SGML AND SQL*TEXTRETRIEVAL

A. Sanchez de Miguel

(Presented at the European Oracle User Group Meeting, Madrid, 18-20 April 1990)
SGML AND SQL*TEXTRETRIEVAL

Adolfo Sánchez de Miguel
CN Division / SW Group
CERN, GENEVE

0. ABSTRACT

This paper describes a system based on ORACLE for handling documents marked-up with the ISO standard SGML text formatter.

At CERN an ORACLE application has been developed which retrieves and manipulates documents describing routines of the CERN Program Library.

Documents and their structure are stored in the data base, so that SQL*TEXTRETRIEVAL, PRO*C, and SQL*REPORTWRITER can be used to generate indexes by keywords, to reconstruct marked-up SGML documents, or allow editing of text interactively.

A future development of this system could be the updating of the structure of the documents within the ORACLE data base without going through the SGML interface.

1. INTRODUCTION

The CERN Program Library is a large collection of general-purpose programs maintained and offered on the CERN central computers. The library contains about 2500 subroutines and programs which are grouped together into about 450 program packages. These packages are briefly described in so-called “short write-ups” which allow users to be able to determine their suitability for a particular problem.

All these short write-ups are published every three months in a manual, which was produced using a set of programs and a set of flat files without a standard format. This situation led to several maintenance problems as well as to a low performance when creating the indexes of topics, which had to be done manually.

At the end of last year it was decided to store all the write-ups using the ISO standard SGML which has taken the position of standard text formatter at CERN.

A natural evolution of the Manual was thought to be the creation of a data base containing the relevant information, and every new edition of the manual would just be produced by the generation of a data base report tagged in SGML.

This ORACLE data base and the development of the application that manages the write-ups, are the subject of this paper.
2. SGML

2.1. INTRODUCTION

SGML, Standard Generalized Markup Language, ISO 8879, is a language for representing documents in a descriptive markup form.

The main idea of SGML is that text consists of logical components called elements. The way in which elements of a document are marked in the document is to use so-called “tags”. For example a formula element would begin with the starttag `<DF>`, the text of the formula would follow and the element would conclude with the formula endtag `</DF>`.

In a tag it is possible to change the default values specifying attributes. For example, an align attribute `<DF ALIGN=LEFT>` could be used in order to center the formula on output while the default value for the attribute ALIGN is to align the formula left on output.

2.2. STRUCTURE OF THE DOCUMENT

In SGML the contents and the structure of a document are separated from the way it is processed. The structure of class documents is defined by a DTD, Document Type Definition, which mainly contains information about tag names, tag definitions, and tag frequencies, the order in which tags may appear, the content of the element delimited by a tag down to character level and the tag’s attributes.

These DTDs are compiled by a parser that checks them for consistency and conformance with the standard. Once a document has been marked up conforming to the DTD, the DTD specific parser constructed in the previous step will check that the document conforms to the structure defined by the DTD.

2.3. ARCHITECTURE OF AN SGML APPLICATION

The first step of an SGML application is the design and compilation of the DTDs that describe the structure of the system’s documents. Once the DTDs have been parsed error free the documents have to be tagged according to these DTDs, possibly with a syntax driven editor.

A document that has been successfully parsed can be sent as output to a text composition and presentation system. It can also be stored in a data base in order to make it accessible to data base applications. This storage is done simply by writing a program as part of the SGML parser which is executed during the parsing procedure (many SGML parsers give the possibility to execute operating system commands when a tag is found). When a new version of the document is needed the document is generated as a data base report tagged in SGML as we see in Fig.1.
3. SQL*TEXTRETRIEVAL

3.1. INTRODUCTION

Text retrieval systems provide the ability to store, manage and retrieve textual information. A key requirement of a retrieval system is that the text can be placed in a free unstructured format and that the user is able to retrieve upon this unstructured information and not only upon the structured data (date, author, name, ...).

For efficiency reasons the searching through the unstructured information can’t be done every time that someone needs it. It is necessary to construct text indexes in order to retrieve upon them.

It is also very important to define a retrieval language in order to allow users to formulate their queries in a simple way.

