# Not Just Another Database Project: Developments at UW

*Frank Wm. Tompa*

UW Centre for the New OED and Text Research
Department of Computer Science, University of Waterloo
Waterloo, ON, Canada  N2L 3G1

fwtompa@uwaterloo.ca

## 1. A Short History of the New OED Project

In 1983, the Oxford University Press circulated a call for proposals to convert the *Oxford English Dictionary* into electronic form so that it could be updated and maintained, new editions could be published, derivative reference works could be produced, and online access could be provided. The following year a partnership was struck between the University of Waterloo and the Press, in which Waterloo would be responsible for helping the Press to design the New OED database, for helping to convert the dictionary into database form, and for creating database management utilities suitable for storing and manipulating the dictionary in that form.

Traditionally text is stored, accessed, and maintained in an *ad hoc* fashion, with no support similar to that provided by the discipline of the database approach. What was needed was to incorporate text-aware features typically found in information retrieval systems into the framework typical of a database management system, but supported by a suitable data model based on formal languages. Waterloo committed itself to design and develop the technology to address the needs of users and maintainers of the *OED*, in the expectation that these needs will be required for other large text databases. Thus the thrust of the project was directed towards the development of a database system that is capable of supporting the needs of text *creators* (such as the editors at the Press who maintain and enhance the *OED*), while simultaneously supporting the needs of text *users* (writers, editors, scholars) who will access such machine-readable texts interactively. The most visibly successful results of the project are embodied in flexible and efficient search and display software.

Initial funds to support Waterloo's efforts were provided by the University of Waterloo and by a grant from the Canadian Secretary of State through its "Centres of Specialization" program. From 1986 through 1989 the project was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) through a Cooperative Research and Development grant to investigate "Text-Dominated Database Systems"; this effort has been recognized by NSERC in its 1991 publication entitled *Great Canadian Success Stories*. Subsequent funding has been made available by NSERC through other research programs, by Open Text Corporation, and by Industry Canada.

Waterloo established the UW Centre for the New OED in 1985 to provide a forum for computer scientists, lexicographers, publishers, humanists, and social scientists to experiment together with prototype implementations of the software and preliminary versions of the data. The database system development group, thus, included software designers and implementers, users from the Press, users from within Waterloo's academic community, and short-term visitors to the Centre. This close interaction from the beginning has been key to the success of the project.

## 2. Data Design

### 2.1. Grammar-defined databases

The project began with an attempt to apply conventional modelling tools to characterize the *OED* data. Standard assumptions for business-oriented data were found to be inappropriate — even the fundamental concept of "entity" does not apply — and we therefore developed an important new grammar-defined model of data. In the same way that the relational model was developed based on experiences gained from earlier database approaches, the grammar-based model arose from traditional foundations of computer science, and particularly database theory and practice.

In terms of the model, a text-dominated database is described by a schema expressed as a grammar, which can be used to parse and subsequently represent a collection of data. A valid database instance, then, contains data that conforms to a given grammar. Thus, just as numeric data is structured in a business database, string data must be structured in a text-dominated database. Rather than taking the form of tables, hierarchies, or networks, however, grammar-based data takes the form of parsed strings, or ''p-strings,'' each representing a derivation tree for the corresponding text. A similar approach forms the basis of the Standard Generalized Markup Language (SGML), which has since become increasingly important in commercial text processing.

Going beyond SGML, however, p-strings are subject to alteration via a data manipulation language, for which we developed suitable operators forming a p-string algebra. The p-string model is representation-independent and the operators non-navigational, so that efficient implementations can be developed for unknown future hardware and operating systems. Over the course of the project, several approaches to storage structures and efficient processing algorithms for representative hardware configurations were investigated.

Prior to the widespread introduction of SGML-support software, texts were not constrained to match precise grammars. The *OED* complied to a sufficiently well-structured format that structural elements such as headword, part-of-speech label, etc. could be recognized and tagged, but no detailed grammar was available. Thus, the p-string model emphasized data instances and could even be adopted in the absence of a grammar.

