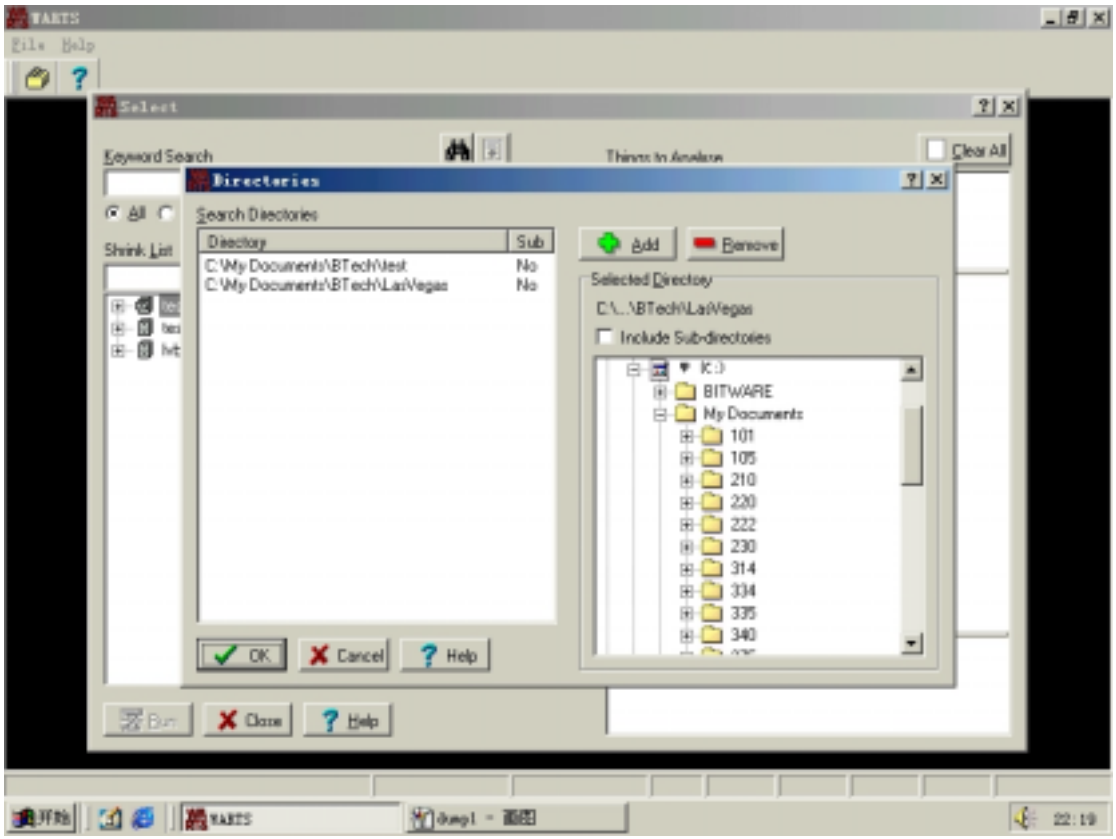
As I am using ADL, I can find features of the language that I didn't notice or even finding problems of the language. The better I understand the language, the better it is for the future work of this project.

A good example is that, while using ADL, I found some features that is supported by ADL but not stated in the ADL manual. This will result in an update of the manual in later version.

