# VERY LARGE DATABASES

A growing number of database applications require on-line access to large volumes of data to perform a variety of tasks. For example, telecommunication service providers need to store terabytes of phone call data to satisfy the requirements of billing, fraud detection, and strategic planning. Furthermore, many of these applications require support for new types of digital data, such as images and video. For example, a detailed requirements analysis for NASA's Earth Observing System indicates that, at the turn of the century, the daily growth in stored image data will be 2.7 Tbytes, and the total stored volume will exceed 1.5 Pbytes. In this article, such new application domains for database management will be identified, and the issues that arise from large data volumes, application-specific requirements, and new types of data will be discussed.

## APPLICATION-ORIENTED DATABASE MANAGEMENT SYSTEMS

Database management systems (DBMS) are designed to provide the data storage and manipulation functions common to tasks that depend on very large volumes of data. Economic and technological changes, including the development of high-speed networking, are fueling a new family of data-intensive applications. Traditional DBMS applications, such as banking applications, required fast access by multiple users to large, dynamic datasets. To meet these requirements, traditional DBMS support on-line transaction processing (OLTP), using transactions as the basic mechanisms for ensuring data consistency in the face of concurrent updates by a host of users. The data are typically highly structured and represented in a structured data model such as the relational model. In contrast, the new applications discussed in this section may require infrequent updates and the queries may be more complex, including aggregation and intricate pattern-matching queries. In addition, the data may be less structured or completely unstructured. Some of the most prevalent of these applications, and the underlying DBMS support technologies will be described.

### Data Warehouses

Data warehouses provide integrated access to historical data collected from legacy data sources (1). In a typical business, numerous on-line software systems manage and collect data as part of the daily operation of the company. These systems may be transaction-processing systems that use a traditional DBMS, or they may be specialized applications that squirrel away data in files. The data used by these different applications hold valuable information about past business decisions and outcomes that can be used to improve future decisions. To accomplish this, warehouses integrate the data under a unified schema (structure) and provide access mechanisms that enable efficient use by analysis and decision-support packages (see Fig. 1).

### On-Line Analytic Processing (OLAP)

OLAP refers to the statistical analysis of data in support of decision-making tasks. In OLAP, the focus of data management shifts from one of ensuring consistency and durability of data to one of providing flexible, convenient access to data. As a result, many of the principles that guided the development of data management solutions for OLTP (e.g., the need to minimize data replication and to normalize data) do not apply to OLAP.

### Digital Libraries

A digital library is an electronic version of a classical library in which the information resources (e.g., books, art work, films), and the indexing information used to locate resources are stored digitally (2). By its nature, a digital library must be able to store and manage a highly heterogeneous collection of data, ranging from unstructured data (e.g., images or videos) to semistructured data (hypertext documents) to structured data (descriptive metadata). Digital libraries use techniques from both information-retrieval and structured databases, and extend these with new browsing and searching techniques.

