World Wide Web, 4: 5–20 (2001)

LAURIN: A Distributed Digital Library of Newspaper Clippings *

D. CALVANESE, T. CATARCI and G. SANTUCCI {calvanese;catarci;santucci}@dis.uniroma1.it
Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza,"
Via Salaria 113, 00198 Roma, Italy

Abstract

Among the wide range of digital libraries, an interesting, yet quite neglected, subclass is constituted by those exclusively dealing with newspaper clippings. Compared with book-oriented digital libraries, clipping libraries are more difficult to seize, since they are wide and unstructured, and the subjects and content of a clipping are completely heterogeneous. LAURIN is an EU-funded project involving seventeen participants from several countries, including two software companies and a large group of libraries, whose main purpose is to set up a network of digitalized newspaper clipping archives that can be easily accessed through the Internet, for searching and retrieving clippings. The project also provides the libraries with models and methodologies to be used for scanning, digitalizing, storing, indexing, and making accessible newspaper clippings. This paper concentrates on the main architectural features of the LAURIN distributed system, exposing the peculiarities deriving from the diverse users and tasks it supports.

Keywords: digital libraries, multilingual thesaurus, user centered design, distributed data replication

1. Introduction

During the last ten years Digital Libraries (DLs) have become an important and diffused information technology, with particular attention to book DLs and scientific document collections (see, e.g., [2,4,5,9–14]).

However, there is an important kind of physical library, namely the newspaper clipping collection, which had not got sufficient attention in the digital world (a notable exception is the Historical Newspaper Digital Library project [1], which deals with old clippings).

Compared with book-oriented digital libraries, clipping libraries are more wide and unstructured, since there are not specific standards to collect and classify newspaper clippings. The subjects and content of a clipping are completely heterogeneous: it could be a small article, some photos with some text as caption, or a whole article with diagrams and photos spanning several pages. Additionally, the information associated with clippings may differ from the usual one stored in digital libraries. For example, the author information, which is mandatory in traditional archives may be irrelevant or even missing for clippings. Also, users of clipping archives are typically interested in articles in the original form in which they appeared in the newspaper. As a consequence, not only the full-text of

* Work supported by the EU under the Telematics Program, Libraries Project LB-5629/A.

the clippings and the images possibly associated with the text must be stored in the archive, but also a picture of the scanned version of the original newspaper page, and obviously this requirement poses particular challenges with respect to data storage.

Given this diversity, it is extremely difficult to come up with a general clipping catalog system, whereas many specific clippings can be retrieved in a library that is systematically interested in some subjects or in a library that institutionally collects and catalogs newspapers.

LAURIN (Libraries and Archives Collecting Newspaper Clippings Unified for their Integration into Networks) is an EU-funded project[1] involving seventeen participants from several countries, including two software companies and a large group of libraries that want to make it easily available and give a wide visibility to the large cultural heritage from which they collect and catalog daily. The high number of users/libraries involved in this project gives the opportunity to spread culture and information to a wider public by means of the Internet. LAURIN has two major goals:

1. To set up a network of digitalized newspaper clipping archives that can be accessed through the Internet in a centralized fashion, for searching and retrieving clippings.
2. To provide a generic model to be used by individual libraries for scanning, digitalizing, storing, and indexing newspaper clippings, and making them accessible via the LAURIN network.

Concerning objective 1, since many users are ill equipped to translate their search requirements into precise queries, and they often prefer to use browsing as retrieval strategy, the LAURIN interface will offer, besides traditional keyword based search methods, also the possibility of browsing the clipping collection by argument, organizing the document space in a manner that is readily understood by users. Such an activity is supported by the use of an *integrated multilingual Thesaurus*, which plays a central role in the LAURIN system. The user sees a unified search space and therefore s/he can ignore the existence of different information sources, i.e., libraries. However, s/he is also allowed to select a library on demand, based on the description of its characteristics, in order to restrict her/his attention to specific topics covered by a certain library only. Requests can be formulated in any of the languages supported by the system (currently English, French, German, Italian, Norwegian, Spanish, and Swedish) and the system will provide translations for the purpose of keyword and content based search.

To fulfill the above requirements, the LAURIN system is organized around a *central node*, which is connected via the Internet to a set of *local nodes*, one for each participating library. The digitalized clippings and their full-text (obtained via optical character recognition (OCR)) are stored in the local nodes, together with a local, possibly personalized, copy of the Thesaurus. The central node contains indexing data about all clippings stored in the local nodes, and a centralized copy of the multilingual Thesaurus with globally validated entries. A constant flow of information from the local nodes to the central node ensures that the latter is up to date.

Concerning objective 2, the integrated Thesaurus system supports librarians in indexing and handling the clippings. This facilitates both the librarians' archiving activity and an improved local access.