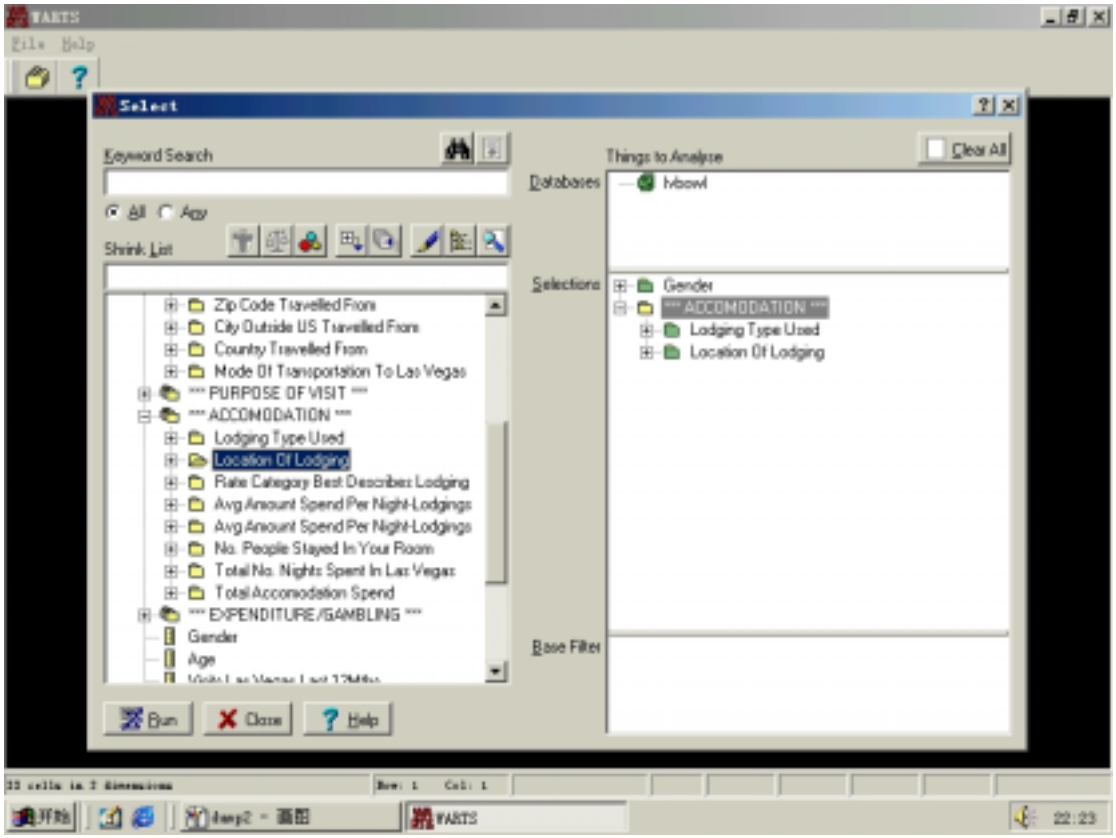
3.1.2 Analysis Application

ITL has produced various brands of Analysis application. Each has slightly different features. For instance, some have more graph types supported; some have slightly different calculation and other supportive functionalities provided. Different brands suits different company's specific requirements. The following screen shots are from one of the analysis applications, called "WARTS".