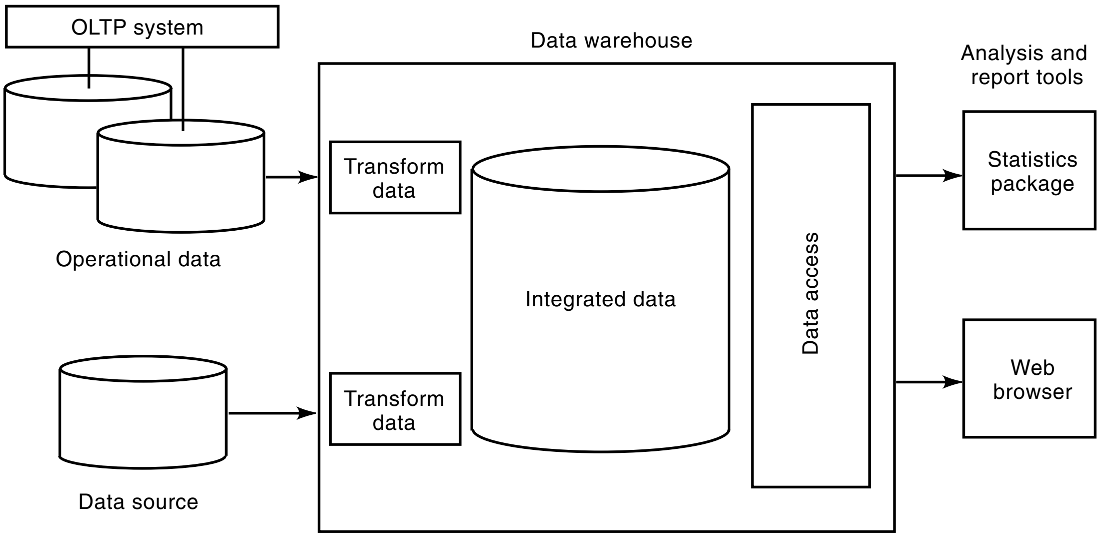

**Figure 1.** Data warehouse architecture.

### Statistical and Scientific Database Management Systems (SSDBMS)

Statistical DBMS are designed to manage socioeconomic datasets (e.g., census data or economic forecasting data) (3). Scientific DBMS manage complex collections of data used in and gathered from experiments and other scientific endeavors. As in OLAP, SSDBMS must support sophisticated browsing, summarization, and analysis functions. In addition, this support must be provided over a diverse collection of complex data, including not only numeric and text data, but also data with complex types. These types may represent such objects as molecular structures, terrain maps, or architectural plans.

### World Wide Web and Databases

The use of DBMS to store World Wide Web (web) content has proven to be an effective means of creating dynamic, scalable web servers. Using interfaces such as the common gateway interface (CGI), web application programs can access DBMS to retrieve static web pages or to dynamically create pages based on query results. While DBMS may be used to store web content, the web also permits the electronic publishing of existing (or legacy) databases. Users of published databases, unlike traditional DBMS users, are typically unfamiliar with the data and structure of the database. As a result, users may be unable to effectively formulate structured queries and require new solutions for browsing and effectively locating data in large, complex datasets (4).

To meet the data-management needs of these emerging applications, new support technology has to be incorporated into DBMS. This new technology is examined, including extensions to data models, query languages, indexing methods, query processing engines, and query optimizers.

### Data Model

Traditional DBMS use structured data models such as the relational model, hierarchical models, or object-oriented models. Structured data models assume that data can be grouped into collections of tables or classes, each having a well-defined structure (or schema). To accommodate the needs of new applications, data models have been extended in three primary directions: direct support for abstract data types; addition of conceptual structures to help in the summarization and browsing of large, complex data collections; and support for unstructured and semistructured data. Each of these extensions are examined in turn.

**Abstract Data Types.** Traditional DBMS support a fixed set of simple data types (e.g., integers and dates). Extensible DBMS can be extended dynamically with user-defined types and functions. These types can be used to model complex objects, for example, molecular structures, along with the behavior of these objects. Most commercial DBMS (including Informix, DB2, and Oracle) now provide such extensibility. To fully support these new data types, a DBMS must provide data-management support, including new indexing- and query-processing techniques.

**Multidimensional Models.** In data warehousing, OLAP, and statistical applications, data are often conceptually modeled as having multiple dimensions. For example, census data can be viewed as having the dimensions profession, age, years-at-current-address, and so on. Product design data can be viewed as having dimensions designer, product type, date-of-production, etc. In these examples, the tables containing census or product data are called fact tables. Such a multidimensional view of data facilitates the direct modeling of potentially complex relationships among dimensions. For example, the date-of-production dimension may be refined into subdimensions—day-of-sale, month, and year. There is a functional relationship from data-of-production to each of its subdimensions. A multidimensional model also facilitates the expression of aggregation or summarization of data along different dimensions (or subdimensions) of interest to a user. For example, a user may retrieve the number of people over age 35 in each state who have technology-related professions. The explicit modeling of dimensions provides a convenient formalism on which language operators for aggregation and summarization can be built.