SQL*TEXTRETRIEVAL, the ORACLE approach to a retrieval system, is the extension of the ORACLE RDBMS in order to manage unstructured information without losing all its advantages in dealing with structured information.
3.2. ARCHITECTURE

SQL*TEXTRETRIEVAL consists of several software routines that provide the following:

- text loading and indexing
- extended retrieval language based on SQL and user interface (STRRMU) for using it
- user exits and 'C' function library for developing applications
- maintenance of text data dictionary, and word and location lists for individual text tables

Fig.2. SQL*TEXTRETRIEVAL ARCHITECTURE

All data, including original text, text indexes, and the text data dictionary are stored within the ORACLE data base. The original text can be stored in columns of datatype LONG (1 per table) as well as in columns of datatype CHAR (as many as needed). There is a third type of text column, the KEYWORD column, of datatype char, which is used for essentially keeping a set of keywords for this row.

The information of the text data dictionary is stored in several tables which belong to the text DBA account called SYSTEXT, and which are used for keeping track of the different text users, text tables, and columns and text indexing
options of the different text tables. All these tables are maintained automatically by SQL*TEXTRETRIEVAL as shown in Fig.2.

The information needed in order to create the text indexes and store them is kept in three tables for every different text table. The word list table contains all the words and keywords required for text indexing, the location list table contains the position in the text table for each word in the word list, the wordlink table contains the thesaurus information. These tables are maintained by the SQL*TEXTRETRIEVAL indexing tools.

3.3. LOADING AND INDEXING TEXT

The text has to be indexed before it is accessible to the retrieval system. It is possible to index text while loading it: STRLIU (Load and Index Utility) and STRLTT (Load Text Table). It is also possible to index already existing tables: STRGRU (General Reindexing Utility) and STRIUI (Incremental Indexing Utility).

The difference between the two lies in the ability of STRLTT to load also non-textual columns with the price that the input data has to be in a very special format which is not necessary with STRLIU.

STRGRU indexes all the rows of a given text table while STRIUI indexes only those rows whose text column has been updated since the last indexing of the table.

SQL*TEXTRETRIEVAL provides a range of options that permit users to generate the required applications and contribute to retrieval and text indexing efficiency.

Indexing options include the choice of indexing mode, case sensitivity of the text, number handling, delimiters of keywords if they are to be used and to enable proximity searching.

The indexing mode can be STOP or PASS: in STOP mode all the words that should not be indexed are included in the word list as stop words ‘S’ before indexing (the table STOPLIST owned by SYSTEXT contains the most common English stop words), during the text indexing any of these words found in the text would be ignored for indexing purposes. Words that are indexed will be added to the word list as free words ‘F’. In PASS mode only the keywords required to be indexed are placed in the word list as pass words ‘P’. During the text indexing all other words will be ignored and no word will be added to the word list saving therefore processing time and space but losing retrieval flexibility.

SQL*TEXTRETRIEVAL, as we have seen before, lets users specify keywords in a special column of the table in order to classify items against a controlled vocabulary of keywords. In STOP mode the keywords are included first in the word list, and the text indexing utilities include in the location list only references to keywords already present in the word list. In PASS mode the keywords that are not already present during text indexing are inserted in the word list and an informative message appears.
3.4. RETRIEVAL LANGUAGE: EXTENDED SQL

SQL*TEXTRETRIEVAL provides a retrieval language that consists of the simple addition of the CONTAINS clause to the SQL language. Queries can be performed on text and on structured data base fields.

For example, we can formulate the following text query:

```sql
SELECT * FROM TEXT_TABLE
WHERE AUTHOR='G. ORWELL'
AND TEXT_COLUMN CONTAINS 'DATA'
```

STRRMU (Retrieval and Maintenance Utility) is the user interface supplied by SQL*TEXTRETRIEVAL for carrying out ad hoc queries.

When a text query is executed SQL*TEXTRETRIEVAL stores the information retrieved in one table belonging to SYSTEXT called the hitlist table in order to speed up further requirements on this query (e.g. displaying text, editing text, ...).

3.5. APPLICATION DEVELOPMENT

SQL*TEXTRETRIEVAL offers the data base application designers the ability to develop applications in SQL*FORMS using a set of user exits, or by writing PRO*C programs using an equivalent set of ‘C’ functions.

These user exits and functions allow the designer the manipulation of text queries (create, edit, drop, load, refine, save, ...) and their execution and also allows him the displaying, creating, editing, deleting and printing of text columns as well as initializing and shutting down the system.

The application described in this paper has been developed in SQL*FORMS using the user exit toolkit mentioned above.

