IAEA FAO UNESCO UNIDO

INTERNATIONAL SYMPOSIUM ON THE FUTURE OF SCIENTIFIC, TECHNOLOGICAL AND INDUSTRIAL INFORMATION SERVICES

Leningrad, USSR, 28–31 May 1990

IAEA-SM-317/ 42

HYPERDATABASE: A SCHEMA FOR BROWSING MULTIPLE DATABASES

M.A. SHEPHERD

Computer Science Division, Dalhousie University, Halifax, Canada

C.R. WATTERS

Computer Science Department, University of Waterloo, Waterloo, Canada

HYPERDATABASE: A SCHEMA FOR BROWSING MULTIPLE DATABASES[1]

0. ABSTRACT

In order to insure effective information retrieval, a user may need to search multiple databases on multiple systems. Although front end systems have been developed to assist the user in accessing different systems, they access one retrieval system at a time and the search has to be repeated for each required database on each retrieval system. More importantly, the user interacts with the results as independent sessions.

This paper models multiple bibliographic databases distributed over one or more retrieval systems as a hyperdatabase, i.e., a single virtual database. The hyperdatabase is viewed as a hypergraph in which each node represents a bibliographic item and the links among nodes represent relations among the items. In response to a query, bibliographic items are extracted from the hyperdatabase and linked together to form a transient hypergraph. This hypergraph is transient in the sense that it is "created" in response to a query and only "exists" for the duration of the query session. A hypertext interface permits the user to browse the transient hypergraph in a nonlinear manner.

The technology to implement a system based on this model is available now, consisting of powerful workstations, distributed processing, high-speed communications, and CD-ROMs. As the technology advances and costs decrease such systems should be generally available.

---

[1]This research was supported in part by the Natural Sciences and Engineering Research Council of Canada, operating grant number A9163.

1. INTRODUCTION

Over the past decade, the number of publicly available online services has increased dramatically from 59 services offering a total of 400 different databases in 1979 to 600 services offering a total of 4062 different databases in 1989 [1]. While this indicates a tenfold increase in the availability of information, it also indicates an increase in the complexity of conducting an online search.

In order to insure effective information retrieval, a user may need to search more than one database. Although some current systems offer multi database cross indices to direct the user to the appropriate database(s), the user has to search each individual database and amalgamate the results. Also, it may be that not all of the required databases are available on the same retrieval system, or that a given database may be mounted on different systems offering different search features. This necessitates searching multiple databases on multiple systems. These systems tend to have different communication and logon protocols, different command languages, and different search features making it difficult for a naive user (or even an experienced user) to conduct a comprehensive search. Although front end systems have been developed to assist the user in accessing different systems, they access one retrieval system at a time and the search has to be repeated for each required database on each retrieval system (see [2] for a bibliography on front ends). More importantly, the user interacts with the results as independent sessions.

In addition to an increase in the complexity of conducting an online search, the advent of powerful workstations and bit-mapped displays have added a new dimension to data access, i.e., browsing. Traditionally, access to data has consisted of the user presenting a query to a system in a query language, possibly natural language, and the system processing the query against the database and returning a set of items satisfying the query as presented. This set

may be empty or it may contain a great many items. If the user is not satisfied with the results of the query, the user typically rephrases or modifies the original query and presents this modified query to the system.

In addition to querying a database, a user may now browse a database. Browsing a database involves viewing a database item and linking to related items on the basis of some attribute or attribute value. This is the basic data access methodology of hypertext systems [3-6]. In hypertext, windows on the screen are associated with objects or nodes in a database and the objects are linked in a network or graph. The user browses the database by traversing the links between the nodes. The notion of hypertext browsing in a database is not new [7], but was not a feasible search technique until the appropriate technology became readily available.

The hyperdatabase approach, described in detail below, views all of the requested databases on all of the requested systems as one virtual database, i.e., a hyperdatabase, against which a user query is processed. This approach reduces the complexity of multiple database/multiple system searching and, at the same time, integrates database querying and hypertext browsing into a single system. In order to keep the examples simple and the discussion straightforward, the model presented deals only with bibliographic databases.

A hyperdatabase is viewed as a single virtual hypergraph in which each node represents a bibliographic item and the links among nodes represent relations among the items. In response to a query, bibliographic items are extracted from the various databases and linked together to form a transient hypergraph [8]. This hypergraph is transient in the sense that it is "created" in response to a query and only "exists" for the duration of the query session. A hypertext [3-6] interface permits the user to browse the transient hypergraph in a nonlinear manner.