**Unstructured and Semistructured Models.** Traditional information retrieval (IR) systems use unstructured data models to represent data. Data are stored in *documents* of arbitrary type and structure. Hence, documents may be images, video sequences, or full-text data stored in any format. Each document is modeled as a bag of words (which may be a subset of the words in a document or a set of words describing an image or video). No structure is associated with these words so a document may contain the word `Washington`, but the model does not include information on whether `Washington` is the author, the subject, or the location of the document. Unstructured models are appropriate for data that truly have no inherent structure. However, they fail to provide sufficient functionality when used to model data (such as web pages) that have some structure. Consider an XML document which may have tags indicating the author, creation date, and title of the document along with large portions of unstructured data (e.g., the body of the document). Using an unstructured data model, a query could not be posed to retrieve web documents written by `Washington`. Using a structured data model to represent this information is equally unsatisfying, since web documents rarely share the same structure. At best, one could define a table containing attributes common to most documents. To provide better support for such data, semistructured data models have been developed. These models are often self-describing data models in which a data object is described by both a value and its structure. Hence, each object may have its own unique structure. In addition, these models often permit objects to be associated with other objects, typically using labeled graphs (5).

### Query Language

In structured data models, the structure (or schema) is used as the primary vehicle for querying. In structured query languages (e.g., SQL or OQL), schema components (e.g., attribute, relation, or class names) are used to specify what data should be retrieved. Hence, the user must know and understand the schema in order to pose queries. In unstructured data models, the query model is based on keyword matching. A set or Boolean combination of user-specified keywords are matched against the words representing the stored documents. To support efficient querying, indexes such as inverted indexes are used to quickly map keyword(s) to documents which they annotate. Sophisticated techniques are used to ensure that all relevant documents are retrieved and no irrelevant documents. These techniques include linguistic techniques for detecting synonyms among keywords.

Query languages for semistructured data models permit the specification of structured queries over data objects with known structure. However, given that each data object may have its own structure, understanding the structure of an entire database may not be feasible. In a data warehouse, where the structure of the data may be extremely complex, users may need to pose queries without knowing the full structure. As a result, semistructured query languages permit the specification of pattern-matching style queries (e.g., "Find all building plans designed by Maria that include a heating system with more than 100 subcomponents.") Such queries permit the browsing and location of data in unknown or partially known structures (6).

Pattern-matching queries are also useful in querying heterogeneous structures. Multidatabase languages provide additional data restructuring and merging operations to facilitate data integration. Metadata, which is descriptive information about database schemas, can be extremely valuable in enabling the integration of heterogeneous data sources. Higher-order languages that permit the querying of schemas along with the data have been used successfully in heterogeneous DBMS.

Data warehouses provide powerful aggregation and summarization facilities to permit the extraction of relevant information. The aggregation functions typically include the basic functions provided in SQL (and object-based variants of SQL) for computing counts, averages, sums, maximums, and minimums, along with more sophisticated statistical functions over numeric data. Some DBMS permit the user to define new aggregation functions. The summarization techniques extend the simple horizontal partitioning permitted by the SQL *group by* operator. The group by operator partitions the tuples of a table (or instances of a class) into groups based on the values of a set of attributes. Aggregation functions are then applied to each group to compute a summary for each set of attribute values. Extensions permit the partitioning to be based on the values of any function applied to the table attributes. The *cube* operator is used to compute cross-tabulations on a table (7). In contrast to the group by operation, a cross-tabulation includes subtotals (or subaggregates) for every subset of attributes.

Browsing techniques provide a convenient way of introducing the data and schema to new users. Users can navigate through the data, effectively locating data of interest. Browsing techniques can be broadly grouped into two strategies. The first uses concept classifications from library science to logically organize the database. Documents are associated with concepts in the classification, such as agriculture or welding. Concepts are related to each other based on their semantic relationships. Users can browse the concept classification, which is often presented using hypertext, to locate documents. The second type of browsing technique uses OLAP style summarizations of the database to permit users to locate data of interest (4). These systems group together subsets of the database and present aggregates of the data items in each group. Hierarchical abstractions, or dimensions, over the data are used to form the aggregates. A user may *drill-down* into a set of data by successively restricting one or more of the dimensions, while the system presents aggregates of the underlying data at each step.

### Indexing

New data-intensive applications require much more complex forms of querying. This complexity can take on many forms. In semistructured data, path queries, which retrieve a subset of objects directly or indirectly associated with a given object, are common. In OLAP and SSDBMS, aggregate queries and multidimensional queries are common. In many of these application, queries may join multiple tables (e.g., a fact table may be joined with many dimension tables). These query characteristics require the development of new indexing mechanisms for enhancing query performance (8,9).