4. DATA BASE DESIGN

4.1. CONCEPTUAL DESIGN: ENTITY/RELATIONSHIP MODEL

Like most documents the short write-ups of the CERN Program Library PLWUP include well defined structured information e.g. author, submitter, revision date, common blocks, ... coexisting with free unstructured textual information e.g. introduction, description of the method, examples, ... This design attempts to extract first all the structured information of the write-up as we see in Fig.3.
In a second step the textual information had to be modelled. An SGML document has always a tree structure where each tag except the very first one has a father tag and could have children tags. In our problem we first cut out the structured leaves of the tree and afterwards, for efficiency reasons and because of the fact that the second level of the tree is well defined for all the write-ups, we decided to go one level down in the tree.

Therefore it was decided to create an entity for the tags that can appear in the second level of the tree. This entity is exactly the same for all these tags, and we show it in Fig. 4 as a generic entity PLW_TAG_ITEM, representing the 14 different tags in the second tree level.
4.2. LOGICAL DESIGN: RELATIONAL MODEL

Now that the conceptual design has been described we can examine the resulting relational schema shown in Fig.5.

The word list, location list and wordlink tables of all the text tables PLW_TAG_ITEM have to be included in the data base. To each text table a column called text_key has to be added for the internal use of SQL*TEXTRETRIEVAL.

The column PLWUP_KEYWORD is the CHAR column containing the keywords for each write-up and is the one used by SQL*TEXTRETRIEVAL when indexing. The column TAG_TEXT is a long column containing the text introduced by a write-up's tag.
5. DEVELOPMENT OF AN INTEGRATED WRITE-UP MANAGEMENT SYSTEM

5.1. ARCHITECTURE

The architecture of the application follows the idea of the standard SGML application described in Chapter 2. In a first step the SGML write-up files edited with a standard editor are inserted during the parsing process into the data base already described. The layout files are also created during this procedure.

On top of this data base, there has been developed a menu driven application in SQL*FORMS (Fig.6.) which permits the user to manage the structured and unstructured information of the write-ups. The structured information can be inserted and updated with the help of SQL*FORMS itself.

SQL*TEXTRETRIEVAL is used for constructing the queries on textual information that allow the users to retrieve rows with certain keywords chosen by them.

SQL*REPORTWRITER is used within the application in order to generate the SGML files with the topic indexes by write-up name, user entries, library objects or keywords.
The indexing utility of SQL*TEXTRETRIEVAL STRIIU is run before generating the topic index of keywords in order that the location list necessary before this report is created.

![Application Architecture Diagram]

**Fig. 6. APPLICATION ARCHITECTURE**

Two user exits in PRO*C have been written to cover two other aspects of this application: for removing a document from the database and for generating the SGML file for a specific write-up.
5.2. SELECTING A DOCUMENT

5.2.1. CREATING AN SGML FILE FOR A DOCUMENT

One of the most important aspects in the development of this application was the generation of the SGML file of the write-ups from the data base tables. It was decided to use a precompiler because of the possibility of creating SQL statements dynamically and in particular C for its advantages in managing files.

The reconstruction of the structured parts of the write-up is immediately achieved by means of the host variables that contain the data base columns. The reconstruction of the textual parts is done by executing the following dynamically constructed SQL statement containing a CONNECT BY clause:

```
SELECT * FROM PLW_TAG_ITEM
WHERE PLWUP_CODE=:PLWUP_CODE
CONNECT BY PRIOR TAG_SQ=TAG_SQ_FATHER
AND PLWUP_CODE=:PLWUP_CODE
START WITH TAG_TXT_TAG='TAG'
AND PLWUP_CODE=:PLWUP_CODE
```

This statement reconstructs the tree structure of the text and provides the necessary information for marking it up correctly.

5.2.2. UPDATING A WRITEUP

The utilities that SQL*FORMS provides for updating structured information are well known, so it is not necessary to emphasize them. The important feature of this application is that unstructured information can be updated as easily as structured data.

In order to update the textual information we have to find first the row in which the information is contained: in a first step the user is asked in which of the main tags the information that should be updated is held. What the user is actually telling the system is the text table in which the information is stored. In a second step the user is asked for a word that identifies the part of the text he wants to update or is given the possibility to search through the whole tag. In both cases the system constructs the text SELECT statement into an SQL*FORMS variable, calls the SQL*TEXTRETRIEVAL user exits to load the query, run it, display the text, search through the text and edit it if needed (for editing, it calls the operating system editor).