Section 2, below, presents an overview of a system architecture for a hyperdatabase system for bibliographic databases. Section 3 presents the hypergraph-based model while Section 4 describes querying and browsing within the model.

2. SYSTEM ARCHITECTURE

The hyperdatabase consists of a collection of distinct databases and search services with some overlap of items between databases and/or services. These databases may be physically distant and accessed via telecommunication networks or they may be available locally on CD-ROM. Whether physically distant or on local CD-ROM, the databases available from a search service are accessed through the same software and command structure.

A local database is constructed from items downloaded from the hyperdatabase in response to user queries. This local database can be built either dynamically from scratch at each search session or can consist of the results of previous search sessions and be updated dynamically during the current session. The items in the local database are represented by nodes in the transient hypergraph and the hyperedges are instantiated dynamically from the downloaded items.

Assuming that no local database exists at the beginning of a search session, the user initiates the session by presenting a query to the hyperdatabase. Those items satisfying the query are downloaded from the hyperdatabase into the local database where they can be browsed in a hypertext manner. If the user information need is not satisfied by the items in the local database, the hyperdatabase may be accessed further and more retrieved items downloaded and incorporated into the local database.

Figure 1 illustrates the layered architecture of such a hyperdatabase system. The hypertext layer supports the user interface, communicates queries for the hyperdatabase to a front end system, and supports the querying and browsing of the transient hypergraph.

[Figure 1 goes here]

As each item is downloaded from the hyperdatabase, the local database manager instantiates a representative node containing attribute values extracted from the item and assigns a node_id. Links between nodes are instantiated initially as per the query. These nodes and links form an initial transient hypergraph which can then be queried and browsed in a hypertext manner by the user.

Before instantiating a node, the local database manager checks to see if the item is already in the local database. If the item is already in the local database and is from the same search service and database, then it is a duplicate item and no action is taken other than to instantiated appropriate links as per the query. If the item is already in the local database but is not from the same search service and/or database it is entered into the local database with the same node_id as the item already in the local database. These items can then be browsed as a composite node and also viewed individually.

The front end translates the queries for the hyperdatabase into the syntax of the command languages of the various target systems and databases, accepts the responses from the various target systems, and passes the downloaded items to the local database manager. As the hyperdatabase is distributed, physical access to the various retrieval systems can be executed in parallel. The availability of databases (with appropriate software) on CD-ROM would limit the data communications costs and limit the need for downloading.

3. THE DATA ACCESS MODEL

The hypertext layer supports the transient hypergraph-based model of data access. A data access model must provide both browsing as per hypertext as well as retrieval of sets as per DBMS and traditional information retrieval. This section describes the data access model by first presenting a brief overview of hypergraphs followed by the concept of transient hypergraphs.

3.1. Hypergraphs

The following discussion of the hypergraph [9] as a model for data access is described in more detail in [8]. For the purposes of this discussion, it is assumed that the hyperdatabase is a bibliographic database that may have links between the items.

Let a database, X, be a finite set of such items $\{x_1,x_2,..,x_n\}$ with which we can associate a family of edges $E = \{E_1,E_2,..,E_m\}$ where each edge, E_i, describes a subset of X such that all of the items in that subset are grouped together on some basis.

A hypergraph representation of the database, X, can be defined where the following properties hold:

1) The union of all of the subsets defined by the edges must contain all of the items in the database:

$$\cup E_i = X \qquad (E_i \in E)$$

2) All of the edges are non-null:

$$E_i \neq \varnothing$$

Where these two properties hold, we can say that *E*, the family of subsets of X defined by the set of edges, is a hypergraph, H, on X, defined by the couple (X,*E*).