To support complex queries over multiple tables, multitable indices have been developed including join indices and star indices. These indices materialize (i.e., cache) common joins enabling complex queries to be performed efficiently. Traditional indices have been generalized to enable the indexing of new user-defined types. To support queries with independent selection conditions, bitmap indices may be used. A bitmap index is a modification of a traditional index (e.g., a $B^+$-tree or hash index), where for each index value (or key) the index stores a bitmap representing which tuples contain the given value. Bitmap indices have also been shown to be useful in enhancing the execution of some aggregate queries. Other specialized access structures are tailored to materialize specific, commonly used queries (e.g., projection indices). The update characteristics of these structures may be unacceptable for OLTP applications. However, for read-only or read-mostly applications, the improved query speed may offset any additional update cost. Data may also be replicated and stored under materialized views. Many data-management products make extensive use of materialized aggregate views, including materialization of the data cube, to permit fast computation of aggregate summaries.

### Query Processing

The query language extensions and new indexing structures outlined above introduce a variety of new challenges for query processing. The proliferation of new physical structures for accessing data has required the development of new techniques for determining when an index structure or materialized view can be used (correctly) in answering a query (10). Similar techniques have also been applied to heterogeneous DBMS to enable query processing over heterogeneous views of data (11–14). New efficient algorithms for computing the data cube and other aggregate queries have been developed and incorporated into commercial query engines.

### Query Optimization

To complement the new query-processing strategies, new techniques for query optimization have been required. Given the new language operators and the new access methods available, the task of deciding which combination of operators and which indices or view to use in executing a query has become significantly more difficult. Query optimization is already a complex task in conventional relational systems. The challenge for new applications is to introduce new operators and access structures in a way that does not adversely affect the performance or quality of the query optimizer.

Recent research has addressed some of the issues involved in optimizing aggregate queries and queries with expensive (possibly user-defined) functions. Magic sets, and their cost-based extensions, have proven valuable in optimizing complex relational queries, including queries over views (15). Algebraic and cost-based optimization of queries over heterogeneous DBMS has also been addressed, though much work remains to be done [see (16) for a summary]. Work on optimizing queries over semistructured data has just begun.

## MULTIMEDIA DATABASE MANAGEMENT SYSTEMS

Recent advances in computing, communication, and storage technologies have enabled the proliferation of *multimedia data,* such as images, graphics, audio, video, and animation, in a number of diverse application domains. Examples of such domains include digital libraries, satellite image archival and processing, training and education, entertainment, and medical databases containing X rays and MRIs. Currently, the bulk of multimedia data reside in either conventional or multimedia storage servers, offering special-purpose, application-specific functionality. This situation, however, raises a number of problems, including redundancy, inconsistency, concurrent access anomalies, as well as integrity, atomicity, and security problems. The continuously expanding diversity of multimedia applications and volume of multimedia data further exacerbate the problem. Incorporating database technology in multimedia application development can offer several benefits, including declarative query facilities, transparency from physical aspects of storage, associative access through indexing, data consistency through well-defined access methods, multiuser access through concurrency control, and reliability through recovery mechanisms. This understanding has given rise to a significant amount of recent interest in *multimedia database management systems* (17–21).

Providing database functionality for multimedia data types presents a host of new challenges not addressed by conventional DBMS. These challenges stem from the fact that multimedia data types differ from traditional alphanumeric data in their characteristics and, hence, require different techniques for their organization and management. A first distinguishing characteristic of multimedia data is their *volume*—a JPEG-compressed color image can require several megabytes of storage space, and a 100 min video compressed using the MPEG-I standard requires about 1.25 Gbytes of storage space. Conventional DBMS and file systems provide only very limited support for such large objects, typically in the form of special data types, such as *long fields* and *BLOB*s (binary large objects), with very poor semantics. Reducing multimedia data to single, large, uninterpreted data values is clearly inadequate for supporting the rich semantic content of multimedia data types and places the whole burden of data processing within the application. A second, and perhaps most important, characteristic of multimedia data types is that, in contrast to alphanumeric data, they are typically characterized by a *spatial extent* (e.g., images and graphics), a *temporal extent* (e.g., audio and speech), or *both* (e.g., video). As a consequence, multimedia data have much richer semantics than conventional symbolic data and any meaningful interpretation of a multimedia object is typically based on its relationship to a system of spatial coordinates and/or a constantly progressing time scale. Furthermore, time-dependent multimedia data [also known as *continuous media* (CM) data], like audio and video, have specific *timeliness* constraints associated with them. For example, a video clip consists of a *stream* of video frames which must be delivered to viewers at a certain rate (typically 30 frames/s). For MPEG-I compressed video, this translates to a data rate of approximately 1.5 Mbps (megabits per second). The underlying storage manager needs to ensure that the storage and retrieval of CM data proceeds at their prespecified real-time rates (22). Integrated support for the spatiotemporal nature and semantics of multimedia data requires nontrivial extensions to various basic building blocks and functional units of a DBMS.

