|  |
| --- |
| **FUNCTIONAL SPECIFICATION** |
| DATA GENERATOR v1.0 |
|  |
| *It contains functional details of project “Data Generator” done by project group SKNCOE-2.* |
|  |
| ***SKNCOE-2 Group*** |
| **11/10/2011** |
|  |

DATA GENEATOR

1.0

REVISION HISTORY

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Date | Version | Remarks |
| 1 | SKNCOE2 | 10-Nov-2011 | 1.0 | Draft |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Table of Contents

[Problem Statement 5](#_Toc308790800)

[Functional Details 6](#_Toc308790801)

[Enhancements 9](#_Toc308790802)

[Limitations 10](#_Toc308790803)

[Appendix-A 11](#_Toc308790804)

# Problem Statement

**Design and implement a system to generate large amounts of artificial data for use with machine learning techniques.**

By designing and implementing a system which can produce large quantities of artificial data where the distributions of attributes are known and the complexities of the relationships between attributes are understood.

It is a general tool to generate records for any data types in a given format. In this XML user will pass the list of tables for which he wants to generate the records. For each table there will be another metadata xml file which has details of data type and format for each columns. The tool will parse the xml and start to generate the records.

Key points:

1. It will generate the records for different tables in parallel.

2. It will manage the foreign key relationship between two tables.

3. Give option to upload the generated records into a data base using bulk load utility.

4. Distributed data generator: Use nodes in a cluster to generate the records.

5. Supporting all possible formats for a column is the most challenging part of the tool.

# 

# Functional Details

**1. Preprocessing details:**

We will be retrieving the details of the files containing the database details such as table name, column names.

There should be a working database and we will need an [ODBC](http://www.easysoft.com/developer/interfaces/odbc/index.html) driver to connect to a database. ODBC drivers should work equally well on MS Windows and other operating systems with some minor alterations.

**2. Input details:**

Xml File: Table name and Column names.

Data types available: names, phone numbers, email addresses, cities, states, provinces, counties, dates, street addresses, number ranges, alphanumeric strings.

Datatypes in Oracle and their Range.

| **Datatype** | **Description** |
| --- | --- |
| VARCHAR2(*size* [BYTE |CHAR]) | Variable-length character string having maximum length *size* bytes or characters. Maximum *size* is 4000 bytes or characters, and minimum is 1 byte or 1 character. You must specify *size* for VARCHAR2.  BYTE indicates that the column will have byte length semantics; CHAR indicates that the column will have character semantics. |
| NVARCHAR2(*size*) | Variable-length Unicode character string having maximum length *size* characters. Maximum *size* is determined by the national character set definition, with an upper limit of 4000 bytes. You must specify *size* for NVARCHAR2. |
| NUMBER[(*precision* [,*scale*]]) | Number having precision *p* and scale *s*. The precision *p* can range from 1 to 38. The scale *s* can range from -84 to 127. |
| LONG | Character data of variable length up to 2 gigabytes, or 231 -1 bytes. Provided for backward compatibility. |
| DATE | Valid date range from January 1, 4712 BC to December 31, 9999 AD. The default format is determined explicitly by the NLS\_DATE\_FORMAT parameter or implicitly by theNLS\_TERRITORY parameter. The size is fixed at 7 bytes. This datatype contains the datetime fields YEAR, MONTH, DAY, HOUR, MINUTE, and SECOND. It does not have fractional seconds or a time zone. |
| BINARY\_FLOAT | 32-bit floating point number. This datatype requires 5 bytes, including the length byte. |
| BINARY\_DOUBLE | 64-bit floating point number. This datatype requires 9 bytes, including the length byte. |
| CHAR [(*size* [BYTE |CHAR])] | Fixed-length character data of length *size* bytes. Maximum *size* is 2000 bytes or characters. Default and minimum *size* is 1 byte.  BYTE and CHAR have the same semantics as for VARCHAR2. |
| NCHAR[(*size*)] | Fixed-length character data of length *size* characters. The number of bytes can be up to two times *size* for AL16UTF16 encoding and three times *size* for UTF8 encoding. Maximum *size* is determined by the national character set definition, with an upper limit of 2000 bytes. Default and minimum *size* is 1 character. |

# Vector: It is a class template in the [C++](http://en.wikipedia.org/wiki/C%2B%2B) [Standard Template Library](http://en.wikipedia.org/wiki/Standard_Template_Library), which functions like a [dynamic array](http://en.wikipedia.org/wiki/Dynamic_array). It is one of several [data structures](http://en.wikipedia.org/wiki/Data_structure) called containers.It is implemented as a class [template](http://en.wikipedia.org/wiki/Template_(programming)), and can thus be used as a generic framework that may be instantiated e. g. as a vector of [integer](http://en.wikipedia.org/wiki/Integer) values, a vector of [strings](http://en.wikipedia.org/wiki/String_(C%2B%2B)), a vector of instances of a user-defined class.

**Map**: It is a standard C++ structure. It is a sorted [associative array](http://en.wikipedia.org/wiki/Associative_array) of unique keys and associated data. The types of key and data may differ, and the elements of the map are internally sorted from lowest to highest key value.

