Browsing Large Digital Library Collections Using Classification Hierarchies

S. Geffner D. Agrawal A. El Abbadi T. Smith

Department of Computer Science
University of California
Santa Barbara, California 93106
(805) 893-4321

{sgeffner, agrawal, amr, smithtr}@cs.ucsb.edu

ABSTRACT

Summarization of intermediary query result sets plays an important role when users browse through digital library collections. Summarization enables users to quickly digest the results of their queries, and provides users with important information they can use to narrow their search interactively. Techniques from the field of data analysis may be applied to the problem of generating summaries of query results efficiently. Such techniques should permit the incorporation of classification hierarchies in order to provide powerful browsing environments for digital library users.

Keywords

Aggregation, summarization, classification, digital libraries, browsing, searching.

1. INTRODUCTION

Data browsing capabilities are essential in digital libraries. Many users often have fuzzy or incomplete queries, which they wish to refine iteratively once they see what a library actually contains. In order to refine their search, these users need feedback about the results of intermediate searches; this feedback should summarize the results of the user's query in an easily digestible form. We are studying issues in data browsing in the context of the Alexandria Digital Library (ADL) [16], which contains large collections of maps, air photos, and other geospatially-referenced information. Many of ADL's users are so-called "naive" users, i.e., people who are neither librarians, geographers nor database experts. These users commonly wish to find a map of a certain geographic area at a certain scale, and perhaps from a certain time period, e.g. "find me a map of the City of Santa Barbara from the 1950's".

Users who query the collection face several obstacles. First, they may not be aware of exactly what is contained in the collection. Without guidance, they may issue many queries that return no results. Second, and conversely, they may issue queries that return very large result sets. This is problematic because the results returned by a query become increasingly unmanageable for a user as the size of the result set grows. For example, suppose a user makes a request for satellite photographs in a map library. A query result set that contains a list of 50,000 photos would probably not be very useful in narrowing the search further; the result set is too large to be digested easily [2].

These problems in information discovery, to a large degree, result from inadequacies in the query model supported by most database indexes. In general, traditional index structures support a "one-shot" query model: users submit a query to the index, and the index returns a list of items in the database that fulfill the query. For very large collections, this model is inherently unwieldy. The large size of the collection forces users to form very precise queries, or risk getting enormous result sets that are essentially meaningless by virtue of their indigestibility. But forming very precise queries on a collection, about which a user may know very little, is likely to result in many queries that return empty result sets. The one-shot query model is appropriate when queries are issued by people who know the collection, its schema, its vocabulary, and its contents intimately. It does not fit casual users, who want to browse collections all over the world without having to know the associated schemas, vocabularies, etc., of those collections [5].

Many users also have an entirely different approach to information discovery than the one-shot model supports: they wish to develop their queries through browsing in an interactive fashion, iteratively forming new queries as they see the results of previous queries. Iterative queries, also called dynamic queries, are a powerful tool for information browsing and discovery in unfamiliar collections; by refining searches based on feedback about previous, broader queries, users can quickly narrow down their result set to items of interest, even in unfamiliar collections [15]. The traditional query model, however, does not take into account the notion that users may need several query/feedback steps to iteratively create their desired query, and it does not provide the user with the kinds of information that would be useful in further refining their search. There are two deficiencies: first, there is no facility for obtaining a "top-level" description of all the items in the collection, i.e., to answer the question "what kind of items does this collection contain?" Every query is issued "blind": the user never knows beforehand whether the collection contains items that might match the query. Second, there is no capability for providing high level summaries and feedback regarding the results that are returned for arbitrary queries, which

	Latitude			
Longitude	0^o	1^o N	2^o N	3^o N
0^o W	2	5	6	3
1^o W	8	2	9	2
2^o W	3	4	1	8
3^o W	5	4	6	3
4^o W	1	5	3	2

Figure 1. A hypothetical data cube presenting the number of satellite photos in a collection which cover given latitude, longitude pairs. Data are hypothetical.

would enable users to quickly digest large result sets to determine if their query needs refinement.