### Data Model

Complex multimedia objects require sophisticated modeling mechanisms with rich semantic capabilities. An important requirement for these conceptual tools is the ability to model the complex spatiotemporal structure of a multimedia object through well-defined abstractions. For spatial noncontinuous data, like images, the modeling problem is probably simpler, since the semantics of objects and operations are clearly defined and their properties can be derived from geometry. CM data, on the other hand, present the much more difficult problem of modeling *time* with conceptual mechanisms that can capture: (1) *intramedia continuity,* that is, the real-time delivery requirements of a CM stream; (2) *intermedia synchronization,* that is, the precedence and real-time synchronization constraints among the component CM streams of a complex multimedia presentation (e.g., audio and video lip-synching); and (3) *user interaction,* that is, the ability of a user to interact with the presentation through standard VCR-type functions (e.g., fast-forward or rewind), which can change the presentation speed or randomly access specific points in a presentation.

Most efforts for managing multimedia data have been based on flexible object-oriented or extended relational models that allow for the modeling of complex structured multimedia objects, the definition of abstract media types, and operations on media data units. However, despite their ability to model complex structures, such data models lack the temporal modeling capabilities required by CM data—the problems of stream-oriented, real-time access, and synchronization still remain. A number of conceptual models have been developed for capturing the temporal aspects of multimedia data. They can be roughly classified into three categories, namely: (1) *graph-based models* [e.g., object composition petri nets (23) and presentation graphs (19)], *language-based models* [e.g., HyTime (24) and MHEG (25)], and *temporal abstraction models* [e.g., temporal intervals and relations (26) and timed streams (27)]. Nevertheless, the efficient mapping of such conceptual constructs to the physical level of a full-fledged multimedia DBMS still remains an issue of concern.

### Query Language

Declarative query languages are an important part of DBMS and have played an important role in their success. A powerful declarative querying facility allows associative (i.e., content-based) access to the underlying data and helps to maintain the desired independence between the DBMS and the application. Conventional DBMS query languages are typically based on the assumption of highly symbolic alphanumeric representations, and thus cannot accommodate the much richer spatiotemporal semantics of multimedia data. More specifically, query languages for complex multimedia objects need to address the following issues:

1. *Similarity Queries.* Conventional declarative content-based querying is based on *exact-matching* between well-defined sets of symbols using simple equality or comparison operators. An example of such a query is: "Select all employees with salary >45K." For any employee in the database, the search condition will evaluate to either `TRUE` or `FALSE`, based on a well-understood numerical comparison. Such exact matches are rarely of interest for multimedia data types such as images or video. Users are usually interested in discovering multimedia objects that are perceptually *similar* (to each other or to some query object), where the notion of similarity typically depends on the data type and the requirements of the application. Answers to such similarity queries will be *ranked,* based on grades of similarity obtained using an appropriate similarity function and users will usually be interested in obtaining the `TOP-k` results, that is, the objects with the `k` highest grades (28,29).
2. *Spatiotemporal Queries.* A complete declarative query facility for multimedia DBMS should allow users queries not only on the content, but also the *structure,* that is, the spatiotemporal characteristics of multimedia objects. Examples of such queries include spatial searches (e.g., "Find all the roads passing through Murray Hill."), temporal searches (e.g., "Find all scenes where President Clinton is shaking hands *after* stepping off an airplane."), and simple spatial or temporal computations (e.g., "Find the area of this object."). Of course, supporting spatiotemporal queries is intimately related to the facilities offered by the underlying data model for modeling the complex spatial and temporal structure of multimedia objects.
3. *Quality of Service (QoS) Specifications.* Multimedia objects can often be accessed at multiple levels of resolution or *quality of service (QoS)* that correspond to different service requirements on the underlying DBMS resources. Important QoS parameters include the average delay (experienced by the user), the actual presentation rate and image resolution, and the allowable deviations for temporal synchronization constraints. Some application environments can be flexible about certain QoS parameters (e.g., audio quality or image resolution). Since such flexibilities can be directly translated to flexible resource requirements, effective QoS specifications play a very important role in effective query processing and optimization in a multimedia DBMS (30).

