DTL's DataSpot: Database Exploration Using Plain Language

Shaul Dar, Gadi Entin, Shai Geva, Eran Palmon
Data Technologies Ltd.
{dar,gadi,shai,eran@dtl.co.il}

Abstract

DTL's DataSpot is a database publishing tool that enables non-technical end users to explore a database using free-form plain language queries combined with hypertext navigation. DataSpot is based on a novel representation of data in the form of a schema-less semi-structured graph called a hyperbase. The DataSpot Publisher takes one or more possibly heterogeneous databases, predefined knowledge banks such as a thesaurus, and user-defined associations, and creates the hyperbase. The DataSpot Search Server performs searches and navigation against the hyperbase, returning answers to the user either in HTML pages or through an object API. The DataSpot product has been successfully deployed in diverse application areas including electronic catalogs, yellow pages, classified ads, help desks and finance.

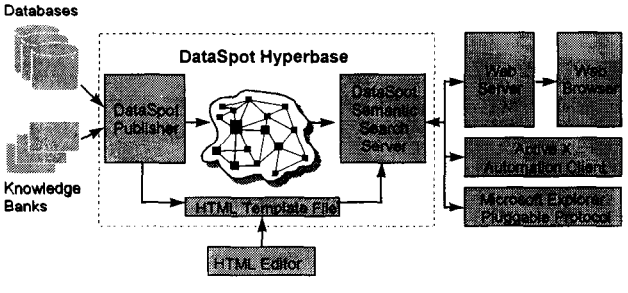

Figure 1: The DataSpot Architecture

Proceedings of the 24th VLDB Conference
New York, USA, 1998

1. Introduction

Database publishing on the Web is a major issue in today's world. The reasons are clear: the vast amounts of important business information that resides in corporate databases, the explosion of the Internet and in particular the familiarity of Web browsers as standard user interfaces. Currently, to publish databases on the Web, one must often design and implement forms, screens, and application flow (using HTML, DHTML, ASP, Java, SQL, etc.). In addition to being costly such applications present the user with a limited set of pre-designed queries. This method is often inappropriate for non-technical users seeking information in complex databases, a scenario commonly found in Internet and Intranet applications, e.g. an end user searching for a product in a large electronic catalog or a support representative looking for a solution in a corporate help-desk database. These examples demonstrate the need for a standardized, intuitive and friendly way for users to search online databases (see e.g. F96). It is this need that DTL's DataSpot seeks to address.

DataSpot introduces a new approach to database query and retrieval by providing end users with the capability of exploring databases using free-form queries and navigation. DataSpot lets lay users locate structured information much in the same way as they use a Web search engine, such as Alta Vista, to locate unstructured (textual) information. This capability is based on a novel, schema-less representation of data, called a hyperbase (also called a Web View). The DataSpot Publisher translates the source data into the hyperbase representation, which in turn may be queried efficiently by the DataSpot Search Server (see Figure 1). The hyperbase is a universal representation of data as a graph of associated elements, tailored to support free-form query algorithms. This representation lends itself naturally to a hypertext presentation and navigation via the mapping of associations to links.

The DataSpot representation and search technology is the foundation of the DataSpot system, available as a commercial product on Windows NT/95 platforms.

The paper is organized as follows. In Section 2 we describe the DataSpot data representation. In Section 0 we discuss how queries are processed. In Section 4 we describe the DataSpot system and outline application development. In Section 5 we review related work.

2. The DataSpot Representation

The data representation model used in DataSpot is called a *hyperbase*. A hyperbase is a graph structure comprised of nodes, edges and node labels. Nodes may be related via directed edges, which can be of two types. A simple edge is used to indicate inclusion and connects a node called the parent to a node called the child. The set of children of a node is ordered. The set of parents of a node is not ordered. A leaf node is a node with no children. A leaf node must have a label. An internal (non-leaf) node cannot have a label. The set of simple edges in a hyperbase cannot include a cycle. The second type of edge is an identification edge, used to indicate that one node, called the reference, uniquely identifies another node, called the subject. A node can have at most one reference.

Intuitively, nodes in the hyperbase represent data objects and edges represent associations between these data objects. For example in a hyperbase corresponding to relational data internal nodes may represent tables, records and fields, and leaf nodes may represent via their labels atomic data values, such as numbers or words. Edges represent the associations between data elements, such as the association between a record and its fields and between a field and its name and value. Identification edges represent a 1:1 association, such as the association between the key of a record and the record itself.