To support truly grammar-defined databases, we began to develop algorithms to infer meaningful grammars from SGML-like tagged instances. The inference engine we implemented is based on equivalence of productions (such as $S \rightarrow A\ B\ C \mid A\ C \equiv S \rightarrow A\ B?\ C$) and a few user-supplied heuristics (such as "replace $S \rightarrow A\ B\ B\ C$ by $S \rightarrow A\ B{+}\ C$"). In addition, a prototype teaching mechanism was also implemented, through which a teacher, or grammar editor, can suggest meaningful productions to be included in the grammar inference process. In preliminary results we generated a grammar for the *OED* (without assistance via a teaching mechanism) having a size reduced by about 80% of the "*de facto* grammar" represented directly by the parse tree induced from the structural tags. For example the number of grammar rules to describe the uppermost structure of an entry in the *OED* was reduced from an unmanageable 1105 to 137, which is well within the limits of study by lexicographers. Using the preliminary result, we identified some interesting structures that appear within the dictionary, including several that were unanticipated by the *OED*'s principal editors.

In subsequent work, we investigated the role of SGML in data modelling, with particular concentration on the role of embedded markup. Data modelling *through* text and data modelling *of* text remain as ongoing research interests.

### 2.2. Text transductions

The problem of transforming text from one encoding to another is significant and recurrent in text database management. In fact, the first requirement of the project in 1984 was to convert dictionary data from its input format into a tagged form more suitable for further manipulation. Text transduction software was developed in 1985 to work with a previously-implemented program, known as INR, that converts grammars based on rational expressions into finite automata. Concurrently with its development, the text transduction system was applied to the *OED* on the Press's premises in Oxford, and within only a few months the technology was transferred to the Press.

In 1987, members of the Oxford University Press outside the New OED group expressed an interest in applying the technology to another domain: the development of the fourth edition of the *Shorter OED*, a distinct dictionary project. A major aspect of the project was to transduce data from the *OED* database to form a rough draft of entries for the *Shorter*. Because this task was more suitably addressed with a system based on parse trees than with one based on finite automata, algorithms were designed and implemented using the p-string algebra.

In subsequent years we developed additional high-level tools to aid users transducing text from one format to another. As part of this work we developed a stand-alone interactive language for describing and running text transductions using Mealy machines, a particular form of finite automata conducive to efficient execution. This work included the design and implementation of algorithms to construct deterministic finite state automata from regular expressions, and to construct Mealy machines from transduction specifications.

Resulting from our investigation of various tools that have been used at the Centre or elsewhere for transducer definition, we classified the tools by the power of the underlying formal model and evaluated the language support provided for users of the tools. Arising from this study, we plan to investigate approaches to transduction development that are based on interactively querying and modifying sample extracted data using the database's data manipulation language and subsequently compiling the database operations into automata-based transducers that can be executed on the whole data.

## 3. Text Search Software

### 3.1. GOEDEL

A data manipulation language was defined as part of the grammar-defined model during 1985 and a first prototype (named GOEDEL, after the most famous mathematician with "OED" in his name) was implemented as an adaptation of the kernel of the interpreter used in Waterloo's Maple Symbolic Computation System.

As GOEDEL was the first query language available to extract data from a structured text the size of the *OED*, visitors to the Centre for the New OED used it extensively. In fact, during his extended stay in Waterloo in 1985, John Simpson, co-editor of the *OED2e*, became the first expert user of GOEDEL outside the language designers and implementers. It was GOEDEL that later also allowed the "Shortening" project to be successfully completed.

In spite of repeated successful applications using this language, the implementation severely hindered more general use — both because of the inefficiency inherent in an interpreter and because of incomplete support of needed language features. To address this concern, in 1991 we re-designed and re-implemented the grammar-defined operators in the form of an embedded data manipulation language for `C`, which we dubbed GDL. Our approach was "object-oriented" and included support for lists, sets, tables, and strings, as well as p-strings. In addition to giving us a modular basis for software development, we expect this approach to continue to facilitate complex text manipulation.

Extended text search support was investigated to provide for efficient search based on a text's structure rather than merely on its content. A prototype implementation attempts to find an efficient strategy based on statistical information derived from the parse tree of the data (similarly to determining a suitable order to perform joins in a relational database). In preliminary testing, searching for rarely occurring combinations of structures in the *OED* has shown the promise of the methods.