These problems can be remedied by presenting the user with summary information about any arbitrary query result set, i.e., aggregating the items in the result set in some manner. When the result set is very large, users are better served when they are given the results in summary form [15]. When a user searches for satellite photos, a graphical summary showing the numbers of photos found, organized by geographic region and date, might be very helpful in further narrowing the search. Such a summary of an entire collection would enable users to immediately ascertain whether the collection has what they are looking for. Or, summaries can show a user that the items they want are not in the collection at all, e.g. "this collection only has maps from the period 1900 to 1997", when the user is looking for 18th Century maps. Summarization capabilities are particularly helpful when a user first encounters a collection, e.g. when searching for interesting collections on the web.

In addition, when users start with broad or incompletely-specified queries, summaries and feedback regarding query results can be very useful in helping them narrow their query to manageable result sizes. The presentation of such summaries at the broad-query stage can prevent users from pursuing query directions that will result in empty result sets [2]. Summaries and aggregations can alert users to dimensions they may use to further refine their queries, and can help users discover trends that occur when they make modifications to their queries [15]. Efficient summary capabilities for arbitrary queries on large collections will be an essential element in providing the interactive querying and data browsing that users desire with very large collections, such as those in digital libraries. Traditional database index structures are not themselves designed to provide efficient aggregated summaries of arbitrary queries. To produce such summaries, a traditional index structure would have to retrieve all the items matching a query and use them to generate the desired summary; this can be very costly when query result sets are large.

The data cube [8], also known in the OLAP community as the multidimensional database [1, 14, 3], is a tool for information analysis which aggregates information contained in databases and data warehouses. A data cube is constructed from a subset of attributes in the database. Certain attributes are chosen to be measure attributes, i.e., the attributes whose values are of interest. Other attributes are selected as dimensions or functional attributes. The measure attributes are aggregated according to the dimensions. For example, consider a hypothetical database containing information about a collection of satellite photos. One may construct a data cube from the database with COUNT as a measure attribute, and LATITUDE and LONGITUDE as dimensions; such a data cube would provide aggregated total counts of all photos covering a given location on the Earth (Figure 1). We will limit our discussion to two dimensions merely for ease of presentation; in real-world applications there could be many additional dimensions, such as scale, date of publication, etc.

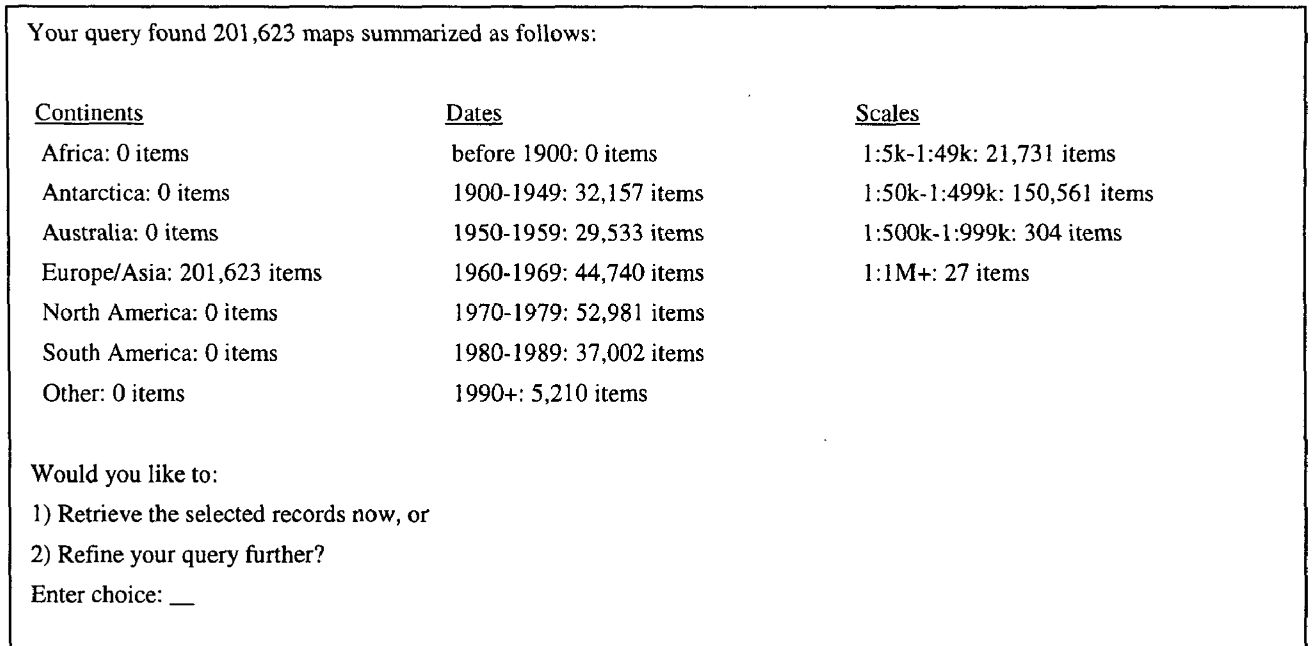
While use of the data cube has primarily been confined to data analysis and data mining applications, we will present techniques that permit the data cube to form the basis for efficient generation of summaries of query result sets. We further offer a technique for integrating classification hierarchies into the data cube framework, to support essential browsing capabilities in digital libraries.