Let us consider the following database of bibliographic items to illustrate the model:

doc A by author A_1 with descriptors D_1, D_4, D_6

doc B by author A_1 with descriptors D_2, D_5

doc C by author A_2 with descriptors D_1, D_6

doc D by author A_3 with descriptors D_1, D_4

doc E by author A_2 with descriptors D_2, D_5

doc F by author A_1 with descriptors D_3, D_6

Within the framework of a hypergraph model the following would be true for the sample database:

$X = \{A,C,E,B,D,F\}$

$E_1 = \{x_i |$ descriptor $= D_1, x_i \in X\}$

$E_2 = \{x_i |$ descriptor $= D_2, x_i \in X\}$

$E_3 = \{x_i |$ descriptor $= D_3, x_i \in X\}$

$E_4 = \{x_i |$ author $= A_1, x_i \in X\}$

$E_5 = \{x_i |$ author $= A_2, x_i \in X\}$

$E_6 = \{x_i |$ author $= A_3, x_i \in X\}$

$E = \{E_1, E_2, E_3, E_4, E_5, E_6\}$

The hypergraph, $H = (X,E)$, that represents the sample database, X, is shown in the Fig. 2. In the graphical representation of the hypergraph, an edge containing a single item (such as E_6) is shown as a loop starting and ending at that item, an edge with only two items (such as E_5) is shown as a line, and an edge with more than two items (such as E_4) is shown as a loop enclosing the items.

[Figure 2 goes here]

Each edge in this hypergraph represents a subset of the items in the database, that satisfy some condition. The condition may be a specific value for some attribute (such as author = A_i or descriptor = D_j) or it may be an entity type (such as memo or employee in a heterogeneous database). As the nodes in the hypergraph represent individual items of the database, the terms "node" and "item" can be considered to be equivalent and may be used interchangeably.

3.2. Transient hypergraph model

The traditional approach to querying a textual database has been that a query is put to the database and a set of records is returned where each record in the set satisfies the query conditions. For example, in the sample data set, the query (author = A_1) returns the set of records {A,C,D}. The user can then look at the items in this answer set but not outside this set. In other words, the database has been partitioned for the user with respect to some attribute with no links outside that set. The user either finds text that satisfies the information need (including the null case) or rethinks the query and starts again. A *raison d'etre* for hypertext access is to allow the user to follow paths amongst items both within the subset and leading from the subset rather than to restrict the user to viewing a completely isolated subset of the database. In this section the hypergraph model is shown to be a suitable vehicle for describing both traditional data access and hypertext data access methods for the retrieval of information.

3.2.1. Hyperedges and Transient Hypergraphs

If E were the complete set of all possible hyperedges in H, then E would be the power set $P(X)$ without the null set $\{\emptyset\}$. This would result in a hypergraph consisting of 2^n-1 hyperedges, where $|X| = n$. When n is very large, as in bibliographic databases containing hundreds of thousands of items, it would not be feasible to attempt to create or maintain a complete hypergraph. The transient model allows access across the complete hypergraph by generating the hyperedges of a transient hypergraph dynamically in response to user queries. At the start of any search session the only hyperedge that is assumed to exist is $E_X = \{X\}$. Edges, called transient edges, within this identity edge are generated during the session and may be discarded at the end of the session.

Edges amongst the items of this entity type are neither instantiated nor made explicit except in response to specific queries. The response to a given initial query can be seen as the generation of a transient hypergraph, H_T, defined as

$$H_T = (X_T, E_T)$$

where X_T is initially the set of items satisfying the query, and E_T is initially only the edge, E_Q, containing the items in X_T.

In order to generate the transient edges during the search session, each node in the hypergraph must have attribute-value pairs associated with the database item, where each node, $x \in X$, represents an item in the database. For a database of instances of a given item, such as bibliographic records, we can define a finite set of attributes, $T = \{A_1, A_2, \ldots, A_m\}$, for which a set of values (including the null value) from the database can be associated with each instance of the item. The database can then be represented as a subset of the Cartesian product of the values of the attributes. Each item is defined by an m-tuple that is a member

of that Cartesian product, where m is the number of attributes. Each such item is represented in the hypergraph as a node.

The transient hypergraph is created by the dynamic generation of edges. In order to generate edges dynamically, we need to be able to manipulate edges to produce new edges. The edges of a hypergraph can be manipulated by a set of operators to form new edges from existing edges. By allowing the possibility of generating a null edge, $E_{\varnothing} = \{\varnothing\}$, where $E_{\varnothing} \notin E$, we can view the operations as closed on E.

These edge operators are INTERSECTION, UNION, DIFFERENCE, and GETFROM (γ) and are described in detail in [8]. A system based on this data access model may be implemented in many different ways. If, as suggested in this paper and in [8,10,11], a relational or universal DBMS [12] is used as an implementation platform, these operators may be implemented using the DBMS operators of INTERSECTION, UNION, DIFFERENCE, and SELECT (σ).