The LAURIN project, aiming at producing a highly interactive system, is being carried out by following a rigorous "user-centered" design methodology [8], so that the envisioned solutions are really based on the user needs and requirements. This kind of approach is particularly appropriate for LAURIN given the large number of libraries involved in the project, playing the double role of end users and test sites. Also, it is worth noting that librarians are "extremely expert" users in their application domain (i.e., libraries and archives, books and journals). For instance, such users have their very precise idea of what a digital library is, and do not accept something different from computer scientists. From a librarian point of view, a digital library is very different from an XML repository! Also, librarians are very familiar with classifications, thesauri and taxonomies. They usually have their own classifications and do expect something "better" from information technology, but very often this does not seem to be the case.

Among other things, librarians involved in LAURIN have stressed the importance of having an indexing system and especially a thesaurus reflecting both their requirements and the needs of people who want to access the clipping archives to retrieve information of interest. Up to now the lack of structured thesauri, supporting a semantic classification of clippings, has prevented final users from accessing the clipping libraries themselves. What usually happens (at least in the many European libraries we have analyzed in LAURIN) is that the user asks a vague query to a "human interface" (i.e., the librarian) and s/he first tries to refine the query (for instance, enlarging or restricting it) and then searches for the clipping potentially matching the user's interests in the archive. This is obviously a very time-consuming activity and can be carried on only through a physical interaction (or, at least, phone-based interaction) between the librarian and the user. It is very difficult to replicate the same pattern on the Internet, where, on the other hand, the user could have a remote and universal access to all available digital libraries of clippings.

In order to realize an Internet-based service which aims at replicating at least the efficiency (even if it could never get all other qualities of a human–human interaction) of the librarian-mediated retrieval, the LAURIN project concentrated on two crucial components of the system, namely the multilingual thesaurus and the visual interfaces (both the indexing interface for the librarian and the retrieval interface for the end-user).

In this paper[2] we focus on the system architecture, while the project methodology, the users' tasks and the user interfaces have been described in [2], and we will just recall them in the following.

Comparing the LAURIN approach with existing literature, one may note that during the last years, digital library systems have not made many efforts to solve user-interaction problems. Only recently, new projects (e.g., University of Stanford[3] and University of Michigan[4] DL Projects) are developing a more complex model of information-seeking tasks. Display of information, visualization of, and navigation through large information collections, as well as linkages to information manipulation/analysis tools can be identified as key areas for research.

Other recent proposals deal with multi-language access to digital libraries and archives; integration of many different services, where information search is just a subpart of a more complex task; and easy refining of results and revisiting of search process. For example, the expansion and refinement of queries based on lexical relationships between documents,

which are automatically extracted from the document collection, is addressed in [3]. A prototype implementation of a general user interface paradigm which is capable of modeling iterative query refinement is described in [6].

Finally, another key issue addressed by the LAURIN project is the distributed nature of the collection of clippings. A distributed query system for preexisting library catalogs and structured databases (storing bibliographic data), based on an *ad-hoc* query language, has been also developed in the HARP project [7].

The paper is organized as follows: Section 2 recalls the LAURIN classes of users and their main tasks supported by the system; Section 3 introduces the overall system architecture and details its main modules; finally, Section 4 draws the conclusions.

2. Users' tasks and system functionalities

The LAURIN system provides different functionalities to two classes of users, namely internal and external users. Generally speaking, the users' activities are supported by the LAURIN multilingual thesaurus, whose structure is also briefly described in this section.

Internal users are part of the library staff who operate on the system to accomplish the following tasks: (a) to ask queries (in this case they embody the role of external user); (b) to input clippings; and (c) to administer the system. The main task of the internal user is clipping input, that is, scanning, OCR-ring and cataloging of clippings. This activity is performed only on local nodes. Some internal users, playing the role of system administrators, are also allowed to deal with the inner part of the Central Node. In particular, the system provides an interface to periodically validate new Thesaurus entries, coming from the local node clipping classification.

External users are users who access the system, independently from the location of the nodes, to submit a query and, hopefully, get an answer. The most general query is supposed to be formulated as follows: give me all clippings about *something*. The "something" part must be defined in a way that produces valid results (low noise in results), which can be incrementally refined, and must be simple to define by an average user (not extremely expert on the clipping collection or "casual").

2.1. Internal users' activities

Internal users perform their activities related with indexing and storing clippings only on local nodes through an *ad-hoc* interface to a sophisticated OCR system.

Several indexing mechanisms, which have been decided in strict cooperation with the librarians, are available. In particular, the Prime Index is the basic information on clipping/article that otherwise will be lost during the clipping process (name of newspaper, page, rubric, date, ...). The Bibliographic Index contains the basic bibliographic information on clipping/article (author, title, subtitle, text type of an article). The Keyword Index is an association of known terms from the Thesaurus with clipping/article which is automatically generated from the article full-text. The Content Index is an association of clipping/article with normalized terms from the Thesaurus resulting from a human content

analysis of the clipping/article. The Free Index is an association of clipping/article with subject headings that are not part of the Thesaurus resulting also from the human content analysis. Indeed, it may happen that, while indexing a clipping using the thesaurus concepts, a librarian is not able to find a thesaurus entry satisfying her/his needs. In this case the clipping acquisition module allows for associating the clipping with a *new* (*candidate*) *concept* that is in the Free Index. Candidate concepts are locally available for query formulation and are candidates to become new entries in the Thesaurus. The Full-text Index is a computer based retrieving of all normalized terms in the clipping/article (including terms that are not in the Thesaurus), and is generated and maintained by a full-text information retrieval engine.