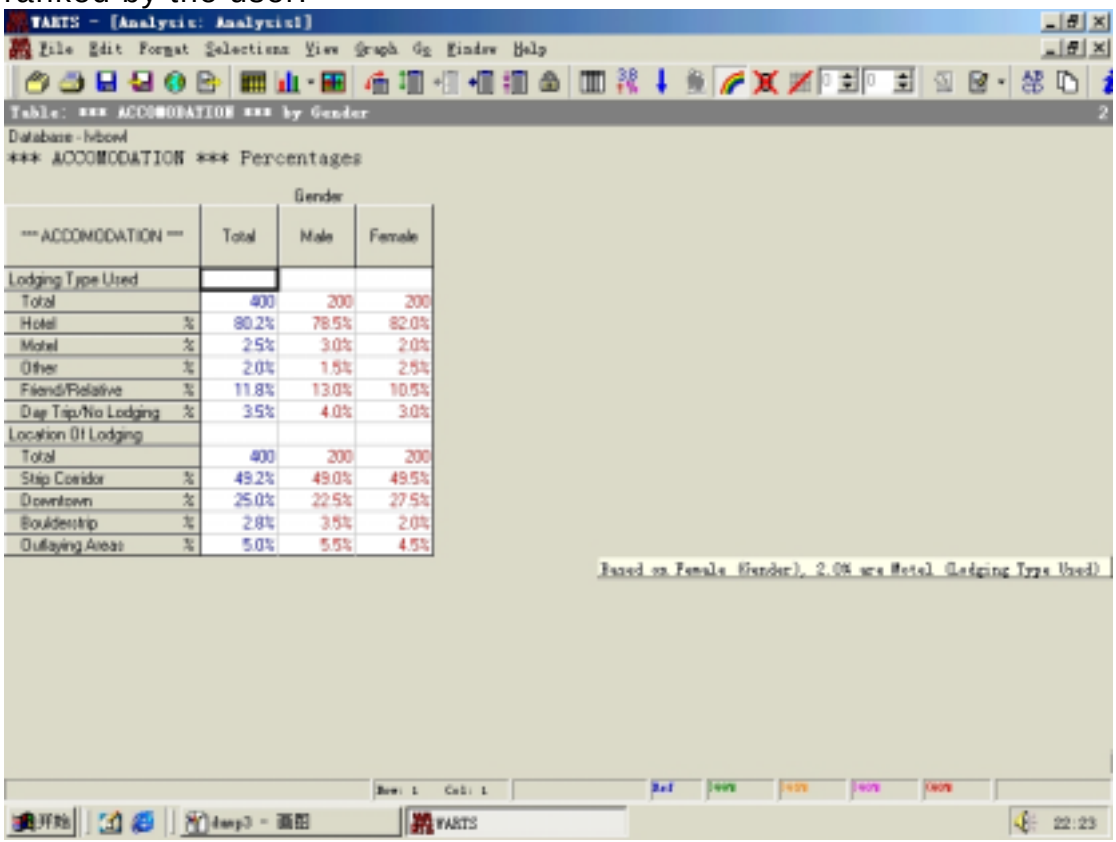
Shot 1: Directory selection dialog box. User selects the directory that contains valid X-Files database, i.e. .xbf, .xef, .xdf files.



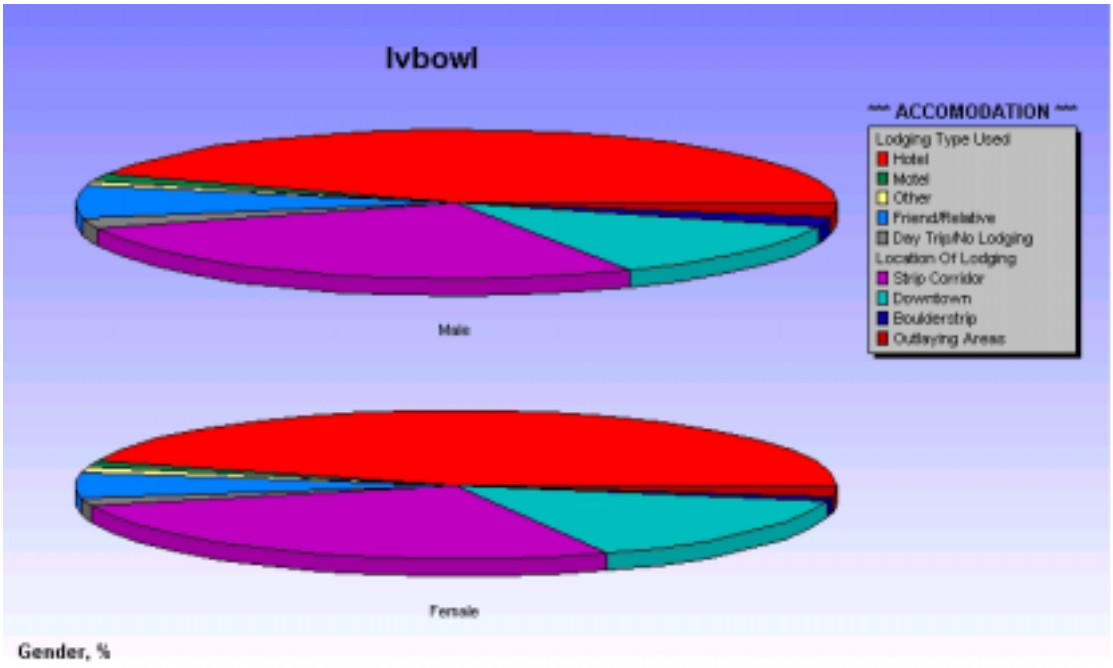
Shot 2: Axis selection dialog box. A tree style database is displayed in the left text area if selected directory was valid. Double click on desired axis and measures, selected ones will be copied over to the right middle text area. Click on "RUN" button, selected items will be tabulated.