**3. Output details:**

Data Generation procedure stores data in memory similar to the way structured data is stored in an XML file. We use data tables, data rows and data columns where data table contains a list of data rows and a data row contains a list of data columns. Collections of data (e.g. in a database) can be distributed across multiple physical locations.

Formats supported: Option to generate data given in XML, Excel, HTML, CSV or SQL .

**4. Post processing:**

Bulk Loading:

The Bulk Loader is a facility that allows you to populate database tables from flat files. To use this facility, one requires

(i) flat file(s)

(ii) A control file that tells ORACLE how to "map" fields of the flat file to columns of an RDBMS table.

**5. Unit testing:**

The Data Generation generates a meaningful set of test data in our database to support accurate testing and we can define a unit test plan which produces the same set of test data on each test run. An instance of testing as such **Proof Of Concept** is a realization of certain methods to check the feasibility and unit testing must set up its execution environment before it is run so that it can consistently verify the code it is meant to test. Once the test has been implemented and verified, it is checked into source code and will be used for the rest of the product's lifetime to ensure that the method continues to behave as expected. Once the test has compiled and failed, the developer writes only the code necessary to make the test pass in the next run. With each iteration, all unit tests are run to verify that there are no regression errors. Each unit test is run at least twice for the method being developed and it covers all data types in one platform. This type of unit plan makes the implementation of software with all covered functionality.UTP will have details like data imported into our database through using MYSQL and through various test plans the expected output is achieved. Actual size of data to be processed can be taken into account with the suitable space complexity and timing complexity.

# Enhancements

More distributions would increase the possible datasets and types of relationships a user could generate. Since nowadays a lot of **data** are stored in relational database we propose an algorithm to automatically generate **XML** Schemas. The algorithm may be used to facilitate automatic **generation** of **XML** documents containing information extracted from relational databases by preserving constraints during the translation process.

We choose to define a mapping into **XML** Schema because the usage of the eXtensible Markup Language (**XML**) has spread to many application fields in the recent years and one of the main aim of **XML** is to facilitate the exchange and integration of **data** between different applications. Moreover, **XML** Schemas have more expressive power w.r.t. DTD’s.

* We can extend the system we would introduce a **multiprocessing version**. This would enable the process of classification to be completed almost n times faster, where n is the number of other processors. The standard system would be run on one machine and by receiving pings would have a list of all other programs that were running the software. The created list of **queries** for classification would be split into n subsets. These would be sent to each other machine with the rule file. Each slave system would then calculate the portion of queries they received and send back the list of solutions. The main system would integrate these into the Dataset.
* An future area of research and development for the DGT could be to add support for a multi-relational version. This would allow relationships between different Signatures to be defined, further enhancing the research possibilities.

# 

# Limitations

In order for the system to be used successfully by the Machine Learning Group there are several key aims for the project. Although these aims are typical of many software projects their particular relation and importance to this project are stated below.

a. FLEXIBILITY:

With regard to the Data Generation Tool (DGT) various machine learning file formats are not supported this leads to ponderous situation. The software should also be able to produce data for use within different domains of machine learning.

b. EFFICIENCY

Many a times Generation time is not of great importance as operations can be run overnight

but the system should aim to be as fast as possible without reducing the scope of options available.

c. EXTENSIBILITY

The part of the system are inter dependent so alteration to any single aspect of the system may cause significant changes to other parts. In particular accommodation for future data formats and new generation engines should be provided.

# 

# Appendix-A

Source and binary distributions installation instructions are available for the following platforms:

* [Windows](http://xerces.apache.org/xerces-c/install-3.html" \l "Windows)
* [UNIX/Linux/Mac OS X](http://xerces.apache.org/xerces-c/install-3.html#Unix)
* [Cygwin](http://xerces.apache.org/xerces-c/install-3.html#Cygwin)

WINDOWS:

The Xerces-C++ source is available in the source distribution: xerces-c-3.1.1.zip.

Install the Xerces-C++ source distribution by using unzip on the xerces-c-3.1.1.zip archive in the Windows environment. You can use WinZip, or any other UnZip utility.

This creates the 'xerces-c-3.1.1' sub-directory containing the Xerces-C++ source distribution.

If we need to build the Xerces-C++ source after installation we follow the [Build Instructions](http://xerces.apache.org/xerces-c/build-3.html).

Xerces-C++ source distribution comes with Microsoft Visual C++ projects and solutions. The following describes the steps you need to build with this compiler.

To build Xerces-C++ from the source distribution you will need to open the solution containing the project. The solutions containing the Xerces-C++ project files are in the following sub-directories in the xerces-c-3.1.1 directory:

Once you have the solution open, you need to build the project named XercesLib. You can select Debug/Release, Static/DLL, and, for VC8, VC9, and VC10, 32/64 bit builds using the Configuration Manager dialog. You can also select whether the Xerces-C++ library should use ICU for transcoding.

When building your own applications you need to make sure that you are linking your application with the xerces-c\_3.lib (Release) and/or xerces-c\_3D.lib (Debug) libraries (or the static versions of them) and also that the associated DLLs are somewhere in the executable/DLL search path (PATH).