The above indices are used in developing different clipping classifications, which are in turn exploited by the search mechanisms the external users are provided with.

Local node Thesaurus Administrators are special internal users, whose main goal is to administrate the local node Thesaurus. They typically update the local node Thesaurus with information associated with new clippings. The Thesaurus is queried and/or browsed to find relevant entries that can be associated with a clipping. Whenever an entry that is already in the Thesaurus needs to be associated with a clipping the association is stored in the local node database and transmitted to the central node together with the clipping data. Candidate entries, i.e., entries coming from the Free Index that are not in the Thesaurus, are analyzed, inserted in the local Thesaurus, and eventually associated with the clipping. The candidate entries are transmitted to the central node for validation and also kept in the local node (together with their association with clippings) until validation is performed.

Central node Thesaurus Administrators have two main tasks, namely to build, refine, and modify the Thesaurus and to validate candidate entries. The former is an off-line activity, that alters the Thesaurus content, independently from the activity of local nodes (e.g., correcting errors, adding new terms for existing concepts, etc.). The updates resulting from such an activity are propagated towards local nodes. The latter is part of the routine LAURIN job, and implies the analysis of the candidate entries coming from local nodes, which may be inserted in the Thesaurus, merged with existing entries or even rejected.

2.2. External users' activities

External users interact with the central node and the system provides (on demand) a description of the LAURIN consortium and of the involved local nodes, allowing a direct connection to local nodes hosting a Web query interface. If a user wants to ask a query across two or more local nodes (all nodes as an extreme case) s/he interacts only with the central node that acts as a broker with respect to the local nodes. Also, an external user connected to the central node can browse the central Thesaurus to search for associated clippings. Using several kinds of interfaces the user is allowed to formulate a multilingual query in which the Thesaurus plays three different roles:

1. It is a guide to understand the *classification* of the clippings stored in the LAURIN distributed database.
2. If the user has requested a multilingual search it translates the involved terms.

3. If requested by the user, it can be used to modify the scope of a query (e.g., finding not only the clippings containing the word X but also the clippings containing a synonym of X or a more specific term for X). The system provides the user with a Thesaurus browser, allowing for hierarchical navigation among terms. As an example, the user is able to select the location "Rome", either using the alphabetical order of "Rome" within a subset of the geographical Thesaurus data, or following the path "Earth → Europe → Italy → Rome". Every domain is multilingual, that is every concept is translated in the corresponding word in every language involved in the project.

Summarizing, when keywords are used in a query, the Thesaurus is accessed to expand the set of keywords according to the user specified criteria (more general terms, related terms, terms in different languages, etc.). For keyword expansion the Thesaurus of the node to which the user is connected (either central or local) is used. The identifiers of clippings associated with the expanded set of keywords can then be retrieved and presented to the user.

When the query has been processed, the user can interactively refine the result. When s/he has reached her/his goal, s/he can ask the system for a summary of the results, containing all the necessary information needed to get the clippings (involved nodes, cost, etc.).

2.3. Thesaurus structure

The LAURIN multilingual thesaurus is presently implemented as part of the overall database comprising several other data sources, namely *clipping data*, *periodical data*, *author data*, and *administrative data*.

In the database, thesaurus entries constitute the class "CONCEPT", and are related to clippings, languages, categories of entries (i.e., Persons, Institutions, Organizations, Companies, Geographical Locations, Keywords, Events, Actions, Properties, Time Keywords), and especially to other thesaurus entries through various relationships. A limited number of thesaurus entries are chapter headings, they represent the up-most entry points for the thesaurus browser. Several relations are defined between entries. For instance, if X and Y are entries, we may assert that X is-a Y, or X is-part-of Y, or X is-associated-to Y. If Y belongs to the category Geographical Locations, we may state that X is-located-in Y. If both X and Y are geographical locations, we may assert that X is-geographic-parent of Y. Other specific relations can be defined on terms of certain categories, such as persons working in institutions, facts happening at a certain time, etc.

These relationships are exploited by the Thesaurus Browser, which is part of the LAURIN user interface.