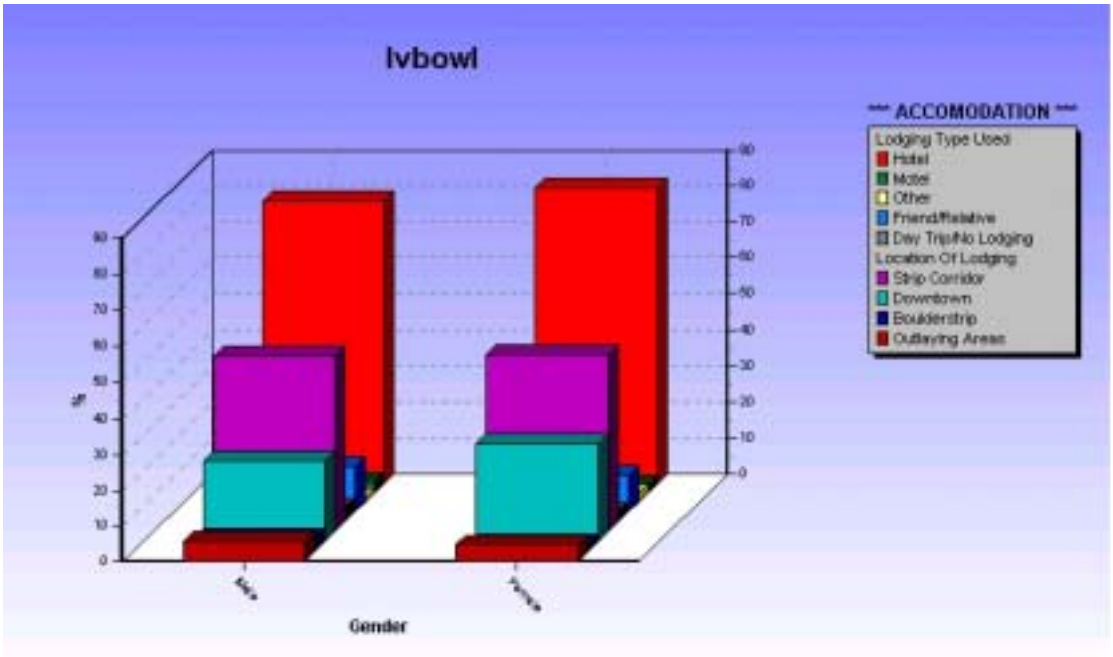


Shot 3: A successfully generated table according to the selected items above. Table's contents can be dynamically selected, filtered, ranked by the user.



Shot 4: 2 different graphs generated automatically on the same table. There are many more types of graphs to choose from.





The practice on the analysis application might seem not too relative to the purpose of project or to ADL. However, this is what the ITL's clients are most interested in, this is the ultimate purpose of the whole setup process and complex ADL script file generation process.

The analysis tools is useful for understanding some of the real-world databases, and how each component of the database can be combined so that meaningful information can be extracted from the tables and graphs.

3.2 ADL documentation

As described in the project objectives, my second task is to produce documentation for the ADL language.

ADL language is similar to many other programming languages, it has two main parts: semantics and the syntax. Since the semantics are clearly defined in the ADL manual, my documentation will focus on the syntax part. After some research and discussion with my supervisor, as well as confirmation of University of Auckland BTech coordinator, Dr. S Manoharan, I've decided to use the ***BNF (Backus-Naur Form notation)*** for the documentation.

