# Introduction

Data warehousing involves gathering data from various sources, conforming it and storing it as multidimensional cubes to allow OLAP (Online Analytical Processing) or data mining. A data warehouse is a database specifically used for reporting; thus it is usually optimized for answering queries on existing records and not for the insertion or updating of records. They tend to be much larger than operational databases, often hundreds of gigabytes to terabytes in size. A dimensional model is often used to provide the logical design used in a data warehouse. The data warehouse generally consists of an ETL tool, a database, a reporting tool and other facilitating tools, such as a Data Modeling tool.

Multidimensional cubes are often considered more appropriate for OLAP applications than schemas normalized to the third normal form (3NF). Multidimensional cubes have important benefits for business intelligence which include understandability and query performance. In query performance, the number of join operations is greatly reduced when using a multidimensional cube as opposed to a relational schema. Furthermore, the query plan can be improved through “star joins” which may be performed faster through indexing or result set size prediction. Finally, dimensional models are generally easier to understand (cite Kimball) OLAP is meant for domain specialists and not IT professionals (cite litOLAP).

In this project, we build a small data warehouse over texts and build an OLAP cube of the frequencies of locations mentioned. Similar work has been done on this area such as the LitOLAP project (cite here). LitOLAP involves applying Business Intelligence techniques of Data warehousing and OLAP to the area of text processing. In their data warehouse only the information required to build their OLAP cubes is obtained from the data sources. One of their dimensions is the word dimension. This dimension allows roll-up by stem suffix, and (several layers of) WordNet (reference to WordNet) hypernyms.

Our project consists in downloading books from the Project Gutenberg Canada, a website in the Canadian public domain that offers ebooks at no charge. (To implement this project we used ROLAP since we are storing the OLAP cube in a relational database, I think). For the ETL process, we used GATE to perform Natural Language Processing (NLP), Nokogiri an XML parser for ruby, and Pentaho’s ETL tool, Kettle. As our RDBMS, we used MySQL. Finally for our reporting tool and the construction of OLAP cubes we used JasperServer 4.0 which provides a Mondrian and JPivot installation.

## The Design of the Star Schema

To build multidimensional cubes we use dimensional modeling which is a technique of logical design for structuring data so that it is intuitive to business users and delivers fast query performance. Multidimensional cubes are often considered more appropriate models for OLAP applications as opposed to normalized. The normalized models we talk about go up to the third normal form (3NF). Industry also refers to them as 3NF models or entity-relationship (ER) models. These models seek to reduce redundancies, and are considered better for transactional processing or OLTP applications. Normalized models and dimensional models contain the same information, but are structured differently. The key difference between them is the degree of normalization. While normalized models are completely normalized to 3NF, dimensional models normalize some tables to 2NF and others to 3NF. Dimensional modeling divides the information into measurements and context. The measurements are captured by the organization’s business processes and are usually numeric; they are called the facts. The context is represented by the dimensions which help answer the questions of who, what, when, where, why and how of a measurement. Dimensional models may be stored as star schemas or cubes. When stored in a relational database platform, they are called star schemas, and when stored in an OLAP structure they are called cubes. Because we store the tables for our OLAP cube in a relational database, we are essentially designing a star schema.

In a dimensional model, fact tables are normalized to 3NF because the related context is moved to dimension tables. In contrast, the dimension tables are kept denormalized as flat dimensional tables. Often, dimension tables resemble 2NF tables with many low cardinality descriptors. Fact tables are comprised of facts which are numeric measurements. In our project we picked as our fact table, a table of sentences and places, where a fact is the occurrence of a given place in a sentence. We measure the frequency at which a place appears in the sentence.

Our dimension tables are a table of places, and a table of sentences. The table of sentences has a surrogate key without semantics that provides better performance for lookup operations on it. The sentence table provides a hierarchy of books. Where sentences can be rolled up to books and books can be rolled up to authors. However, because author and book may have a many-to-many relationship, this part of the hierarchy becomes ragged and difficult to add to the multidimensional model. There are some possible solutions to this problem, which include:

1. Allocating: We give each author a fraction of the frequency of the locations. So, if we have four authors, each fact of the table containing it will have a frequency of ¼. Thus, when we add up the sentences by book, we get a total of one for each sentence and place occurrence instead of four.
2. I also wondered on the possibility of allocating the frequency of each author by ½. Why not instead divide by the number of authors when aggregating by book.
3. Treating all the authors of a given book as a single author.