3. System architecture

The overall LAURIN architecture, depicted in Figure 1, foresees a network of nodes connected through the Internet: one node for any participant library plus a *central node* col-

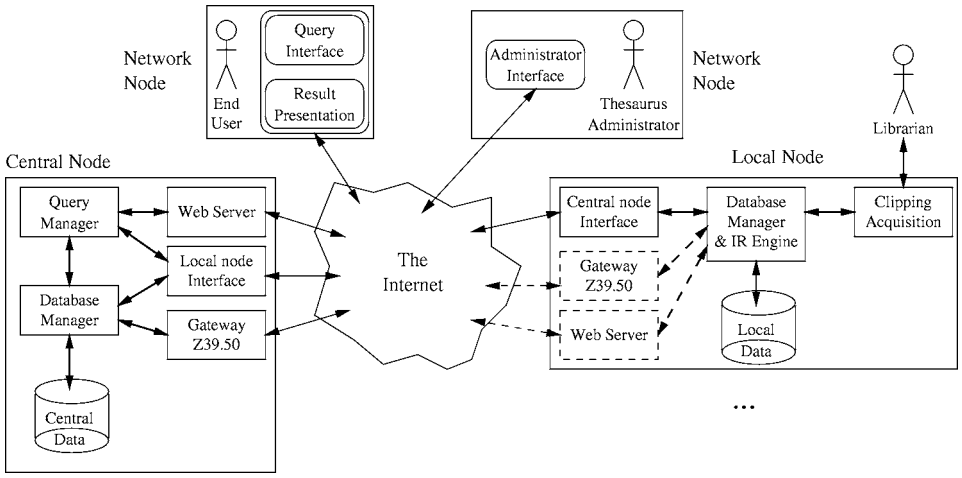

Figure 1. The overall LAURIN architecture.

lecting data from the local nodes and providing the end user with a uniform query environment. The central node hosts a relational database in which summary data coming from the local nodes are stored (i.e., clipping title, date, newspaper, author, etc.). Local nodes are in charge of clipping scanning and indexing; moreover they store all information about acquired clippings: summary data, full clipping text, and clipping images. LAURIN clippings are strictly related with the LAURIN Thesaurus that is stored in the central node and replicated in the local nodes. There is a constant flow of information from the local nodes towards the central node, updating the central database with new clippings and new thesaurus entries. The central node also contains a Z39.50 [15] interface, which allows for acting as a Z39.50 server, exporting all LAURIN summary data. Depending on local policy issues, each local node may be directly queried by the end users through the same interface used to query the central node and/or by a Z39.50 interface.

We now illustrate in more detail the overall system workflow and the various components of the architecture.

3.1. Data workflow

We can single out four main workflows across the LAURIN architecture:

Clipping acquisition. This is mainly a local node activity (performed by internal users) that involves the physical scanning of the clipping, the optical character recognition, the indexing of the clipping (according to the indexing criteria described in Section 2.1), and the clipping data memorization. The result of such an activity is a new set of clippings and a new set of candidate Thesaurus entries. This information is propagated towards the central node that updates its central database.

Thesaurus database maintenance. As we said above (see Section 2.1), it may happen that, while indexing a clipping using the thesaurus concepts, a librarian is not able to find a thesaurus entry satisfying her/his needs. In this case the clipping acquisition module allows for associating the clipping with a *new* Thesaurus entry (*candidate entry*) that is sent to the central node together with the clipping data. That implies that the central node handles a set of canonical Thesaurus entries plus, for each local node, a set of candidate concepts. Candidate concepts are available for query formulation as soon as they reach the central node but are *not* replicated on local nodes. Periodically, the Central node Thesaurus administrators inspect and validates such concepts. Once a concept has been validated it is sent across the network to all local Thesauri. A full handshake protocol is adopted in this phase to avoid inconsistent clipping classification.

Clipping database maintenance. This activity involves a local node and the central node. With a timing that depends on the local node constraints (rate of clipping acquisition, urgency, speed of connection, etc.) the summary data of the new scanned clippings plus their connections with Thesaurus concepts are automatically sent through the network to the central node that updates its clipping database, keeping track of the clipping owner and making this new information available to the end user. A simple but effective general agreement about clipping IDs allows for avoiding collisions of clipping keys. In order to cope with the complex semantics associated with the Thesaurus entries and their validation process discussed above and to build a system as open as possible (i.e., to allow the presence of different DBMSs and different information retrieval engines) the replication process of both clippings and Thesaurus entries is performed by a specific module, called *Delta Replicator (see below)*.

Distributed query processing. Once the user has formulated a query, the central node query manager answers it as follows. First of all, it analyses the query, splitting it into two parts, one related to the central node database and one related to the local nodes (i.e., the part of the query that refers to the full text of clippings). To solve the former it starts a query against the central database; to compute the latter it selects the local nodes that may possibly contribute to the query (e.g., to look for an Italian clipping at the Uppsala node in Sweden makes no sense) and then sends the query to the selected local nodes. Once each local node has returned a list of clipping IDs the query manager merges such lists with the answer it got by the central database, presenting the final result to the end-user.

3.2. Delta Replicator

The replication module has two main objectives: *replication*, to transfer new pieces of information between local nodes and central node (so to let them be always in a consistent status), and *Thesaurus concept alignment*, to manage the entire process of validation and alignment of concepts between local Thesauri and central Thesaurus.

It is worth noting that the Delta Replicator is open and generic enough to be used in any project requiring data replication across distributed nodes. A customized version of such a replication module has been *ad-hoc* developed for the LAURIN project.

3.2.1. Delta definition and replication phases. With "Delta" we mean the portion of a source node database (say X) that has been changed since the last execution of the replication and alignment process with another node (say Y). The Delta contains those records of the X database that have been added, modified, or deleted, and must be sent from X to Y to guarantee the consistency of the two databases.