There were many resources on the web about BNF documentation, the one I studied on was
"*The BNF Web Club Language SQL, ADA, JAVA, MODULA2, PL/SQL…*"
----http://cui.unige.ch/db-research/Enseignement/analyseinfo/BNFweb.html

There was complete documentation for various programming languages, as well as database languages which I found really helpful for my own documentation.

On top of the traditional BNF, I invented a few "meta-symbols" in order to have clearer description of the ADL language. The list of meta-symbols that were used in the documentation can be found in the "***BNF_ADL documentation.html***" file.

The documentation was written originally in plain text file, after general completion, I modified it to a HTML file and added hyper-links within the document. This gives better navigation help to the user, simplifies the reading process.

However, there are still errors and statements that are ambiguous. ITL intends to keep the documentation for life-time use, so it should and surely will be kept on changing and correcting through the following years. As long as there is modification to the ADL language, there will be change to the documentation. For instance, if the enhanced ADL language was completed and approved after this project, the documentation will need to be updated.

3.3 Language design

After the completion of BNF_ADL documentation, I have gained even better understanding on the current ADL language and observed many of the strengths and weaknesses of the language. Next is to design a model that is enhancing the current language, so that ADL can be more productive.

My design is carried out according to the requirements from both conceptual level and the practical level as listed in section 2.1 and further discussed in the "*Enhanced ADL design document*". (See in MS word file "DesignDoc.doc")

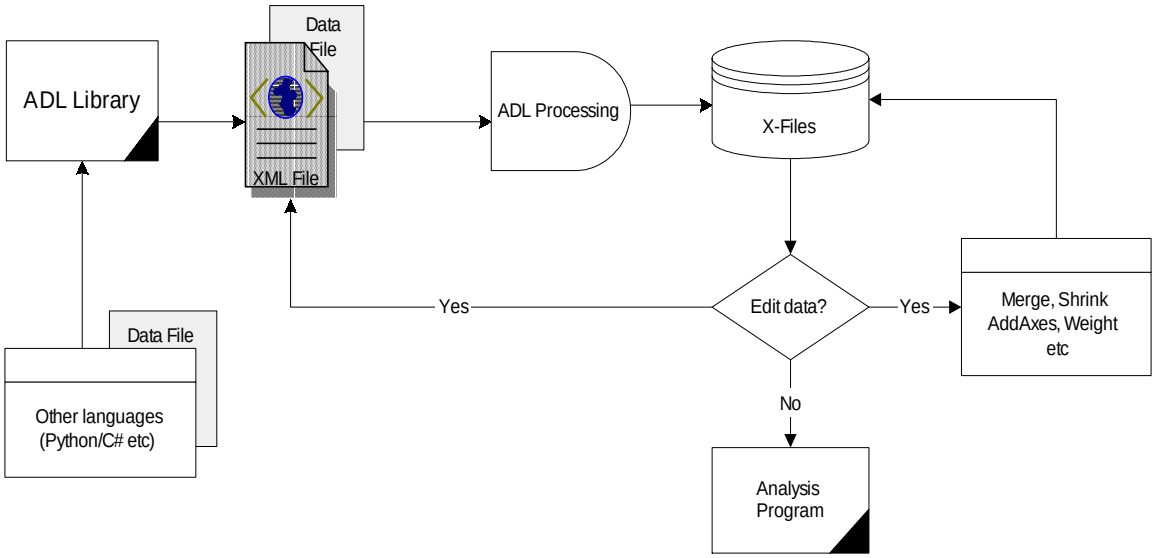
Three initial models were raised, each using a different approach:

- ***Model One***: "Super ADL" --- Original ADL language with extra features
- ***Model Two***: ADL script file generated using another programming language
- ***Model Three***: XML format of X-Files Database & XML to X-Files engine

Pros and **Cons** of each alternative are discussed in the design document. A test table was set up with regards to the requirements; each alternative was rated according to these requirements.