4. DATA ACCESS IN THE HYPERDATABASE MODEL

A distinction has to be made between the hyperdatabase and the local database. The hyperdatabase consists of a collection of distinct databases and search services with some overlap of items between databases and/or services. The hyperdatabase is represented by the hypergraph, H_X, which consists of all the items in all the databases and the single hyperedge, E_X.

The local database consists of the set of items retrieved from the hyperdatabase in response to a user query. The local database is represented by the transient hypergraph, H_T, consisting of the items in the local database and the family of edges, E_T, which represent the links among those items.

The hyperdatabase is queried to establish a local database that the user can browse and/or query further. In order to facilitate the "creation" and the subsequent hypertext browsing of the transient hypergraph, attribute values are instantiated from values extracted from each downloaded item. If implemented in a universal relation database management system, an appropriate universal relation might be as follows:

hyper(node_id,cit_data,system,database,au,key,desc,class)

Once the attribute values have been instantiated a transient hypergraph exists in which the nodes represent the hyperedge members. Links among these nodes, both explicit and implicit, are instantiated dynamically during browsing or further querying [8,10,11]. Explicit links of types *system*, *database*, and *classification* are instantiated and appear as buttons on the screen. These links are traversed by clicking on the appropriate button. Implicit links of types *author*, *descriptor*, and *keyword* (from the title and abstract) are instantiated and traversed by clicking on the appropriate text string on the screen when viewing a node. While the transient hypergraph can be searched as a closed set of items, it can be expanded by adding more items from the hyperdatabase. The local database can be saved, of course, for future searching or it can be discarded.

A bibliographic item in the hyperdatabase may be present in more than one database and/or system. These multiple occurrences are identified through comparison of citation data (*cit_data*) attribute values which include such information as journal, year, volume, issue, pages, etc. Each such occurrence of an item is assigned the same node_id as the other occurrences of that item. The following example is given with hypothetical data.

hyper

node_id	cit_data	system	db	au	key	desc	class
1	doc_X	S_1	DB_1	$\{A_2\}$	$\{k_i\}$	$\{D_1,D_4,D_6\}$	C_1
2	doc_Y_1	S_1	DB_1	$\{A_1\}$	$\{k_i\}$	$\{D_1,D_5\}$	C_2
2	doc_Y_2	S_2	DB_3	$\{A_1\}$	$\{k_i\}$	$\{D_1,D_6\}$	C_3
3	doc_Z	S1	DB_2	$\{A_3\}$	$\{k_i\}$	$\{D_1,D_7\}$	C_4

The values for *author*, *key*, and *desc* are sets [13]. In this example, two nodes assigned node_id = 2 represent different occurrences of the same item and can be treated as a composite node for browsing. The user may request to view the composite node with node_id = 2, and the system will indicate that there are multiple representations of the same item. The user may choose to browse all such representations and to continue browsing from any of them.

Let us consider three scenarios, from the user's perspective, for the satisfaction of an information need. In the first scenario, *querying*, the user presents a query to the hyperdatabase and the answer set itself (i.e., the local database) satisfies (or doesn't satisfy) the user's information need and the session is terminated. In the second scenario, *browsing*, the user chooses to examine individual items or sets of items in the local database in order to satify the information need. In the third scenario, *expanded browsing*, the user needs to look outside the current local database in the hyperdatabase to find the required information.

The following describe the three scenarios, assuming the above example database is the hyperdatabase and the universal relation, hyper, describes the entity represented by the hypertext nodes.

4.1. Querying

In the first scenario, the user presents a query to the hyperdatabase. Let us consider the processing of a query within the framework of the transient hypergraph model. At the beginning of a search session, the hyperdatabase can be considered to consist of at least one edge, $E_X = X$, although the edge is not actually instantiated. A query against the hyperdatabase returns a set of items which are downloaded to the local database. A transient hypergraph is constructed from this set of nodes, generating a new edge, E_1.

Let a query consist of attribute-value pairs connected by logical and/or arithmetic operators. A simple query is one based on a single attribute-value pair, such as (author= A_2). A complex query is one with more than one pair connected by operators, such as (author = A_1 AND descriptor = D_6).