Paper Organization The paper is organized as follows. We discuss related work in Section 2. Section 3 describes a means by which data cubes may be utilized to form the basis for efficient generation of summaries of user query result sets. Section 4 addresses the need to incorporate classification hierarchies into a browsing system, and presents a method of doing so within the data cube framework. Section 5 considers the case when multiple classification hierarchies are present for browsing. The paper is concluded in Section 6.

2. RELATED WORK

Researchers have noted the utility of providing summaries of query results. User studies involving systems that give visual feedback of query results have been shown to assist users in forming and refining their queries [2, 15, 18]. Other such systems [4, 9, 13] have incorporated the use of aggregations to enable easier understanding of large query result sets. These systems are typically designed to work on top of existing database management systems (DBMS), and do not directly address the question of efficiently supporting summaries of query results at the system level.

Other work has examined ways to generate summaries efficiently. Salzberg and Reuter [17] have proposed maintaining an auxiliary index and storing frequently used aggregation values into it. The method makes use of a multidimensional index, along with



```
Your query found 201,623 maps summarized as follows:

 Continents                  Dates                       Scales
  Africa: 0 items             before 1900: 0 items        1:5k-1:49k: 21,731 items
  Antarctica: 0 items         1900-1949: 32,157 items     1:50k-1:499k: 150,561 items
  Australia: 0 items          1950-1959: 29,533 items     1:500k-1:999k: 304 items
  Europe/Asia: 201,623 items  1960-1969: 44,740 items     1:1M+: 27 items
  North America: 0 items      1970-1979: 52,981 items
  South America: 0 items      1980-1989: 37,002 items
  Other: 0 items              1990+: 5,210 items

Would you like to:
1) Retrieve the selected records now, or
2) Refine your query further?
Enter choice: __
```

Figure 2. Sample interface showing a summary of the query results returned by a query for "items in Europe". Data are hypothetical

supporting data structures to increase the efficiency of synchronizing the auxiliary index with the main index. However, the authors note that updates are very expensive in their model, requiring 2^n updates to the auxiliary index for each update to the main index, where n is the number of dimensions, and they propose that such an index only be built when the data are not likely to change. This model is not appropriate for a library, since library collections are updated frequently. Johnson and Shasha have proposed hierarchically split cube forests to address the problem of obtaining summaries efficiently [12]. The approach results in efficient retrieval of summaries; however, the model requires space that grows exponentially as the number of dimensions increases and as the number of unique values in any dimension increases. The space requirements would not be scalable to a large library collection. We have proposed the Smart Index[5], which incorporates summary information into the nodes of a standard index tree, such as a b-tree. While this method is successful in providing summaries of arbitrary user queries, in general its query performance is related to the area enclosed by the user's query. Since many exploratory queries begin as large-area queries, it would be preferable to find a method that is insensitive to the query area.