The replication process encompasses the following phases:

- *Delta extraction*. This operation is performed on the sender node X (producer) through the following steps: (a) extracting the Delta from the database, (b) packing and compressing it in a suitable data structure, (c) inserting the data structure in a queue, and (d) sending the queue items to the receiver node Y (consumer).
- *Delta processing and handling*. This operation is performed on the consumer node Y that gets from the network the compressed data, expands them, modifies the database according to the information stored in the Delta, and finally sends an acknowledgement to X.

To ensure data consistency, during the replication and alignment operations, each local node must receive and process the central node Delta before sending its own Delta to the central node.

The chosen Delta identification method is independent of the database structure and uses the notion of timestamp. The idea is to add two columns to all tables involved in the replication process, in order to store, for each tuple, both the creation date and the date of the last update. The date of the last replication process is also saved. This method requires a little more work with respect to other proposals available in the literature but allows a more flexible Delta identification policy. Basically, it allows to manage the different situations which may occur with several nodes. Indeed, each local node has to receive a different Delta from the central node. Such a Delta is identified by looking at the replication date. For instance, if a local node is off for a month, the system can continue to replicate data on the other nodes then, when the node returns active, the entire missing Delta is sent to it. This is very important since it is difficult to guarantee the availability of all network nodes all the time.

3.3. Central node

The main components of the central node are:

Web Server. A standard HTML server servlet compliant providing any remote user using an HTML browser with a simple and uniform access to the LAURIN clipping archives. A servlet is activated on the central node interacting with the *query manager* and presenting the final result to the user. In case the user wants to fully exploit the LAURIN system, s/he can download a Java application providing more complex query functionalities. Such application interacts with the query manager using the same communication protocol adopted by the servlet.

Database Manager. This component provides, through the Oracle DBMS, the storage and retrieval of all summary clipping data plus Thesaurus data. Moreover, the Oracle ConText Cartridge is used to search the textual part of the database (i.e., title and abstract).

Z39.50 Gateway. This component provides the access to the centralized clipping data through the Z39.50 standard.

Query Manager. This component acts as a server for the query clients connected to the central node (both HTML and Java connections) and analyses the incoming queries distributing them across the involved local nodes. This is the most critical part of the central node and its tuning and optimization had been a key issue of the project. The following subsection will analyze in detail such a component.

Local Node Interface. This component handles all communication among the central node and the local nodes, including: update of the central node clipping database with new or modified data coming from local nodes, update of the central node candidate Thesaurus entries with data coming from local nodes, propagation of validated Thesaurus entries towards local nodes, and distributed query processing on one or several local nodes. In the latter case the data flow is bidirectional: the central node issues a query and the local nodes provide answers.

3.3.1. The Query Manager. The Query Manager, depicted in more detail in Figure 2, is composed of two main modules:

- the *communication module*, which handles all issues associated with transmitting on the network both queries and results;
- the *distributed query elaboration module*, which analyzes and computes the queries in the LAURIN distributed environment.

The communication module is separated from the elaboration process by four decoupling queues and is composed of:

- Three listening servers (on their ports) which wait for incoming requests of TCP connections and activate their respective reception units, namely:
  1. *Query Server*, which receives queries, through asynchronous connections, from clients and, in the local node case, from the central node;
  2. *Query Synch Server*, which receives queries, through synchronous connections, from clients. In order to increase the overall efficiency, this unit does not close the TCP sockets used for the synchronous connection but stores them for sending back the results;
  3. *Result Server*, which receives results through asynchronous connections from local nodes.
- Two forwarding units which draw data from the respective queues and activate one sending unit, which, as the reception units, uses Java serialization and network support features:

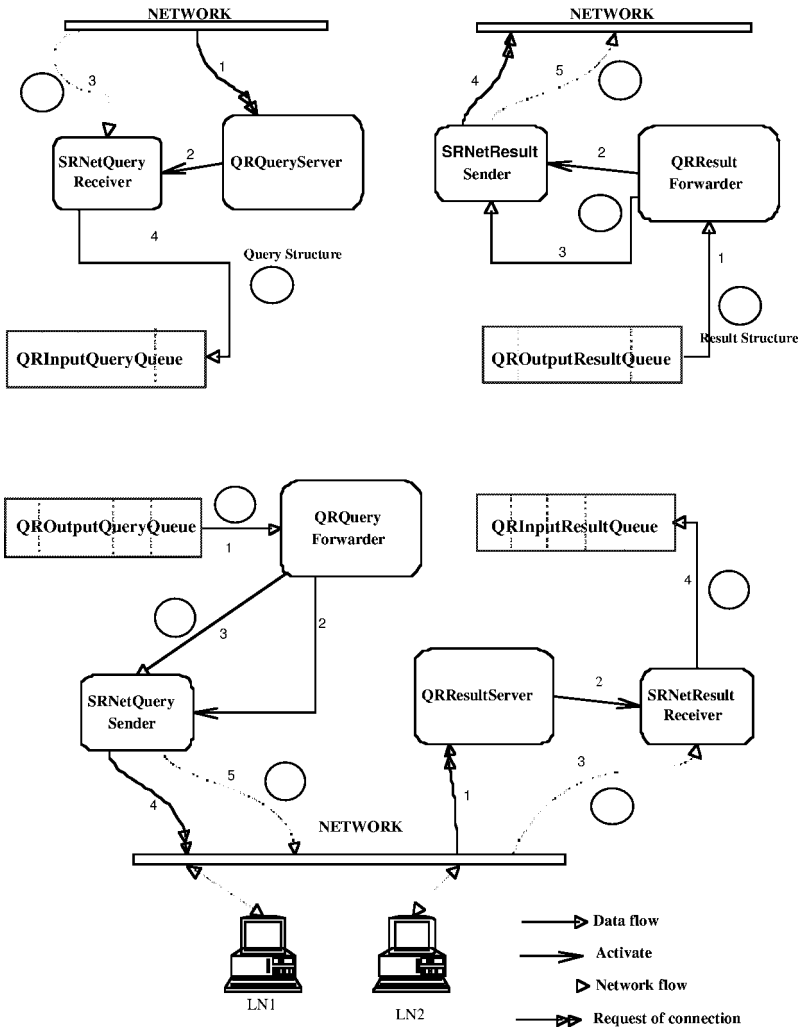

Figure 2. The structure of the Query Manager.

1. *Query Forwarder*, which distributes the query, through asynchronous connections, to the involved local nodes;
2. *Result Forwarder*, which sends results, through synchronous and asynchronous connections, to the clients and, in the local node case, also to the Central Node. This unit reuses the TCP sockets stored by the Query Synch Server.

In the following we detail the communication flow: the QRCentralNodeQueryRouter draws query structures from the InputQueryQueue and, for each of them, creates a new QRCentralNodeQueryContext that represents an active context of elaboration (one for each query) and starts, among others things, four active objects:

1. the QRCentralNodeQueryProcessor loads the requested services, which complete the query, and suitably activates the other active objects according to the query strategy;
2. the SRJDBCSenderReceiver links via JDBC to the local DB, executes the SQL query, received from the loaded service, and puts the results in the CNQueue associated to the local DB;
3. the QRQueryDispatcher dispatches the query to the involved local nodes (if any) and creates a queue in the QueryContext for each of them in order to receive their results;
4. the QRResultCollator collects the results coming from the local DB and, in the Central Node case, from the queues associated with the involved local nodes; all such results are put in the OutputResultQueue, and then they are sent, to clients or local nodes, by the QRResultForwarder.

Till now we have examined the server from the point of view of the query elaboration process, in the following we discuss how the active objects are managed in order to optimize such a process. There are two types of active objects: some of them have just a single instance in the server because they always execute the same activity; other ones can have several instances in execution at the same time because each instance belongs to a different query context. Each of these tasks has to be executed along with others tasks belonging to the same query context, to process the associated distributed query. In order to avoid the case that the parallel execution of a large number of distributed queries exhausts all resources of the machine hosting the central node we implemented a simple but effective way to handle an increasing number of parallel requests.

Tasks are stored in different collections (of type QRTaskCollection): one collection is for QueryProcessors, another one for QueryDispatchers, etc. The QRCentralNodeQueryContext requests one instance (one task) of any type of resource; once these instances are all available the query elaboration process starts, activating the QueryProcessor.

Each activation of a task corresponds to its insertion in a specific queue; there is a pool of special threads that draw the tasks from the queue and finally execute them (the tasks are released after their termination).

In this way we obtain the following advantages:

- the (at least partial) parallel execution of tasks belonging to the same Query Context (the order of the activation sequence is enforced by the queue);
- the predetermined allocation of resources (it is possible to scale the server just by changing the dimension of the pool), thus avoiding the garbage collection work of the JVM (the memory allocation of the process is fixed);
- a limited number of threads always in execution. Such a number is given from the sum of the (also limited) communication threads plus the number of threads having a single instance, plus the ones in the thread pool;
- the total control, through the thread pool, of the execution flow of any task and error-recovery with release of resources.

A manual stress test was performed using up to ten clients to issue queries against the server in a burst fashion. Every client started five queries and the overall query rate was about 30 queries per second. The stress test lasted about 30 seconds obtaining a burst of 900 queries in 30 seconds.