As an example, let us begin a search session with the simple query, (author = A_1). The sample hyperdatabase of six documents is searched for all documents by author A_1. A transient hypergraph consisting of the single hyperedge, E_1, is created. This query can be represented as follows:

$$E_1 = \gamma_{au=A_1}(E_X)$$

The GETFROM (γ) from E_X indicates that the query must be processed against the hyperdatabase. The γ operation may be implemented using the SELECT (σ) operator from relational and universal relation database management systems. The hyperedge, E_1, resulting from this query can be represented as the relation E_1 as follows:

E_1

node_id	cit_data	system	db	au	key	desc	class
1	A	S_1	DB_1	$\{A_1\}$	$\{k_i\}$	$\{D_1,D_4,D_6\}$	C_1
2	B	S_1	DB_1	$\{A_1\}$	$\{k_i\}$	$\{D_2,D_5\}$	C_2
3	F	S_2	DB_3	$\{A_1\}$	$\{k_i\}$	$\{D_3,D_6\}$	C_3

The relation E_1 represents the hyperedge, E_1, of Fig. 3 which was created in response to the initial query. Each tuple represents a node in the hyperedge. The family of transient hyperedges would now be $E_T = \{E_1\}$.

[Figure 3 goes here]

4.2. Browsing within the local database

In the second scenario, the user wishes to browse the exisiting transient hypergraph. In our example, the transient hypergraph consists of the single edge, E_1. Browsing within E_1 entails dynamically generating new edges within the scope of H_T. These edges group or select items from E_1 and can be generated in two different ways. The first way of generating a new edge is by the user selecting and viewing items in E_1. This is accomplished by the user clicking on a node and the system displaying the corresponding item. This should be familiar to hypertext users.

The second way of generating a new edge from the elements of H_T is to create an edge on the basis of some attribute-value pair related to the item that the user is currently viewing. Such an attribute-value pair might be "same_author" or "same_ subject", etc. In this case, edges are again generated within the transient hypergraph. As we can add edges of semantic

content, we can begin to provide hypertext browsing capabilities amongst the items and sets of items in the transient hypergraph where the new edge is defined relative to other edges in H_T.

Notice that any attribute-value pair (or combination of such pairs) can be used to define an edge of two or more items within H_T. For example, if the user is currently viewing item A in the edge E_1, then an edge defined on the condition of (descriptor = D_6) can be instantiated that would include the items {A,F} and be included in H_T as follows:

$$E_2 = \gamma_{\text{descr}=D_6}(E_1)$$

As the GETFROM is from E_1 as opposed to E_X, the browse is processed against the local database. This creates the hyperedge, E_2, which is represented by the relation E_2 as follows:

E_2

node_id	cit_data	system	db	au	key	desc	class
1	A	S_1	DB_1	$\{A_1\}$	$\{k_1\}$	$\{D_1,D_4,D_6\}$	C_1
3	F	S_2	DB_3	$\{A_1\}$	$\{k_1\}$	$\{D_3,D_6\}$	C_3

The relation E_2 represents the hyperedge E_2 which was created in response to browsing within the transient hypergraph. The family of hyperedges would now be $E_T = \{E_1,E_2\}$, as shown in Fig. 4.

[Figure 4 goes here]

4.3. Expanded browsing

In the third access scenario, the user does not find the information needed within the returned set and would traditionally terminate the query and start again. In keeping with hypertext ideals, we would like the user to be able to link from the items in the set to related items outside of the returned set by specifying the instantiation of further attribute-value pairs. The user could specify whether the browse request is directed to the local database or to the hyperdatabase by toggling a button on the screen.

To illustrate expanded browsing, assume that the user was currently at the node representing item A. The user could specify the instantiation of a new edge of items with the same value as one or more of the attributes of item A, e.g., (descriptor = D_1). The user would specify the request by clicking on the appropriate text string on viewing the node. Such a request would expand H_T with the edge containing {A,C,D} where C and D were not previously included in X_T. The user would then be able to browse the expanded hypergraph. Such a request could be represented as follows:

$$E_3 = \gamma_{desc=D_1}(E_X)$$

As the GETFROM is from E_X, the browse is processed against the hyperdatabase. The resulting hyperedge, E_3, can be represented by the relation, E_3, as follows:

E_3

node_id	cit_data	system	db	au	key	desc	class
1	A	S_1	DB_1	$\{A_1\}$	$\{k_1\}$	$\{D_1,D_4,D_6\}$	C_1
4	C	S_4	DB_8	$\{A_2\}$	$\{k_1\}$	$\{D_1,D_6\}$	C_7
5	D	S_2	DB_5	$\{A_3\}$	$\{k_1\}$	$\{D_1,D_4\}$	C_1

The relation E_3 represents the hyperedge E_3 which was created in response to browsing within the transient hypergraph. The family of hyperedges would now be $E_T = \{E_1, E_3\}$, as shown in Fig. 5.