In the area of data cubes, Ho et. al. have presented a method of calculating range sums in the data cube in constant time which we call the prefix sum method [11]. The relative prefix sum method [7] improves upon the Ho et. al. results by lowering the cost of updates to the structures; the dynamic data cube [6] further improves upon these methods by providing balanced query and update performance, and also gracefully handles sparse data cubes and data cubes that grow dynamically. These techniques may be applied when the measure attribute involves sum, count, average, rolling sum, rolling average, and other useful aggregates.

3. BROWSING IN NUMERIC DOMAINS

We envision browsing as an interactive process wherein a user begins with a broad query. At each step in the query process, the system provides feedback regarding the results of the query, at which point the user may modify their query further. Consider the following examples, for which we have constructed a hypothetical database that contains latitude, longitude, scale and date of publication information for the Alexandria Digital Library's collection of maps.

A user is looking for a map of Europe for a study he is planning on weather patterns. He begins by forming a query specifying the latitude and longitude boundaries of Europe, and submits the query to the DBMS. A traditional index would return a list of items falling within the specified coordinates; for ADL this would return approximately 200,000 items. This is a case when a list of items matching a query is indigestible for a user. A traditional system would not provide any information as to how the query might be modified so as to narrow the large result set; no summary information is available about the items in the query result set. A browsing system, however, would at this stage return only a summary of the items in the result set, which would be presented to the user via a suitable interface (Figure 2). The user would then be asked to either accept the query "as is", in which case the actual records would be retrieved, or to modify the query. This process of query, receive feedback is repeated until the user is satisfied that the query results are approximately what the user is looking for, and are also manageable in size. In this scenario, the user sees immediately that 200,000 items is too many for him, so he modifies his query to include scale information in the range 1:1M+. The summary information acts as a guide to these modifications, letting him know which ranges are populated [15]. Since the summary shows that there are no maps in this collection having dates before 1900, the user knows not to search in that range. Similarly, ranges that are highly populated may indicate areas in which the collection is particularly strong, such as maps of Europe from the 1970's.

Another user wishes to find maps of the town of Isla Vista from the 1930's. She forms a query using the latitude and longitude of Isla Vista and the date range 1930-1939, and submits the query to the DBMS. She receives a summary of the results of her query in the same form as Figure 2. Noticing from the summary of her query results that the collection does not have any maps of Isla Vista that fulfill her query, she decides to drop the date requirement and see what the collection contains for Isla Vista in general. She resubmits her query without the date information. The summary she receives shows that the collection has maps of Isla Vista, but mostly from the 1940's through the 1970's. She decides that maps from the 1940's will be sufficient, and modifies her query accordingly. The summary information enabled her to see how her query might be modified so that the system could return items of interest to her.

Summaries take many forms. When a user is searching for maps, suitable summaries might include the number of maps matching the query, perhaps along with breakdowns showing how many maps fall into various categories, date ranges, etc. The nature of summaries depends upon the utility of the summaries to the user population. This paper is concerned with summaries that are numeric in nature. It is assumed that the user's query will take the form of ranges along several dimensions, e.g. a range of dates of publication, a range of latitudes and longitudes covered by the map, etc. The area enclosed by a query is defined to be the area of the multidimensional bounding box formed by these ranges. A method is required that will efficiently generate summary information for the result sets of user queries, and do so in a manner that is insensitive to the size of the area enclosed by the query.

When the summaries are numeric, their generation is related to the notion of range sum queries in data cubes. Range sum queries aggregate the measure attribute in the data cube within the range of the query. Queries of this form are currently used in data analysis environments for finding trends and discovering relationships between attributes in the database; however, we may adapt them to our browsing requirements. Returning to our data cube example, a typical range sum query would calculate the number of satellite photos in the collection which cover a given region on the planet (Figure 3). In the figure, the values in the shaded region are aggregated (in this case, summed) to produce the summary corresponding to the area of the user's range query.