By then, I was clear that each model has its own advantages and disadvantages.
The more advantages my final solution can have the better. However, this is limited by both the time and the complexity.

My decision for now is Model, i.e. to rewrite ADL in another programming language, using API for the programming language (XML handlers, libraries), and output a XML file as an intermediate format. A convert engine will then take this XML file and the original .DAT file as input in order to generate the ITL Database in X-Files format.
This is shown in the following diagram:



3.4 X-Files

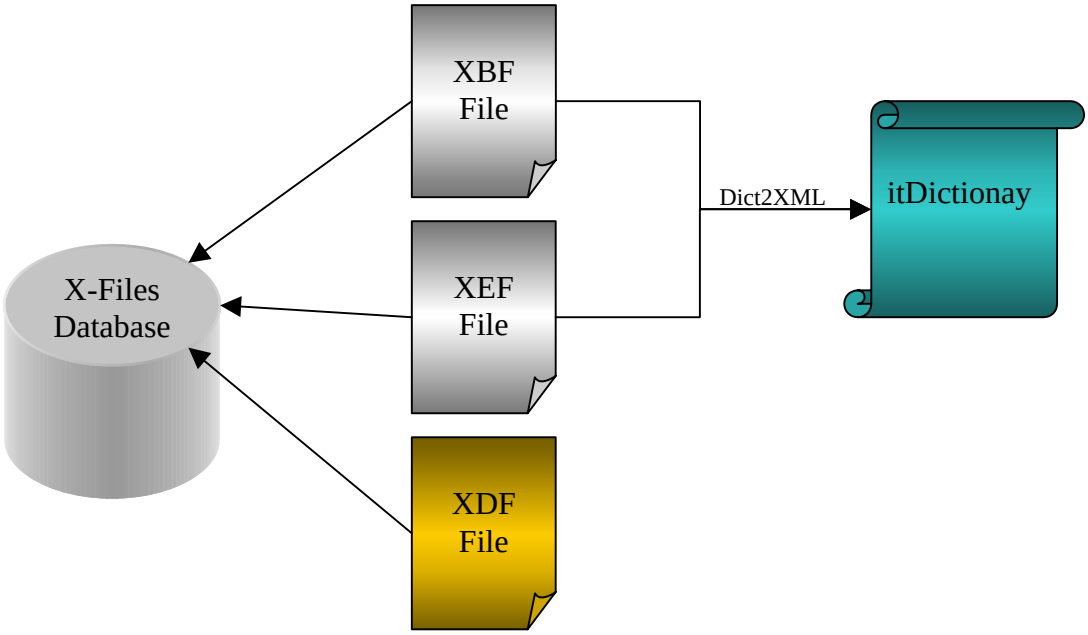
X-Files is the generic name for the ITL database, it consists of three types of files:

- The base file (*.xbf):storing the questions
- The element file (*.xef): storing pre-coded answers.
- The data file(*.xdf): storing the data

The base file and the element file define the DICTIONARY.
Both the base file and the element file consist of two major components:

- File Header
- Item

Following diagram visually describes the structure of the X-Files database and how the XML “ITDictionary” was generated from XBF & XEF files:



The structures of each type of the X-Files are discussed in detail in the presentation document “X Files”, by *Dr. S Manoharan*. (See in XFiles.pdf)

3.5 Research on other project related topics

3.5.1 XML (Extensible Mark-up Language)

There is a tool in the "iTools" suite that is called "Dict2XML" written by *Dr. S Manoharan*. Its function is to generate a XML file from the X-File dictionary, i.e. xef file and xbf file. A copy of the Dict2XML generated XML file is included in this report file group with file name "test.xml"

To refresh the XML knowledge I already have and to gain better understanding of XML concepts, I went through the on-line tutorial materials about XML on *http://www.w3schools.com*. Some on-line forum discussing topics on XML (*http://slashdot.org/comments.pl?sid=57483&cid=0&pid=0&startat=&threshold=1&mode=thread&commentsort=3&op=Change*)

3.5.2 COMPSCI 330 S1 C --- Language Implementation

This stage three computer science course at University of Auckland helps to understand on how computer languages are specified, what they mean, and how they are implemented, is fundamental to computer science.