[Figure 5 goes here]

5. SUMMARY

The hyperdatabase model permits the user to browse and query multiple databases distributed across multiple systems as though they were a single database on a single system. In addition to browsing the database, the user may query the universal relation directly to determine such things as overlap among databases and/or systems, etc. The technology to implement a system based on this model is available now, consisting of powerful workstations, distributed processing, high-speed communications, and CD-ROMs. As the technology advances and costs decrease such systems should be generally available.

6. REFERENCES

[1] Directory of Online Databases, Vol.10, No. 1, Cuadra/Elsevier, New York (1989) v.

[2] HAWKINS, D.T., LEVY, L.R., Front end software for online database searching. Part 3: Product selection chart and bibliography, Online 10 3 (1986), 49-58.

[3] CONKLIN, J., Hypertext: An introduction and survey, IEEE Computer 20 9 (1987) 17-41.

[4] FIDERIO, J., A grand vision, Byte 13 10 (1988) 237-244.

[5] NELSON, T.H., Managing immense storage, Byte 13 1 (1988) 225-238.

[6] SCHNASE, J.L., LEGGETT, J., KACMAR, C., BOYLE, C., A Comparison of Hypertext Systems, Rep. TAMU 88-017. Hypertext Research Lab, Texas A&M University, (1988).

[7] BUSH, V., As we may think, Atlantic Monthly 176 (July 1945), 101-108.

[8] WATTERS, C.R., SHEPHERD, M.A., A transient hypegraph-based model for data access, Rep. CS-90-01, Computer Science Department, University of Waterloo, Waterloo, Canada, (1990).

[9] BERGE, C., *Graphs and Hypergraphs*, North-Holland Publishing Company, London, (1973).

[10] SHEPHERD, M.A., WATTERS, C.R., "Hypertext: User-driven interfaces", (Mid-Year Conference of the American Society for Information Science, San Diego, May 21 - May 24, 1989), In Press.

[11] SHEPHERD, M.A., WATTERS, C.R., CAI, Y., Transient hypergraphs for citation networks, Information Processing and Management. In Press.

[12] ULLMAN, J.D., *Principles of Database Systems*, Computer Science Press, Potomac, Md., (1983).

[13] ROTH, M.A., KORTH, H.F., SILBERSCHATZ, A., Null values in relational databases, Acta Informatica 26 (1989) 615-642.