Since range sum queries calculate summaries for arbitrary range queries, they may be employed to provide the summarization capabilities necessary for browsing. The sum of the cells in the shaded region provides the summary we desire: the number of photos falling within the range of the user's query. However, we would not use the naive approach of actually summing the cells in the range of the query; in the worst case, this would involve summing every cell in the data cube, and thus would not be an efficient approach. There are currently several efficient methods which calculate range sum queries in data cubes, as briefly noted in Section 2. While a detailed description of the workings of these methods falls outside the scope of this paper, two of the methods, the prefix sum method [11] and the relative prefix sum method [7], efficiently calculate these range sums in constant time by precomputing various sums of the data cube. The dynamic data cube method [6] goes a step further by providing both efficient queries and updates on the data cube, and allows for dynamic growth of the data. The size of the shaded region, i.e. the area enclosed by the query, has no impact on the query performance of these methods; thus, these methods are suited for browsing environments, where it is expected that the area enclosed by user queries will vary greatly.

A data cube would be constructed for the dimensions of interest in the collection and stored using one of these methods. Each cell of the data cube would contain appropriate summary information for the items in the collection. While the figures show only one summary value per data cube cell, in general the summaries contained in each cell can be comprised of many values. For example, as in Figure 2, the summary stored in each cell might consist of an array of integers that count the number of library items falling into different continents, decades, and scales. A summary of the results of any query could then be calculated very efficiently using the range sum techniques.

4. CLASSIFCATION HIERARCHIES

The range sum methods discussed in Section 2 may be used to obtain the summaries we need, as long as the domains of the dimensions are numeric. However, digital libraries also utilize domains that are non-numeric, such as classification hierarchies. Support for browsing utilizing classification hierarchies is an important tool for users of digital libraries; and, in fact, is a general functionality that is useful in many database applications. In this section we argue for the importance of including classification hierarchies in the browsing environment. We provide a mechanism for incorporating classification hierarchies into data cube techniques to support the summarization capabilities required for browsing.

	Latitude			
Longitude	0°	1° N	2° N	3° N
0° W	2	5	6	3
1° W	8	2	9	2
2° W	3	4	1	8
3° W	5	4	6	3
4° W	1	5	3	2

Figure 3. A range sum query over a data cube.

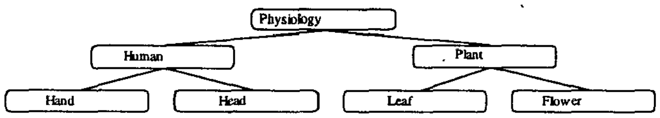


Figure 4. A simple classification hierarchy.

4.1 The Importance of Classification Hierarchies in Browsing

Classification hierarchies provide an organization to data that is uniquely suited to digital library browsing because they organize collections from a user's point of view. In contrast, there are some algorithmic methods that may be used to discover arbitrary associations and relationships in data. For example, Gibson et. al. [10] have sought to discover associations between categorical data items in a database. The method seeks to cluster related items, so that relationships between the data items will be come apparent. However, this method, and other methods that seek to find associations between data in the database (e.g., see the work done by H. Chen et. al.), may not be appropriate for data browsing environments in digital libraries. This is due to the fact that, while such methods are designed to discover arbitrary associations between the data items in the database, the exact nature of the relationships they reveal may not always be obvious. Gibson et. al. show that, in a database of automobile sales information, such methods may reveal that there is a relationship between the sales patterns of automobiles built by the manufacturers "Toyota" and "Honda." However, the exact nature of the relationship is not necessarily discovered by the methods; the methods reveal only that the sales figures for automobiles built by these manufacturers appear to be related in some way. Perhaps the association is due to the fact that these companies are both based in Japan and are subject to the same factors that influence overseas shipping, or perhaps they both are subject to the same seasonal patterns of car purchasing.

Even when the exact nature of the discovered relationships is known, these relationships may not be the ones that users would choose to browse over. The relationships these methods discover are derived from the data items in the database; one could say that they present the database from the data's point of view. Thus the relationships that are revealed are ones that are interesting from the data's point of view; for example, that "Toyota" and "Honda" are related. While data analysts are eager to discover such relationships, users who are browsing may be baffled when data is organized in this manner. When the database is structured from the data's point of view, it may not be clear to the user how the data is organized, why certain items are grouped together, and where a desired item may fall within that structure. Users desire the data to be structured in a way that makes sense from the user's point of view. The purpose of the browsing environment, after all, is to present the data in a structured way that facilitates the user's discovery of desired information. When the very structure of the information is not clear, the browsing process is derailed. After a user has found an item of interest, it may then be useful to ask the system to display related items; however, the user still has to be able to find that first item. Methods that discover arbitrary relationships in the database are not likely to be suitable for the primary browsing environment.