### Indexing

Similarity-based queries are the prominent form of associative data access in a multimedia DBMS. Efficient execution of such queries requires the development of appropriate indexing mechanisms for retrieval by similarity. The standard technique for this purpose is to map both the query and each multimedia object into some multidimensional *feature space,* such that two perceptually similar objects are guaranteed not to be far apart in this space (18). Typical features of multimedia objects include color, texture (e.g., contrast, coarseness), shape, text (i.e., a set of keywords or annotations), and motion. There can also be some features specific to particular application domains. Features are extracted either manually or using automated (usually domain-specific) methods, and stored as a collection of feature vectors in the database. For example, the QBIC (query by image content) system developed at IBM Almaden supports queries based on example images, user sketches and drawings, color, texture, shape, and keywords (29). Color-based querying is implemented by storing a *color histogram* for each image in the database and comparing the color histogram of the query image with those in the database.

Mapping objects and queries onto feature vectors enables the use of appropriate multidimensional indexing mechanisms such as grid files and R-trees, with the query region appropriately expanded around the query point. Given that notions of similarity are, in general, diverse and application dependent, it is important to select appropriate *distance measures* in the multidimensional feature space that closely match the perception of similarity. For example, the distance between color histograms in QBIC is defined as a quadratic form function (a generalization of Euclidean distance), which takes into account the "cross-talk" between two similar colors (e.g., orange and red) (29). One issue that needs to be addressed is that conventional multidimensional indexing methods like grid files or R-trees suffer from the infamous "dimensionality curse," meaning that they result in access times that are exponential in the number of dimensions or they degrade to a linear search as the dimensionality increases. This is a serious problem for multimedia data indexing, since the number of dimensions (i.e., features) can in some cases exceed one hundred (31). One approach for dealing with high dimensionality is to map high-dimensional feature vectors to a lower number of dimensions using a distance-preserving transformation (18). Another approach is to design new, scalable indexing structures or to improve existing ones to scale to higher dimensions (31).

### Query Processing

Multimedia data types introduce a host of new challenges for the query processing component of a DBMS. A central issue is that the real-time access characteristics and the large volumes of CM data mandate the design of *effective resource management strategies* for multimedia query processing. Such strategies should: (1) provide guaranteed service levels for the storage and retrieval of CM data; (2) provide support for the temporal synchronization constraints defined between the CM components of complex multimedia objects; (3) provide support for user interaction (i.e., VCR-type functions); (4) allow for the retrieval of noncontinuous data concurrently with CM data; and, (5) maximize system throughput and reduce system response times. A number of these issues have been addressed in the context of CM storage servers, for example, the Fellini multimedia storage server developed at Bell Labs (32).

Given the limited amount of DBMS resources (e.g., memory, disk bandwidth, disk storage), providing service guarantees for CM data mandates a novel *admission control* component that decides whether to execute or postpone user queries. By initiating the execution of a query, the DBMS commits to satisfy the resource requirements (e.g., memory, disk bandwidth) of the CM streams involved throughout their duration. The service guarantees provided by the admission-control policy can be either deterministic (i.e., based on worst-case assumptions) (22) or stochastic (i.e., based on statistical models of system behavior) (33). Prior research has proposed novel data layout strategies, disk-scheduling algorithms, and buffer-management policies that take advantage of the highly sequential, stream-oriented access patterns to CM data in order to improve system throughput (34,35). A method proposed for handling conventional (noncontinuous) data requests and user interaction is to reserve a portion of the system's resources specifically for that purpose (32). Given that typical CM requests tend to execute for long periods of time, reserving resources in advance is important to ensure that both conventional requests and VCR-type functions observe reasonable response times. Other schemes for implementing VCR-type functionality have also been proposed in the literature, for example, storing a fast-forward/rewind version of a CM stream (36). Finally, efficient resource-scheduling algorithms for complex multimedia presentations with user-defined synchronization between various CM streams have recently been proposed (37).