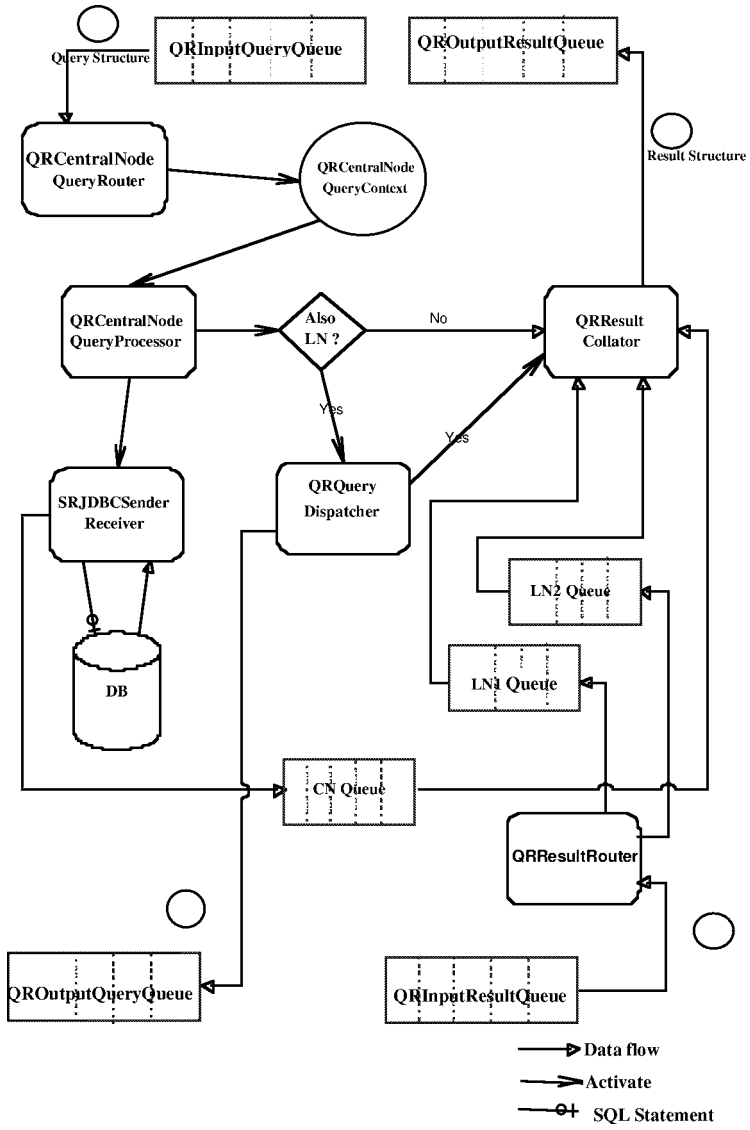

Figure 3. The structure of the Query Server.

These figures correspond to the (really pessimistic) hit rate of 1,296,000 queries per day, assuming that the server is operative 12 hours a day.

The application server handled the test in the following way:

- the first 10 queries were executed in parallel, as the capacity of PC hosting the server PC and the free resources available (thread and memory) allow for performing up to ten

queries at the same time, without significant time response difference compared with normal load conditions (one, two queries per second);
- the remaining queries were queued in the application system queue and spooled as soon as the needed resources were available;
- when the number of incoming queries exceeded the capacity of the input queue, remaining queries were simply rejected, avoiding the application server hanging or crashing.

In this way the response time experimented by the clients degrades gracefully under load pressure and, being every parameter of the internal architecture (e.g., number of thread, number of input queue positions) easily configurable, the server can be scaled based on the number/power of processors and available memory.

3.4. Local node(s)

The main components of a local node are the following:

- *Central Node Interface*: this component handles all communication with the central node including: sending new or modified clippings to the central node, sending new candidate Thesaurus entries to the central node, accepting from the central node validated Thesaurus entries, accepting a central node query based on the full text clipping content, and answering a central node query (i.e., sending the list of clipping IDs satisfying the query).
- *Web Server* (optional): a standard HTML server providing remote users using an HTML browser with a (customized) access to the local node clipping archive. The local architecture is as that of the central node, so the same interface and the same software (applet) are used in both central and local nodes.
- *Z39.50 Gateway* (optional): this component provides the access to the local clipping data through the Z39.50 standard.
- *Database Manager & Information Retrieval Engine*: this component, based on the Oracle DBMS and Oracle ConText Option Cartridge, allows for storing all clipping data (text, images, and summary data) plus the Thesaurus data. This module is able to answer in an efficient way text queries on clipping content.
- *Clipping Acquisition*: this module allows local librarians for scanning and indexing new clippings. The acquired data are stored locally and the central node interface provides updates to the central node.

Note that Figure 1 describes these components in a logical way: as an example, the Web Server and the Database Manager may be physically hosted by two different computers.

3.5. System implementation

A first prototype of the LAURIN system has been implemented under the Windows NT operating system, using Jbuilder 2 with Java 1.1.7 and Oracle 8. All modules foreseen for the Local Nodes have been implemented and installed at each participating library site.

The Local Node system includes a first version of the Thesaurus as well, which contains a large set of geographical data extracted from the TGN thesaurus, and a set of names of famous people (artist, writers, etc.), together with a predefined set of relationships among concepts. In this very moment librarians are going on clipping and indexing articles, filling the thesaurus with new concepts.