We define an equivalence relation $\equiv$ on nodes in a hyperbase H recursively as follows. Two nodes A and B in H are *equivalent* if one of the following conditions holds:

1. A and B are leaf nodes and have identical node labels.
2. A and B have an equivalent reference, i.e. A has a reference node A' and B has a reference node B', and $A' \equiv B'$.
3. A and B have no reference and have an equivalent list (ordered set) of children, i.e. let A's children be $A_1 \ldots A_n$ and let B's children be $B_1 \ldots B_m$, then $n = m$ and for $i=1 \ldots n$ $A_i \equiv B_i$.

A hyperbase H is *normalized* (is in normal form) if only leaf nodes have labels and if no two distinct nodes in H are equivalent.

We now define a *normalization* of a hyperbase H.

1. For each internal node A that has a label L do the following.
   i. Create a node A'.
   ii. Remove the label L from A and attach it to A'.
   iii. Create an identification edge from A' to A'
2. While there are equivalent nodes in H repeat the following steps:
   i. Select two equivalent nodes A and B in H. Replace A and B by a single node AB.
   ii. If A and B are leaf nodes with the same label L (condition 1 above) the new node AB has L as label.
   iii. The parent set of AB is the union of the parent sets of A and B.
   iv. If A and B are leaf nodes (condition 1 above) the child set of AB empty. If A and B have the same set of children (condition 3 above) the child set of AB is the same set. If A and B have the same reference (condition 2 above), the child set of AB is formed by concatenating the child sets of A and B.
   v. Node AB is a reference to any node referenced by A or B.

We define the *integration* of two hyperbases H_1 and H_2 (that may or may not be normalized) into a single normalized hyperbase H as follows:

1. Combine H_1 and H_2 into a single hyperbase H, where the set of nodes in H is the union of the nodes of H_1 and H_2 and the set of edges in H is the union of the edges of H_1 and H_2.
2. Normalize H.

These definitions underlie the process used by the DataSpot Publisher to create a hyperbase from a set of heterogeneous data sources, such as relational tables or text files, and associations, such as stemming and a thesaurus. The publisher first constructs a hyperbase for each data object, such as a relational record or a stem, using predefined translations. It then recursively integrates those hyperbases using a bottom up traversal in which equivalent nodes at each level are coalesced, resulting in a single normalized hyperbase. Note: from here on we use the term hyperbase to refer to a normalized hyperbase.

As an example, Figure 2a shows a relational database fragment consisting of two records, a Customers record and an Orders record, linked by the Customer Id field. In a relational database the link is "implied" by the foreign key in the Orders table having the same value as the key in the Customers record[1]. Figure 2b shows the hyperbase representation of the same database fragment (the figure has been simplified for sake of clarity). Simple edges are drawn using solid lines and identification edges are drawn using dashed lines. The text in internal nodes shows the "role" each node has in the original data source and the text in leaf nodes shows the label's type and value (in particular, the "Key" label type describes a value that serves as an internal id and is not retrievable by the user). The left Record node represents the Orders record, while the right Record node represents the Customers record. The key and foreign key fields have been coalesced and the coalesced field node serves as reference to the Customer record. Also shown is the integration of linguistic information. A morphological stem node is referenced by a stem label and has as children a set

[1] Foreign key specification is not part of the relational model per se. Some database products provide such functionality. The DataSpot Publisher gleans this information from the database if available, tries to guess it otherwise, and finally lets the user add or delete links.