Classification hierarchies, on the other hand, are particularly suited to browsing. Classification hierarchies, such as the Library of Congress Subject Headings (LCSH), organize information in a manner that is hierarchical, consistent, user-understandable, and eminently navigable. Classification hierarchies allow users to begin at a high-level category, such as "Physiology," and proceed to narrow their search, to "Physiology:Human," for example. Indeed, libraries already organize their holdings using classification hierarchies, and users are familiar with the notion of using these hierarchies to refine their query.

Existing range sum methods in data cubes may be used to support browsing in numeric domains. A new method is desired that would allow classification hierarchies to be incorporated into the browsing environment. We offer a solution which permits the incorporation of classification hierarchies into the range sum techniques described, thus creating a powerful, unified environment for database browsing.

4.2 Data Cube Based Aggregation of Classification Hierarchies

Figure 4 shows an example of a simple classification hierarchy presented as a tree. A user may "browse" such a classification tree, using an appropriate interface, by beginning at the root. Shneiderman et. al., for example, have proposed a number of interesting interface techniques for navigating hierarchical structures. The user would refine their search by choosing a node to visit. The user may continue to ascend or descend the various nodes of the tree, until the user is satisfied that their position in the classification tree is consistent with the intent of their query. While the tree in the figure is balanced, this is not required.

We have envisioned browsing as an iterative process, wherein the user forms a broad query encompassing the general area of interest, and refines (narrows or broadens) the range of the query depending on the intent of the query and the results of previous queries. Our goal in earlier work was to generate summary information quickly for the result set of an arbitrary user query to enable the iterative browsing process. As we have argued, such summary information is useful in assisting the user to make decisions regarding the refinement of their search. In the case of classification trees, this means presenting summary information for each node visited; this information summarizes the data items that fall under the node.

Our previous methods support efficient summary generation for arbitrary range queries over numeric domains. The classification tree is not a numeric domain, and does not obey the same properties as a numeric domain; as a result, it is not immediately apparent how range searching relates to such a tree. In the figure, we observe that the nodes Head and Hand are both children of the node Human, but have no natural ordering between them. Furthermore, the nodes Head and Leaf also have no ordering between them. The notion of "range searching" is clearly not generally applicable to such a tree. For example, it would not make sense to speak of a query having the range (Head..Leaf);

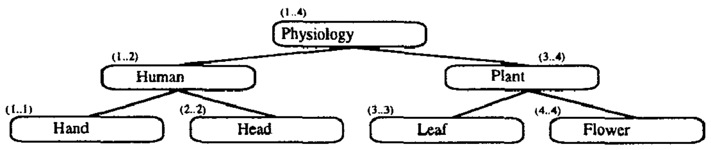


Figure 5. Classification hierarchy with node ranges.

since there is no ordering between the nodes Head and Hand, it is not clear whether or not such a range includes the node Hand. Furthermore, from a semantic point of view, the range query (Head..Leaf) is nonsensical.

We observe, however, that it does make sense to say that a query has a range (Human); and clearly the node Human may be seen as enclosing the nodes Head and Hand. Thus, each individual node of the classification tree can be seen as itself representing a range, i.e., the range that encloses all nodes of its subtree. When a range is defined in this manner, we do not encounter the problems inherent in ranges such as (Head..Leaf). Since all children of a node are included in the node's range, the lack of ordering between siblings is not an issue; also, the semantics of this definition of range are consistent with the expectations of users. A user whose query is, "show me what this database contains regarding Physiology:Human," would expect to retrieve information regarding all nodes in the subtree rooted at the node Human in the tree.

When range is defined in this manner, we may employ a straightforward process to map nodes in a classification tree to an integer domain so that the methods of previous sections can be utilized. The mapping must obey two properties:

1) The range assigned to a node must always enclose the range of all nodes in its subtree.