The table of places has an issue, in which the place that is mentioned may be a continent, a country or a city. If all the places mentioned are cities, then one could have a nice hierarchy, but because in a sentence, one may refer to a specific level of the hierarchy. I am not sure how to best deal with this in our star schema.

Most multidimensional models require mapping each fact to a value at the lowest level in each dimension, but some models relax this mapping requirement (cite Pedersen). This is often called the finest grain of the fact table. This was one of the things we had thought about. We explored two possibilities for it:

* Having the finest grain be the frequency of a place in a given sentence by a given author: AuthorID x SentenceID xPlaceID 🡪 Frequency
* Having the finest grain be the frequency of a place in a given sentence: SentenceID xPlaceID 🡪 Frequency

I have created a star schema that builds a fact table of the presence of a place of a given sentence. The fact table is associated to a sentence and place dimension.

## The ETL Process

Populating the data warehouse from data sources involves a process of three main phases: Extracting the data from each source, transforming it to conform to the warehouse schema and cleaning it, and loading it into the warehouse. This process is known as ETL (Extracting, Transforming and Loading).

The data extraction step consists in bringing data from different sources into a database where they can be modified and incorporated into the warehouse. In our project, we extracted the information from a single data source, the Gutenberg.ca website. Before we extracted books from this website, we first downloaded the HTML file onto local disk and cleaned it of unnecessary text and tags that could confuse our pattern matching regular expressions that would eventually extract information from it. To extract the books from the website we used an HTML parser that would identify text files links in the website and download them into a location on local disk.

The transformation process uses a set of rules and scripts to transform the data from an input schema to a destination schema representation. Most of the work in our ETL process was done on this stage. Data cleaning is also an important part of the transformation stage and it consists in fixing errors and differences in schema conventions. These differences may result in inaccurate query responses and consequently inaccurate mining models. In our project, we had to remove some irrelevant content from the books. We did most of this part with ruby scripts that used pattern matching rules to gather only the necessary information. Some preliminary common information found in the text files was also eliminated as it wasn’t particularly relevant to the book.

## Natural Language Processing

One of the most critical steps in our Transformation process was that of doing Natural Language Processing (NLP) for the texts we had downloaded. We did part of this transformation process with GATE (cite) , and another part of it with the WhatLanguage ruby library (cite). Both are open source free software projects. Initially, we use WhatLanguage to determine the language a text is written in. If the text is written in English we keep it, and if it is not we discard it. Once we have all texts that are written in English, we use GATE to annotate the texts. GATE builds an XML file of the english texts where sentences and places are tagged. Sentences are tagged using GATE’s default sentence splitter, and the places are annotated using a Gazetteer. We illustrate the transformation of a text to xml annoted text by GATE in the following listing:

Hello this is a test file for ANNIE. I went to Halifax last week. I live

in St John, and I am studying at the University of New Brunswick. I also

have a supervisor whose name is Daniel in Montreal. My supervisor's name

in St John is Owen.

<Root>

<Sentence>Hello this is a test file for ANNIE.</Sentence>

<Sentence>I went to <Location>Halifax</Location> last week.</Sentence>

<Sentence>I live in <Location>St John</Location>, and I am studying at the

University of New Brunswick.</Sentence> <Sentence>I also have a supervisor

whose name is Daniel in <Location>Montreal</Location>.</Sentence>

<Sentence>My supervisor's name in <Location>St John</Location> is

Owen.</Sentence>

</Root>

## Incremental Updates to the Data Warehouse

Though we have not planned implement this part of the ETL process, we describe it in this section. In order to perform ETL of the database I would first take the latest html file and do a diff with the previous html file that was added to the database. Using diff I would determine what contents are new in our file, and generate a new file with only the new content. After that, I would attempt to run the fixFile.rb script to fix any manual inconsistencies that could hurt the ETL process. Then, getSources can obtain the relevant information on the authors.