A first version of the Central Node has been implemented as well, providing the main query functionalities. In particular, the system allows for querying the whole LAURIN system through a form-based interface capturing the user requirements. The interface is based on a set of panes allowing for expressing queries characterized by increasing complexity. The Simple Search pane allows for retrieving clippings through subjects, newspaper names, date, paper and clipping types. Moreover, the user is able to restrict the search space by launching the query on a subset of the LAURIN libraries. While interacting with a query pane, the user can change the interaction modality to ask more complex queries, such as looking for clippings written in a certain language and about people and places. A continuous feedback about the query is provided, allowing the user to totally control on the whole process.

The communication protocols among the Central Node and the Local Nodes (i.e., clipping and Thesaurus database maintenance) have been implemented and their testing has been initiated, already giving encouraging results in terms of performances.

4. Lessons learned and future work

In this paper we presented the key architectural aspects of the LAURIN system. LAURIN is the first project which specifically deals with clipping databases, with the final goal of building a large European network of libraries hosting clippings, which will allow users to easily find the newspaper articles (physically distributed over Europe) better matching their interests through a centralized Internet access.

One of the key aspects of the project is the presence in the Consortium of several libraries, which act as end-users. This has permitted to develop a truly user-centered design methodology and to test all design choices against real user requirements. We have discovered that the users of the LAURIN system are very conscious of their needs and have clear expectations (in particular, the librarians, with whom most of the tests were done). They have been exposed to several mock-ups of the interface and workflow simulations, and very often have reacted with constructive criticisms, which has led to several improvements of the system.

From the technical point of view, first of all we discovered that the native Java RMI (Remote Method Invocation), in spite of its elegance, was not suitable to handle several connections across a (usually) not reliable network. Therefore we decided to use the socket approach that gave us full control on each single connection (retry, timeout, etc.). As a second consideration, our application is strongly based on the use of threads, which are extremely well supported by Java. However, to slightly mitigate our enthusiasm about Java threads, we have to note that a high usage of threads makes the debug activity really

hard. Moreover, it seems that a modal dialog (e.g., one in which the system is waiting for an OK) stops all running threads.

Notes

1. Telematics Program, Libraries Project LB-5629/A, `http://laurin.uibk.ac.at/`.
2. A preliminary version of this paper appeared in the *Proceedings of the IEEE International Conference on Digital Libraries (ICDL)*, Japan, November 2000.
3. `http://www-diglib.stanford.edu/diglib/`.
4. `http://http2.sils.umich.edu/UMDL/`.

References

[1] R. B. Allen and J. Schalow, "Metadata and data structures for the historical newspaper digital library," in *Proc. of the 8th Internat. Conf. on Information and Knowledge Management (CIKM'99)*, 1999, pp. 147–153.

[2] D. Calvanese, T. Catarci, and G. Santucci, "Building a digital library of newspaper clippings: The LAURIN Project," in *Proc. of the IEEE Forum on Research and Technology Advances in Digital Libraries (ADL 2000)*, 2000, pp. 15–26.

[3] J. W. Cooper and R. J. Byrd, "Lexical navigation: Visually prompted query expansion and refinement," in *Proc. of the 2nd ACM Internat. Conf. on Digital Libraries (DL'97)*, 1997, pp. 237–246.

[4] J. Frew, M. Freeston, R. B. Kemp, J. Simpson, T. Smith, A. Wells, and Q. Zheng, "The Alexandria digital library testbed," *D-Lib Magazine*, 1996.

[5] T. S. D. L. Group, "The Stanford Digital Library Project," *Communications of the ACM* 38, 1995, 59–60.

[6] L. Kovács, A. Micsik, and B. Pataki, "AQUA: Query Visualization for the NCSTRL Digital Library," in *Proc. of the 4th ACM Conf. on Digital Libraries (DL'99)*, 1999, pp. 230–231.

[7] E.-P. Lim and Y. Lu, "HARP: A distributed query system for legacy public libraries and structured databases," *ACM Trans. Inform. Systems* 17(3), 1999, 291–319.

[8] D. Norman and S. Draper, *User Centered System Design*, LEA: Hillsdale, NJ, 1986.

[9] J. Ober, "The California digital library," *D-Lib Magazine*, 1999.

[10] V. Ogle and R. Wilensky, "Testbed development for the Berkeley digital library project," *D-Lib Magazine*, 1996.

[11] A. Peterson Bishop, "Measuring access, use, and success in digital libraries," *J. Electronic Publishing* 4(2), 1998.

[12] B. Schatz and H. Chen (eds.), *Digital Libraries: Technological Advances and Social Impacts*, Special Issue of IEEE Computer, IEEE Computer Soc. Press: Silver Spring, MD, 1999.

[13] N. A. Van House, M. H. Butler, V. Ogle, and L. Schiff, "User-centered iterative design for digital libraries: The Cypress experience," *D-Lib Magazine*, 1996.

[14] R. Williams and B. Sears, "A high-performance active digital library," *Parallel Computing* 24(12–13), 1998, 1791–1806.

[15] Z39.50 Maintenance Agency, "Information retrieval (Z39.50): Application service definition and protocol specification (ANSI/NISO Z39.50-1995)," Z39.50 Maintenance Agency, 1995.