2) The roots of two subtrees that are disjoint must have ranges that are disjoint.

We present a brief algorithm for numbering nodes in a tree such that the resulting node ranges obey these properties. Leaves in the tree are given the singleton range, e.g. (1..1). Each leaf is assigned a unique range that proceeds incrementally from the left-most child to the right-most child. Each parent node encompasses the range that is a union of all ranges of its children. Since the children are numbered sequentially, the range of the parent will be continuous. Furthermore, the ranges of siblings will be disjoint.

Figure 5 shows the classification tree as numbered by the algorithm. The leaves are assigned unique, incrementing, disjoint singleton ranges from left to right. Note that the relative numbers assigned to siblings have no importance. For example, node Hand was arbitrarily assigned the range (1..1), while Head was assigned (2..2); the assignments for these nodes could just as well have been reversed, since there is no inherent ordering between these nodes. Similarly, the subtree rooted at node Plant could have been numbered before the subtree rooted at Human. The relative numbering of siblings is not important because, as already discussed in the context of the range (Head..Leaf), range queries are only defined when the range of a single node is specified. It makes semantic sense to query over the "range" of Physiology, but not over the range (Hand..Leaf). Since a parent always encloses the range of all its children, either all children of a node will be included in the resulting answer, or none will; therefore, the relative numbering of siblings is unimportant. Of course, a user may still form a query that is a union of disjoint ranges, e.g. "Find Physiology:Human OR Physiology:Plant:Leaf." Since the ranges of these nodes are disjoint, such a query would be answered by summing the results of the two disjoint queries Physiology:Human and Physiology:Plant:Leaf. This is consistent with typical database querying models.

On a system level, this method of constructing node ranges permits a classification tree to be used as a dimension of the data cube. The classification tree is numbered as described. Each leaf of the resulting tree has a singleton range. Summary information for each leaf is then placed in the data cube cell corresponding to the range of the leaf. The user navigates the classification tree via an appropriate user interface; the user need never be aware that nodes of the classification tree are assigned ranges "behind the scenes." When the user visits a node, the system takes the range of the node and uses it in conjunction with the data cube methods described earlier to generate summary information for the user's query. Since the range of a parent node encloses the ranges of all nodes in the parent's subtree, the summary so generated will accurately reflect all items in the database that belong within the node's classification.

4.3 Browsing with Multiple Classifications

There are many useful classification hierarchies for describing collections. For example, a set of library books may be classified using LCSH. If some books are in the computer science domain, they may also be classified using ACM. Each classification hierarchy approaches a collection from a unique viewpoint. It may be beneficial for users to have several, or many, classification hierarchies at their disposal when they are browsing.

To support browsing with multiple active classifications, each classification hierarchy may be placed in its own dimension in the data cube. The data cube may contain many dimensions; using the techniques described here, each dimension may be comprised of a numerical domain, such as date of publication, or of a classification-based domain. Each of these dimensions is a component of a user's browse query. Using an appropriate user interface, a user may modify the range of any or all of the dimensions in the process of narrowing down their search. When a user has not specified a range in a given dimension, the entire range of the dimension is assumed; since the range sum methods provide performance that is independent of the area enclosed by the query bounding box, there is no performance penalty for the large query areas thus formed. Thus, the techniques we have described permit browsing while simultaneously modifying query ranges along many dimensions, thus creating a powerful browsing environment.

5. CONCLUSION

The data cube has been proposed for analytical processing environments. The powerful techniques that have been developed in this area are also useful in other domains. We have shown that data cubes can be used in a library context as a means of efficiently generating summaries of the results of user queries; such summaries can be very helpful during interactive searches of library collections. We have presented a means by which classification-based information can be incorporated into the data cube. The combination of the techniques we have presented allow a user to interactively modify a query through browsing, utilizing many different dimensions simultaneously; the dimensions can be a mix of numeric domains and classification hierarchies. Several classification hierarchies can be active for the same data, and users can interactively browse through the data using the different viewpoints provided by each hierarchy. At each step of the query process, the user is presented with summary information regarding their query results as an aid to further query refinement. The methods we have presented provide system-level support for the powerful database browsing environment we have envisioned. We are currently in development of a system which utilizes these techniques to provide a browsing environment for digital libraries.