Another crucial problem is the design of efficient query processing strategies for handling similarity queries. Given that users are interested only in the TOP-k objects, new query-execution methods are needed to produce the best k results efficiently (i.e., without materializing every intermediate result object that matches at all). Efficient strategies for processing TOP-k queries have been investigated in the context of the Garlic project at IBM Almaden (11,38). Other important query-processing issues for a multimedia DBMS include effective handling of tertiary storage and hierarchical storage structures (given the voluminous nature of multimedia data) (39), techniques for *sharing* CM streams among users to reduce resource demands (40,41), and fault-tolerant operation (42).

### Query Optimization

The declarative query language interface offered by the majority of conventional database systems has definitely been a major factor in their commercial success. This declarative querying paradigm was made possible due to the development of sophisticated query optimization techniques. Likewise, supporting high-level queries over multimedia databases mandates the development of appropriate optimization techniques. Multimedia query optimization is still a very open research area, with most important problems still waiting to be formulated and adequately solved.

The major issue here is that, for a multimedia DBMS, the querying model and, therefore, the resulting optimization questions, differ in many ways from conventional DBMS querying. Recent research has addressed some of the issues involved in optimizing similarity-based selection queries with ranked result sets over multimedia repositories (28). The main emphasis in this work was to explore optimization strategies designed for graded results and TOP-k semantics. Two additional issues that arise in the optimization of multimedia queries are intra/inter-media synchronization and QoS (3). Ignoring synchronization constraints during optimization can lead to excessive buffer requirements and underutilization of resources at run-time or unacceptable flaws in the presentation (e.g., glitches in the video, out-of-sync audio). QoS requirements are significant for optimization since they impact the space of execution alternatives as well as the metric of optimization. For example, a query generated by a fraud-detection application needs to be evaluated speedily with quality of video being of secondary importance. Thus the optimizer should obviously consider the option of returning a low-quality (e.g., compressed) version of the video if this results in lower response time. As of this writing these issues have yet to be addressed by the database or multimedia research community.

## BIBLIOGRAPHY

1. J. Widom, Research problems in data warehousing, *Proc. Int. Conf. Inf. Knowledge Management,* 1995, pp. 25–30.
2. *Comm. ACM, Special Issue on Digital Libraries,* **38**: 1995.
3. A. Shoshani, OLAP and statistical databases: Similarities and differences, *Proc. 16th ACM SIGACT-SIGMOD-SIGART Symp. Principles Database Syst.,* Tuscon, AZ, 1997, pp. 185–196.
4. R. J. Miller, O. G. Tsatalos, and J. H. Williams, DataWeb: Customizable database publishing for the web, *IEEE Multimedia,* **4** (4): 14–21, 1997.
5. P. Buneman, Semi-structured data, *Proc. 16th ACM SIGACT-SIGMOD-SIGART Symp. Principles Database Syst.,* 1997, pp. 117–121.
6. S. Abiteboul, Querying semi-structured data, *Proc. Sixth Int. Conf. Database Theory (ICDT '97),* 1997, pp. 1–18.
7. J. Gray et al., Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals, *Proc. 12th Int. Conf. on Data Engineering,* 1996, pp. 152–159.
8. P. O'Neill and D. Quass, Improved query performance with variant indexes, *Proc. 1997 ACM SIGMOD Int. Conf. Management Data,* Tucson, AZ, 1997, pp. 38–49.
9. O. Tsatalos, M. Solomon, and Y. Ioannidis, The GMAP: A versatile tool for physical data independence, *VLDB J.,* **5**: 101–118, 1996.
10. A. Y. Levy et al., Answering queries using views, *Proc. 14th ACM SIGACT-SIGMOD-SIGART Symp. Principles Database Syst.,* San Jose, CA, 1995, pp. 95–104.
11. M. J. Carey et al., Towards Heterogeneous Multimedia Information Systems: The Garlic Approach, *Proc. 5th Int. Workshop Res. Issues Data Eng.—Distributed Object Management (RIDE-DOM'95),* Taipei, Taiwan, 1995, pp. 124–131.
12. A. Y. Levy, D. Srivastava, and T. Kirk, Data model and query evaluation in global information systems, *J. Intell. Inf. Syst., Special Issue on Networked Information Discovery and Retrieval,* **5**: 121–143, 1995.
13. Y. Papakonstantinou, H. Garcia-Molina, and J. Widom, Object exchange across heterogeneous information sources, *Proc. 11th Int. Conf. Data Eng.,* Taipei, Taiwan, 1995, pp. 251–260.
14. D. Suciu, Query decomposition and view maintenance for query languages for unstructured data, *Proc. 22nd Int. Conf. Very Large Data Bases,* 1996, pp. 227–238.