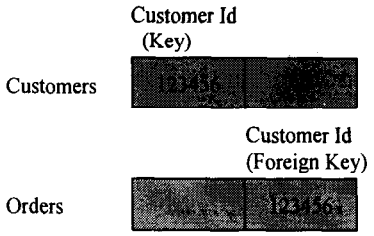

Figure 2a: A Sample Database

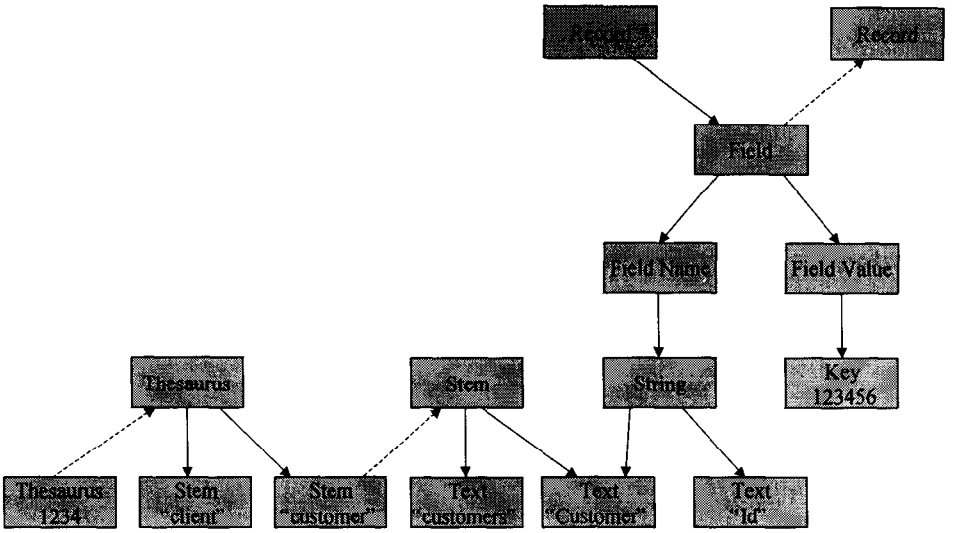

Figure 2b: A hyperbase

of words sharing that stem (including normalization to lower case). A thesaurus node is referenced by an internal thesaurus identifier and has as children a set of stems with similar meaning. The DataSpot system includes many other built-in associations, such as concepts, numbers and dates, and allows for user-defined associations (e.g. geographical information) to be added in similar fashion.

3. A DataSpot Query

A DataSpot query is an associative search over a hyperbase. The input to a query is a set of nodes, called the query sources. The result of a query is a list of answers, where each answer is a connected hyperbase containing the query sources (a partial answer may contain a subset of the query sources). An answer hyperbase is represented by a distinguished answer node selected heuristically according to the data model. The answers to a query are ordered (ranked) according to their score with, preferably, the "best" answer provided first. The score of an answer is based on the size of the answer hyperbase and the strength of the associations used to derive it. The query sources are typically specified as a set of words. However the user may also submit continuation queries whose input consists of words and of nodes or sets of nodes arrived at via previous queries or navigation.

Within the relational framework free-form queries can be understood as follows. The search process finds answer records that are related to the query words through relational and linguistic associations. The heuristics that select the distinguished answer node specify that the node be a record node from a table designated in the Publisher as an answer table (e.g. in an electronic catalog application this would likely be the table of items to be purchased). After receiving answers the user may view specific answer records in detail, navigate to related records, or the user may submit continuation queries from the current record or from a set of records.

As a simple example consider Microsoft's Northwind Traders database, a demo corporate database consisting of eight tables, namely employees, orders, order details, customers, products, suppliers, shippers and categories tables (for the schema and demo queries see www.dtl.co.il/dtl/sample/nwind/nwind.adb). The query "Nancy's seafood orders" returns (as an HTML page, shown in Figure 3) several records from the orders table. The first record, displayed in Figure 4, represents an order for a customer named QUICK-stop that was processed by the employee Ms. Nancy Davolio. One of the products in the order, Boston Crab Meat, is of category Seafood, and is also supplied by a supplier named New England Seafood Cannery). Note that finding this answer required five relational associations ("joins") that are shown under Reasoning at the bottom of Figure 4, as well as linguistic associations used e.g. to connect "Nancy's" with "Nancy (linguistic associations may also include related words and concepts associated via a thesaurus). Note also that the words "Nancy's" and "Seafood" relate to values in the database while the word "orders" is related to a metadata element (the order table), but the distinction is insignificant in the DataSpot representation.