### 3.2. PAT

Since 1987, efficient retrieval has been provided by the PAT text search engine. Each entry in a search index designates a ''semi-infinite'' string that starts at a critical point in the text (e.g., at a word start) and continues uninterruptedly to the end of the text; a separate index for each interesting set of text regions is also maintained, using a pair of semi-infinite strings to designate the extent of a region. PAT's regions are used in much the same way as records and fields in conventional database systems: as units for query, display, input/output, and concurrency control.

PAT has been used to answer queries ranging from "Which words in the *OED* are derived from German?" to "Which words have a first citation from Maclean's magazine?" PAT was also used in 1988 to help proofread the text of the *OED2e* to identify systematic errors (e.g. *obs.* or *Obs* in place of *Obs.*) as well as dated phrases (such as "the current Austro-Hungarian Empire" or "equivalent to 5 s. 3 d. today") within definitional text.

In subsequent research we extended the PAT search engine to include the capability for search based on regular expressions. To this end, we incorporated the language for specifying Mealy machines into the search engine, so that a PAT search view can be supported by Mealy machines. Regular expression searching is then supported by compiling a regular expression into a deterministic finite state automaton, composing the result with the search view Mealy machine, and executing the resulting automata over the PAT index.

In a parallel effort, we have also developed new space and time-efficient algorithms for approximate string searching. From within PAT a user can request a search for all strings that are within a distance $k$ (mismatches) from a given string — where mismatches can take the form of inserted characters, deleted characters, or replaced characters. These algorithms have been subsequently extended by Gaston Gonnet, former co-director of the Centre, to identify interesting clusters of protein sequences based on their standard string representations in large protein databases.

Finally, PAT has also been extended to operate in a distributed computing environment, wherein one PAT handler can serve as a multiplexor for several independently executing PAT processes. This approach has the potential to support very large, possibly heterogeneous, collections of text.

### 3.3. Adaptation for CD-ROM

We have investigated alternative data structures for a full-text index. PAT index searching requires $O(\log n)$ disk accesses to search a text of size $n$ (in practice, about 50-60 accesses for the *OED*). Although this gives acceptable performance using magnetic disks, it is not acceptable for slower media, especially CD-ROM. We have investigated the use of an auxiliary index containing a subset of the PAT index points, small enough to be stored on magnetic disk for fast searching. Index construction attempts to identify those points in the index most likely to be searched, e.g., word boundaries. Testing indicated that an auxiliary index about 10% the size of the *OED* index could reduce the number of disk accesses for a PAT search from about 50-60 to about 10-15.

Another new index for text, related again to the PAT index, has also been developed. Like the PAT index, start points for semi-infinite strings are sorted alphabetically, but rather than pointing to the text, each index point references the location in the index for the subsequent index point. This data structure reduces the number of disk accesses substantially for certain common searches and has great potential for compression of the index itself.

A third approach that shows great promise is based again on an auxiliary index for PAT, this one storing "signatures" of each referenced string. Through the use of a small look-aside table, queries for phrases up to five words in length can be answered with at most three disk accesses, making this index ideal for CD-ROM.

Significant results have also been achieved in one aspect of text search support that we had not originally anticipated investigating, namely text compression. The size of text database that we wish to support efficiently necessitates an examination of space reduction, while preserving full-text search capabilities. This requires a different approach from the currently popular adaptive (e.g., Lempel-Ziv) methods as it puts further requirements on the speed of decompression. In particular we need a relatively small static dictionary; fortunately the PAT index itself contains the information needed to construct such a dictionary. In fact, the compression achieved appears to be competitive with other methods, while satisfying the goals of fast decompression with a small static dictionary.

## 4. User Interfaces

### 4.1. LECTOR

So that highly formatted text could be displayed in response to user queries, the LECTOR display system was implemented in 1988. LECTOR is a general purpose browser that takes as input a stream of tagged text and formats it to the screen using typography to illustrate its structure. It uses a specially-designed formatting, or display-specification, language to accomplish this, through which the choice of typographical strategies is user-selectable. Used together, PAT and LECTOR form a powerful query and browsing facility for text-dominated databases, including the capability of supporting large hypertexts.