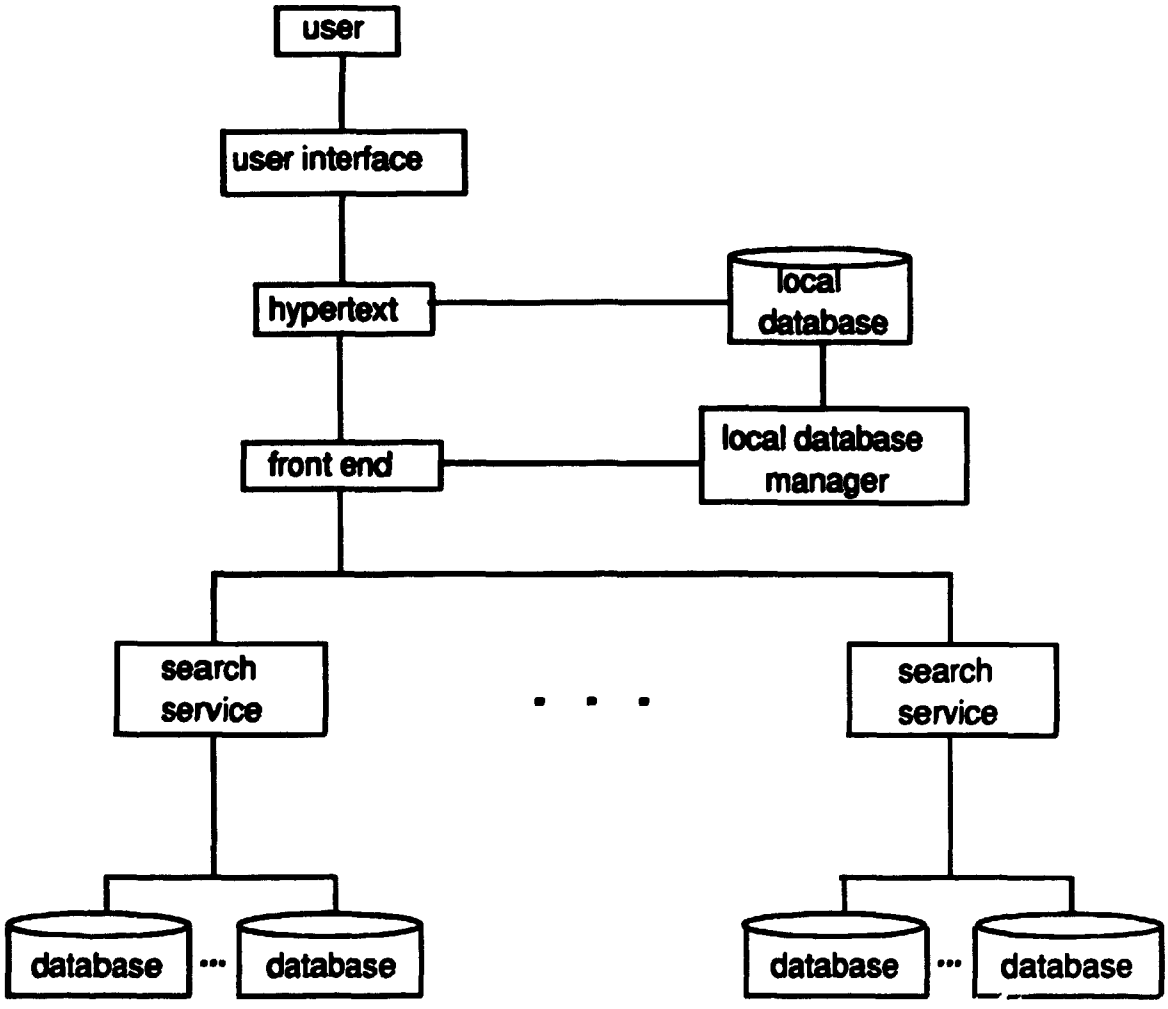
Fig. 1. Layered architecture of the hyperdatabase system.

Fig. 2. Hypergraph of sample database.

Fig. 3. Hyperedge for query, (au = A_1).

Fig. 4. Browsing in the transient hypergraph.

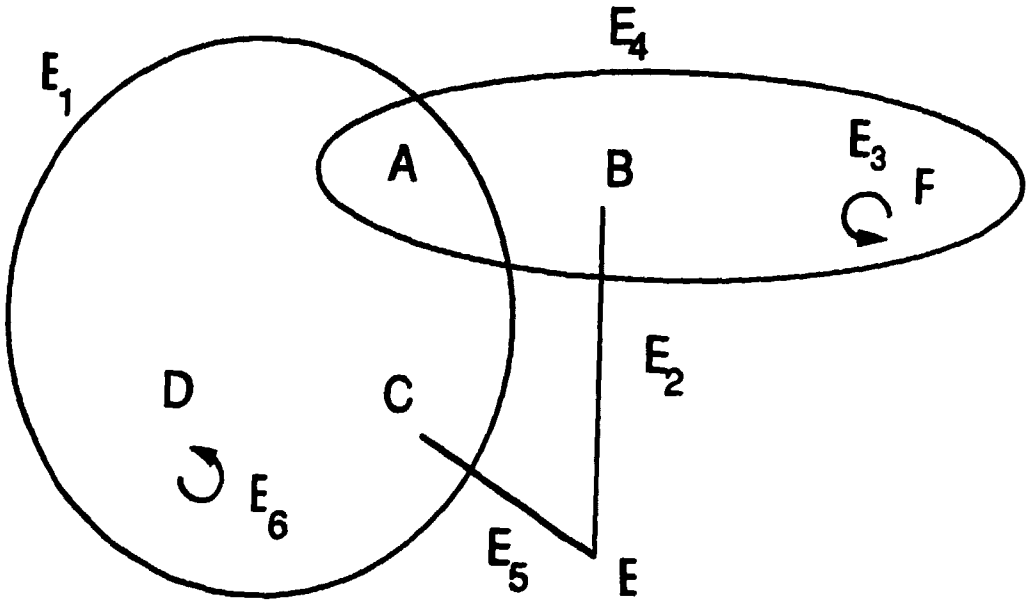
Fig. 5. Browsing in the hyperdatabase.