Given the set of answers to the query the user may submit a refinement query. For example, typing the word "Mexico" would get the orders Nancy Davolio processed for customers located in Mexico. Alternatively, when looking at an answer record the user may submit a continuation query. For example, typing the word "Orders" when viewing the record for QUICK-stop would get all the orders processed for QUICK-stop by any employee. Further details on the DataSpot representation and on query semantics and

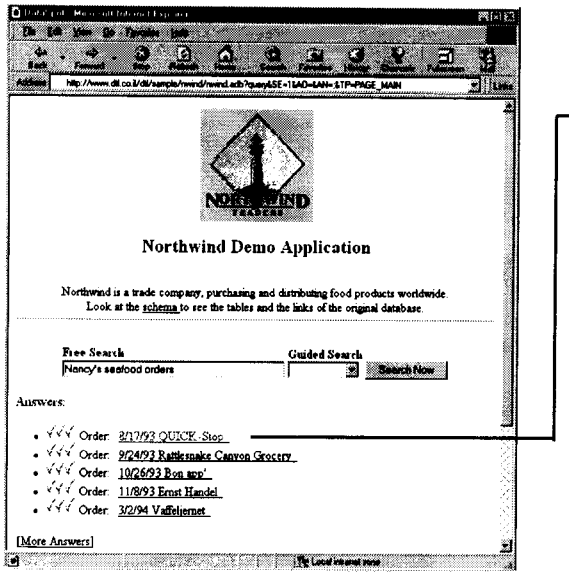

Figure 3: The Query "Nancy's Seafood Orders"

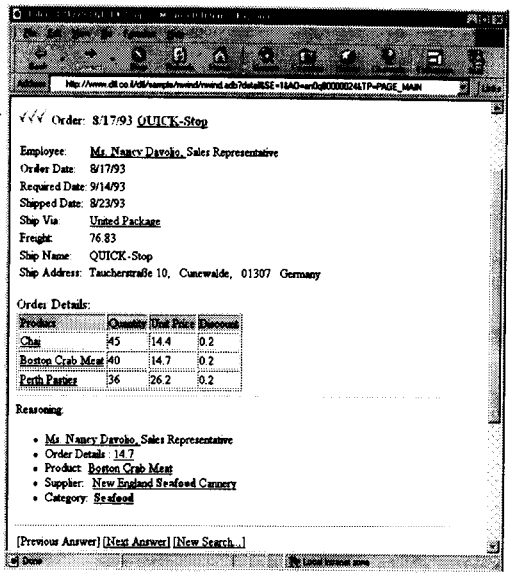

Figure 4: An Answer

computation can be found in [PAL95].

4. The DataSpot System

The DataSpot representation and search technology is the foundation of the DataSpot system, first released as a commercial product in April 1997. The DataSpot system implements the hyperbase structure on top of a proprietary scalable lightweight object system. We review briefly the functionality of the DataSpot system. The two major components of this system are the DataSpot Publisher and the DataSpot Search Server (see Figure 1).

Using the DataSpot Publisher the user specifies a set of data sources to be published, and the associations to be performed on these data sources. The Publisher then integrates these heterogeneous data sources into a single unified hyperbase. Key features of the DataSpot Publisher include:

- Interfaces all major database products via ODBC, DAO or native drivers, as well as flat files.
- Allows for scheduled batch hyperbase generation as well as concurrent incremental updates.
- Supports English and many European and (soon) far-eastern languages.

The DataSpot Search Server performs queries against the hyperbase. It provides a Web interface to the DataSpot technology, allowing users to submit queries and receive replies using a standard browser. DataSpot offers three modes of application setup (see Figure 1):

- Server applications using default HTML templates customizable via editors such as FrontPage.
- Standalone applications using Microsoft's Internet Explorer's pluggable protocol (dataspot://).
- ActiveX automation interface invoked programmatically from another application.

The first commercial release of DataSpot, DataSpot 1.1, has been available since April 1997 on Windows NT/95 platforms. It supports all major database products as well as flat files. DataSpot 2.1, released in June 1998, features several major enhancements: an online update capability to automatically keep the hyperbase synchronized with the source databases, an object interface to allow developers to integrate DataSpot searches into their applications, support for all major European languages, and major performance enhancements including a multi-threaded engine. DataSpot 3.0, currently in pre-release state, provides efficient parametric search, supports application development with Active Server Pages (ASP) technology and is integrated with Microsoft's Site Server Commerce Edition, a leading electronic commerce product. The DataSpot Search Server typically provides response times of few seconds on gigabyte hyperbases. As of this writing, several dozen DataSpot 1.1 and 2.1 systems have been successfully deployed in diverse Internet and Intranet application areas including electronic catalogs, yellow pages, classified ads, help desks and finance.

More information about DTL and DataSpot can be found at www.dataspot.com. In particular, see the DataSpot applications page for examples of applications built using DataSpot.

5. Related Work

Many products address the problem of providing a friendly way for users to retrieve information from databases. In general, these products fall into three categories: text search

engines, form-based interfaces and natural language interfaces.