Unfortunately I did not study the course in advance of starting the project. Lack of knowledge on the aspects this course covers might lead to huge difficulties while proceeding with the project work.

Therefore I downloaded the lecture handouts and some of the reading materials provided on the course web page and studied thoroughly. These materials gave me a slightly different view of programming languages, and I found it helpful at the design stage of the new language.

4. Conclusion

4.1 Conclusion of this semester's work

The first half of the project was done in computer labs of University of Auckland; Information Tools Ltd. Auckland office, Milford; and from home.
My research and learning process on this project was greatly supported by Mr. Grant Black, project supervisor; Dr. S Manoharan, BTech coordinator; All members of the ITL software development team. I do appreciate their help and guidance.

Both technical and practical knowledge were developed during this period. I am sure these are helpful to the rest of my university study, also very important to future work.

- Technical knowledge gained
  - File type conversion
  - XML knowledge
  - Language parsing
  - Documentation skill
  - Language Design concepts

- Practical knowledge gained
  - Research technique
  - Document writing
  - Report writing
  - Presentation skills
  - Team work
  - Communication skills
  - Business Rules (business workflow, terms of confidentiality)

4.2 Plan for next semester

Month	Objective
June	Implementation plan for the enhanced language
July	Program development stage 1: Re-write ADL in other programming language (C# .NET)
August	Program development stage 2: Re-write ADL continues
September	Program development stage 3: XML to X-Files convert engine.
October	Final report and presentation

Note: The arrangement of program development stages might be changed

Reference:

Information Tools Ltd:

ADL V3 Manual	Latest version of ADL language specification
"ADL" Resource CD	18/03/2003
ITL TIF Proposal document	By Grant Black. 05/03/2003
"X-Files" presentation doc	By Dr. S Manoharan 14/02/2003

Internet:

XML on-line tutorial	*http://www.w3schools.com*
"The BNF Web Club Language SQL, ADA, JAVA, MODULA2, PL/SQL…"	*http://cui.unige.ch/db-research/Enseignement/analyseinfo/BNFweb.html*
On-line forum discussing topics on XML	*http://slashdot.org/comments.pl?sid=57483&cid=0&pid=0&startat=&threshold=1&mode=thread&commentsort=3&op=Change*
5 questions about language design	*http://www.paulgraham.com/langdes.html*

University of Auckland:

COMPSCI734 Resource CD set	VS. NET, MSDN
COMPSCI330 lecture notes	Semester one, 2003

Appendix A: What is ADL?

What does ADL stand for? Nothing, it is just the tool that is used by ITL.
If really keen on how ADL was named, well, it could be anything, **A**nalytical **D**ata **L**anguage, **A**dvanced **D**ata **L**anguage …you name it!

ADL consists of two parts:

- ADL Program: a single Windows 32 command line driven executable
- ADL language: a data description language

ADL program: Internally, it consists of language syntax parser 'front end' and backend that writes ITL databases using a “blob manager”. The ADL program is sometimes described as a “compiler”, which technically it is not, as it does not produce executable code. It does however share some features of a language compiler such as a C++ or the TAWK compiler in that it parses a language and generates syntax errors/warnings at both parse ‘compile’ time and when building ITL databases (run time).

ADL language: The ADL language is much more difficult to describe. It is not a full programming language, but is more like XML in that it essentially describes or defines data. Unlike XML, an ADL script can describe not only the location and size of data located in a separate data file, but also manipulate or construct data.

For instance, in the ADL language, the line:

```
; q22
        ["Gender"     where=195_2
        "Male"              =1
        "Female"            =2
        ]
```

Specifies the data that indicates the respondent’s gender, is stored in columns 195 and 196