user
user interface
hypertext
local database
front end
local database manager
search service
search service
database
database
database
database

M.A. Shepherd, C.R. Watters

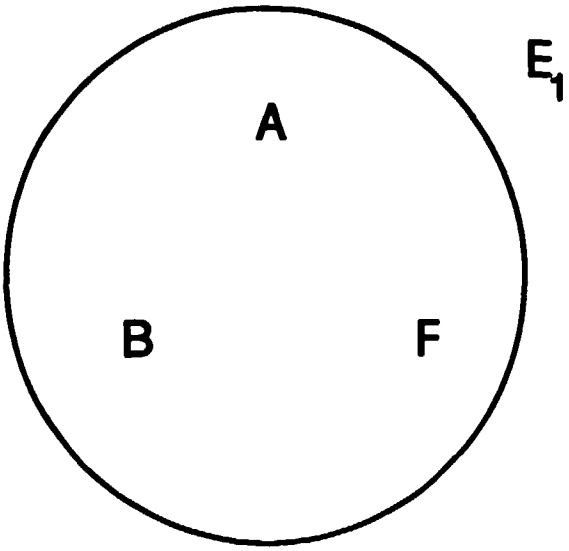
Paper Number: IAEA-SM-317/42

Figure 1


E1
E4
A
B
E3
F
E2
D
C
E6
E5
E

M.A. Shepherd, C.R. Watters
Paper Number: IAEA-SM-317/42

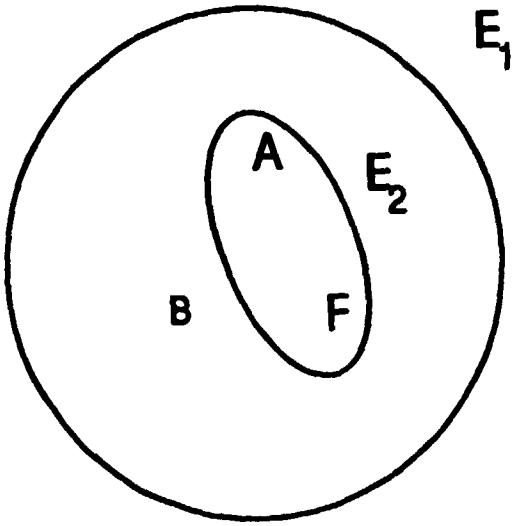
Figure 2


E1
A
B
F

M.A. Shepherd, C.R. Watters

Paper Number: IAEA-SM-317/42

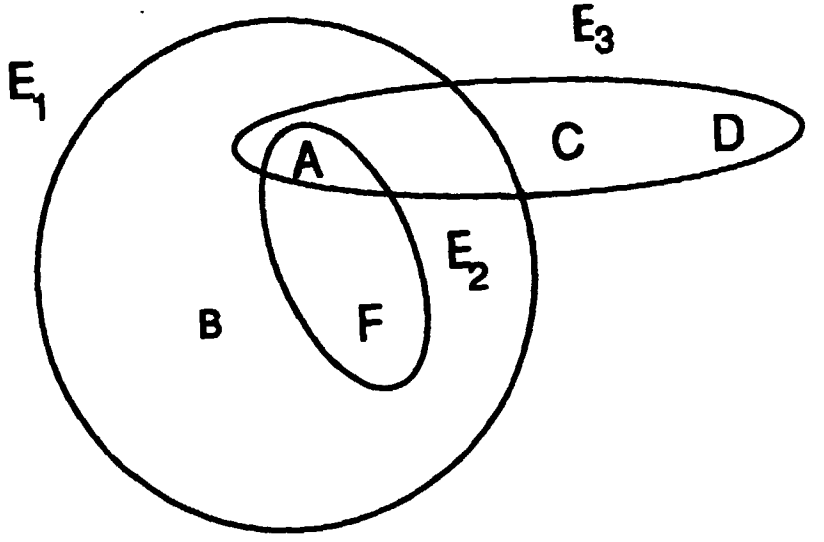
Figure 3.


E1
A
E2
B
F

M.A. Shepherd, C.R. Watters

Paper Number: IAEA-SM-317/42

Figure 4


E3
E1
A
C
D
E2
B
F

M.A. Shepherd, C.R. Watters

Paper Number: IAEA-SM-317/42

Figure 5