Text-search engines (E.g. AltaVista, Yahoo!, Excite, InfoSeek, Lycos, Verity) . These products provide retrieval of structure-free, text-based information. The criteria for the search are usually that the requested words appear in the same proximity in the document. Text-search engines do not typically capture database semantics (e.g. foreign keys) and do not support navigation between related data elements.

Form-based interfaces (e.g. Saphire/web, NetDynamics, Cold Fusion, dbWeb). These products offer a user interface based on structured forms. This approach requires a significant programming effort on the part of the data provider while limiting the user to requesting data via a particular set of queries. In addition, the input to a query must be specified exactly, e.g. linguistic associations would be difficult to incorporate.

Natural language Interfaces (e.g. English Wizard, Microsoft English Query). These products offer a natural-language interface to the user and then translate the query into SQL statements. This approach looks similar to ours in that it supports free-form queries and it uses an external index on the published data values (e.g. to guide the translation of the user's query), but it passes query evaluation to the database query processor. The DataSpot hyperbase representation provides important advantages, such as the ability to integrate heterogeneous databases, extensibility (e.g. non relational data, additional languages and associations), better performance (e.g. joins are "pre-computed" and search algorithm is specialized), and ability to justify answers (as opposed to just showing the generated SQL).

Representation of unstructured or loosely structured data has received much attention in the database research community in recent years (see e.g. BDHS96, GSVG98, MAGQW97, PGMW95). For lack of space we mention only a few salient points regarding the relationship of the DataSpot data model (the hyperbase) to these proposals. First, the emphasis of the DataSpot representation is to enable free-form queries, while most of the above mentioned research work is focused on formal query languages for semistructured data (with the exception of [GSVG98], which also emphasizes keyword search). Second, the DataSpot model is different than the above proposals, and it addresses some important theoretical and practical questions such as the representation of cyclic data and the integration of information from heterogeneous data sources, including relational and non-relational databases, a thesaurus and user-defined associations. Third, a lot of attention has been paid in DataSpot to the efficient implementation of the graph, to the acceleration of the bulk load process that creates it and to the tuning of the query process that traverses it, including support for multi-threaded queries concurrent with updates.

Other work in the database research community addresses ways to organize and query data on the Web (see e.g. KS95, LRO96, MAGQW97, MM97). These studies are complementary to our work.

Acknowledgments

Catriel Beeri helped stimulate the early theoretical work underlying the hyperbase structure. We are also grateful to Divesh Srivastava and S. Sudarshan for providing us with valuable feedback on this paper.

References

References to commercial products are omitted. Information can be found in the respective Web sites.

[BDHS96] P. Buneman, S. Davidson, G. Hillebrand and D. Suciu. A Query Language and Optimization Techniques for Unstructured Data. SIGMOD 1996.

[F96] Searching Text and Tables. Maurice Frank, Editor. Internet Systems (DBMS Magazine supplement), October 1996.

[FFKLS97] System Demonstration - Strudel: A Web-site Management System. By Mary Fernandez, Daniela Florescu, Jaewoo Kang, Alon Levy, Dan Suciu. SIGMOD 1997.

[GSVG98] R. Goldman, N. Shivakumar, S. Venktasubramanian, H. Garcia-Molina, Proximity Search in Databases, VLDB 1998.

[KS95] D. Konopnicki and O. Shmueli. W3QS: A Query System for the World Wide Web. VLDB 1995.

[LRO96] Querying Heterogeneous Information Sources Using Source Descriptions by Alon Y. Levy, Anand Rajaraman and Joann J. Ordille.VLDB 1996.

[MAGQW97] J. McHugh, S. Abiteboul, R. Goldman, D. Quass, and J. Widom. Lore: A Database Management System for Semistructured Data. SIGMOD Record, 26(3): pgs. 54-66, September 1997.

[MM97] A. Mendelzon, T. Milo. Formal Models of the Web. PODS 1997.

[PAL95] E. Palmon, Associative Search Method for Heterogeneous Databases with an Integration Mechanism Configured to Combine Schema-Free Data Models such as a hyperbase, United States Patent Number 5,740,421, April 14 1998 (filed April 3, 1995).

[PGMW95] Y. Papakonstantinou, H. Garcia-Molina and J. Widom. Object Exchange across Heterogeneous Information Sources. ICDE 1995.