6. ACKNOWLEDGEMENTS

Portions of this research have been supported by NSF under grant number IRI94-1133.

7. REFERENCES

[1] R. Agrawal, A. Gupta, S. Sarawagi. Modeling multidimensional databases. In Proc. of the 13th Int'l Conference on Data Engineering, Birmingham, U.K., April 1997.

[2] C. Ahlberg, C. Williamson, B. Shneiderman. Dynamic Queries for Information Exploration: An Implementation and Evaluation. In Proc. CHI'92: Human Factors in Computer Systems, 1992.

[3] E. F. Codd. Providing OLAP (on-line analytical processing) to user-analysts: an IT mandate. Technical report, E.F. Codd and Associates, 1993.

[4] K. Doan, C. Plaisant, B. Shneiderman, T. Bruns. Query Previews for Networked Information Systems: A Case Study with NASA Environmental Data. SIGMOD Record, 26(1):75-81, March 1997.

[5] S. Geffner, A. El Abbadi, D. Agrawal, T. Smith, M. Larsgaard. Smart indexes for efficient browsing of library collections. In Proc. of IEEE Advances in Digital Libraries Conference, pages 107-116, Santa Barbara, California, April 1998.

[6] S. Geffner, D. Agrawal, A. El Abbadi. The Dynamic Data Cube. Submitted for publication.

[7] S. Geffner, D. Agrawal, A. El Abbadi, T. Smith. Relative Prefix Sums: An Efficient Approach for Querying Dynamic OLAP Data Cubes. To appear in Proc. of the 15th International Conference on Data Engineering, Sydney, Australia, March 1999.

[8] J. Gray, A. Bosworth, A. Layman, H. Pirahesh. Data Cube: A relational aggregation operator generalizing group-by, cross-tabs and sub-totals. In Proc. of the 12th Int'l Conference on Data Engineering, pages 152-159, 1996.

[9] J. Goldstein, S. Roth. Using Aggregation and Dynamic Queries for Exploring Large Data Sets. In Proceedings Computer Human Interaction '94, 1994.

[10] D. Gibson, J. Kleinberg, P. Raghavan. Clustering Categorical Data: An Approach Based on Dynamical Systems. In Proceedings of the 24th International Conference on Very Large Databases, 1998. D. Gibson, J. Kleinberg, P. Raghavan. Clustering Categorical Data: An Approach Based on Dynamical Systems. In Proceedings of the 24th International Conference on Very Large Databases, 1998.

[11] C. Ho, R. Agrawal, N. Megiddo, R. Srikant. Range Queries in OLAP Data Cubes. In Proc. of the ACM SIGMOD Conference on the Management of Data, pages 73-88, 1997.

[12] T. Johnson, D. Shasha. Hierarchically Split Cube Forests for Decision Support: description and tuned design. New York University Department of Computer Science, Technical Report TR1996-727, November, 1996.

[13] M. Livny, R. Ramakrishnan, K. Beyer, G. Chen, D. Donjerkovic, S. Lawande, J. Myllymaki, K. Wenger. DEVise: Integrated Querying and Visual Exploration of Large Datasets. SIGMOD Record, 26(2):301-12, June 1997.

[14] The OLAP Council. MD-API the OLAP Application Program Interface Version 5.0 Specification, September 1996.

[15] B. Shneiderman. Dynamic Queries for Visual Information Seeking. IEEE Software, 11(6):70-77, November 1994.

[16] T.R. Smith, J. Frew. The Alexandria Digital Library. Communications of the ACM 38(4):61-62, April 1995.

[17] B. Salzberg, A. Reuter. Indexing for Aggregation. In High Performance Transaction Systems (HPTS) Workshop, 1995.

[18] C. Williamson, B. Shneiderman. The Dynamic HomeFinder: Evaluating Dynamic Queries in a Real Estate Information Exploration System. In Proc. 15th Annual International SIGIR '92, 1992.