During the process described above the user has a view of the document in terms of tags or words, but never of rows or tables.

5.2.3. REMOVING A DOCUMENT

When a document is no longer valid or we want to insert a new version with the SGML parser, we will have to delete all the information concerning this document that exists in the data base. Otherwise we would have useless information in the data base and in some cases get duplicate rows.

For this purpose a PRO*C program has been written that removes from the data base all the information concerning a specific document. This program is called as a user exit from the application and it is important to remark that the
program doesn’t only delete the tables containing the structured information and the text ones, it has to update and delete the word, location, and wordlink tables of each text table. This is achieved by executing the following dynamically constructed SQL statements:

```
DELETE TEXT_TABLE_LOC
WHERE LOC_TEXT_KEY IN
  (SELECT ABS (TEXT_KEY_COLUMN)
   FROM TEXT_TABLE WHERE PLWUP_CODE=:PLWUP_CODE)

UPDATE TEXT_TABLE WORD
SET WL_FREQ= (SELECT COUNT (LOC_TEXT_KEY)
              FROM TEXT_TABLE_LOC
              WHERE LOC_WORDTAG=WL_WORDTAG)

DELETE TEXT_TABLE_WORD
WHERE WL_FREQ=0 AND WL_TYPE=’F’
```

### 5.2.4. INSERTING A DOCUMENT

Complete documents are inserted as we have seen by the parser, but in order to take advantage of the properties of SQL*FORMS it was decided to give the user the possibility to insert the structured information directly from the application. Afterwards the skeleton SGML file can be created from the application as seen before and the textual information can be introduced in the file with a standard editor. So, the users don’t have to go through the tedious work of marking-up the structured information which is done by the application.

### 5.3. REPORTS

In the front pages of the Manual the Program Library users can look for suitable program routines for their problems just consulting different indexes (by write_up code, by user entries, by keywords, by library objects, and by code and user entries). These indexes were generated from the set of flat files executing the above mentioned programs. As all information needed for the creation of the indexes is stored in the structured fields of the data base the generation of the indexes has just become the execution of very simple reports which enormously speeds up the process.

SQL*REPORTWRITER is called from SQL*FORMS via the following host command:

```
#HOST "RUNREP USERID=USER/PASSWORD REPORT=PLW_C1"
```

These reports generate the index files tagged in SGML ready for being processed by the composition and presentation system.

Before generating the topic index by keywords the location list and word list tables of the write-up table PLWUP have to be updated, therefore the SQL*TEXTRETRIEVAL utility STRIIU is invoked in a host command.
The queries below belong to the report that generates the index by keywords:

```
SELECT WL_WORD, WL_TAG
FROM PLWUP_WORD
WHERE WL_TYPE='K'
ORDER BY WL_WORD
SELECT PLWUP_CODE, PLWUP_NAME, PLWUP_TITLE, LOC_WORDTAG
FROM PLWUP, PLWUP_LOC
WHERE PLWUP_TEX=LOC_TEXT_KEY
```

In the first query all the keyword present in the word list are retrieved and in the second query which is a child query of the first one we get the text key of the to the keyword related rows from the location list.

6. FUTURE DEVELOPMENT

The guidelines for the future development of this application can be summarized in the following three topics:

- Development of an interactive querying system so that the users of the Program Library can decide the suitability of a package or find a suitable one without consulting the manual, but instead querying directly the data base.

- Expansion of the insertion module so that the textual information of the new write-ups can be also introduced directly into the data base without the help of the SGML parser. This module could be the design of a syntax driven editor that would interact directly with the data base.

- Extension of the data base structure so that different kinds of CERN documents, not only write-ups, could be stored and managed by the system.

7. CONCLUSIONS

It has been shown that SGML documents referring to a particular application, namely the CERN Program Library Manual, can be successfully stored and updated in an ORACLE data base with the help of SQL*FORMS and SQL*TEXTRETRIEVAL. This offers advantages for the document management and generation, as well as allowing special reports, e.g. a topics index, to be created easily. This technique is a general one which can be applied to other classes of document.
BIBLIOGRAPHY

(1) A.Berglund, CERN SGML User’s Guide, Version 86.1, CERN, Aug.87.


(3) E.van Herwijnen, Electronic submission of physics articles to publishers, CERN, Apr. 89.