### 4.2. Hypertext approaches

Each system component was designed as a stand-alone utility with well-defined interfaces, such that they can be used independently or as subsystems of an integrated software package. Both a multi-window cross-reference resolution program for the *OED* and a program to match *OED* citations to bibliographic entries were implemented in 1989. Such demonstration applications involving several simultaneous views of a text resulted in the design of extended facilities to support hypertext-like linking:

In conventional hypertexts, links are explicit connections between specific regions of a text. We developed an architecture that treats links as the outcome of responses to user pokes. In this architecture, a hypertext is a collection of link-resolving components, each interpreting a user's request according to its own resolution algorithm. When a user points at some location within a window, each link-resolving component is given a key derived according to a previously-stored algorithm; the link-resolving components concurrently update according to their algorithms for resolving the keys.

The location of a poke does not uniquely determine a link's meaning. However, a potential target application can often decide the meaning of a poke *as it applies to the application*. A poke at a word in Shakespeare might mean anything, but from the point of view of the dictionary-lookup application, it almost certainly means to look up the word in the dictionary. Similarly, the meaning of a poke at a Shakespearean quotation in the *Oxford English Dictionary* is determined by whether the supporting application is a bibliographic lookup tool (in which case, the poke means to display full bibliographic information about the quotation) or the Shakespeare lookup tool (in which case, the poke requests the quote in the context of the full play).

The notion of link-resolving components changes the notion of hypertext and its links. Instead of defining a hypertext as a document with interconnecting links, we define it as a collection of link-resolving components. Links then become similar to database keys: not only can individual hypertext links be stored as explicit pointers, as in most other systems, but they can also be resolved "on the fly" through the execution of more complicated algorithms. In two prototypes, we have written link resolution mechanisms in GDL, one for the *OED* and one to browse the University's policies and minutes of meetings.

## 5. Commercialization

As a result of the *Shorter OED* project, Waterloo licensed the technology, including INR, GOEDEL, PAT, and LECTOR, to the Press for use in any of their projects. Thereafter, PAT and LECTOR were licensed to several academic and industrial research centres, to be used with the *OED* or other text data.

In 1989, Open Text Systems was incorporated to continue to develop and commercialize the technology. The first market was primarily academic and driven by the need to access the *OED*. To provide a better user interface, a window-based point-and-click front end was designed and implemented, making the technology easier to learn. In 1991, Open Text Corporation was formed and market emphasis shifted to larger commercial applications, including the support of text databases within financial, industrial, and government organizations. The software technology is being extended by Open Text and Waterloo's development group to encompass storage of text on CD-ROM, extended character sets to support non-English text, manipulation of text described by SGML, management of multi-gigabytes of text, improved support for distributed text databases, update management and version control.

## 6. Text/Relational Database Management Systems

Applying the database approach to text management proved valuable: database management systems have weak facilities for storing and manipulating text, and text-based systems ignore facilities for concurrent access and update and for providing user-oriented views of data. An ongoing major effort is to integrate powerful structured text management with relational database systems, research supported in part by NSERC and by Industry Canada and conducted in conjunction with the Canadian Strategic Software Consortium (including Open Text, Fulcrum Technologies, SoftQuad, InContext, Megalith, Public Service Systems, and Grafnetix).

The objective of the research is to design and implement a multidatabase system supporting text and relational data (T/RDBMS) that will better address the needs of these enterprises. To start, we have begun to design and implement an application program interface based on a data model that encompasses both the relational model and a grammar-defined text model. In this way we shall support both SQL, the industry standard for relational data, and SGML, the industry standard for structured text.

The T/RDBMS will be built as a federated database system with the actual data stored and managed by standard (relational and text) data management systems. Queries expressed in terms of the external data model will be parsed, and the relational and text components will be identified. Query strategies will then be analyzed so that an efficient access plan can be identified. This plan can subsequently be executed under the control of a database monitor, which distributes parts of the query task to component database systems as needed, and integrates the results before they are returned to the application. The heterogeneity of the component systems makes all these tools challenging.

## Acknowledgements

The work reported here resulted from the contributions of many individuals from within and outside the University. Particular acknowledgement is due to Gaston Gonnet, Howard Johnson, Rick Kazman, Tim Bray, Darrell Raymond, Betty Blake, Tim Snider, Heather Fawcett, Gayle Johannesen, Donna Lee Berg, Linda Jones, and the many undergraduate and graduate students who worked at the Centre during the last ten years.