15. P. Seshadri et al., Cost-Based Optimization for Magic: Algebra and Implementation, *Proc. 1996 ACM SIGMOD Int. Conf. on Management of Data,* Montreal, 1996, pp. 435–446.

16. A. Tomasic, L. Raschid, and P. Valduriez, Scaling heterogeneous databases and the design of DISCO, *Proc. Int. Conf. Distributed Computer Syst.,* 1996, pp. 449–457.

17. P. M. G. Apers, H. M. Blanken, and M. A. W. Houtsma (eds.), *Multimedia Databases in Perspective,* New York: Springer-Verlag, 1997.

18. C. Faloutsos, *Searching Multimedia Databases by Content,* Norwell, MA: Kluwer, 1996.

19. K. C. Nwosu, B. Thuraisingham, and P. B. Berra (eds.), *Multimedia Database Systems: Design and Implementation Strategies,* Norwell, MA: Kluwer, 1996.

20. B. Özden, R. Rastogi, and A. Silberschatz, Multimedia support for databases, *Proc. 16th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems,* Tucson, AZ, 1997, p. 223.

21. V. S. Subrahmanian and S. Jajodia (eds.), *Multimedia Database Systems: Issues and Research Directions,* New York: Springer-Verlag, 1996.

22. B. Özden, R. Rastogi, and A. Silberschatz, A framework for the storage and retrieval of continuous media data, *Proc. 1995 Int. Conf. Multimedia Comput. Syst.,* Washington, DC, 1995, pp. 2–13.

23. T. D. C. Little and A. Ghafoor, Sunchronization and storage models for multimedia objects, *IEEE J. Selected Areas Commun.,* **8**: 413–427, 1990.

24. S. R. Newcomb, N. A. Kipp, and V. T. Newcomb, The HyTime Hypermedia/time-based document structuring language, *Comm. ACM,* **34** (11): 67–83, 1991.

25. R. Price, MHEG: An introduction to the future international standard for hypermedia object interchange, *Proc. ACM Multimedia '93,* Anaheim, CA, 1993, pp. 121–128.

26. J. F. Allen, Maintaining knowledge about temporal intervals, *Comm. ACM,* **26** (11): 832–843, 1983.

27. S. Gibbs, C. Breiteneder, and D. Tsichritzis, Data modeling of time-based media, *Proc. 1994 ACM SIGMOD Int. Conf. Management Data,* Minneapolis, Minnesota, 1994, pp. 91–102.

28. S. Chaudhuri and L. Gravano, Optimizing queries over multimedia repositories, *Proc. 1996 ACM SIGMOD Int. Conf. Management Data,* Montreal, 1996, pp. 91–102.

29. C. Faloutsos et al., Efficient and effective querying by image content, *J. Intell. Inf. Syst.,* **3**: 231–262, 1994.

30. S. Chaudhuri, On optimization of multimedia queries, *Proc. ACM Multimedia '94 Conf. Workshop Multimedia Database Management Syst.,* San Francisco, CA, 1994.

31. D. A. White and R. Jain, Similarity indexing with the SS-Tree, *Proc. 12th Int. Conf. Data Eng.,* New Orleans, LA, 1996, pp. 516–523.

32. C. Martin et al., The *Fellini* multimedia storage server, in S. M. Chung (ed.), *Multimedia Information Storage and Management,* Norwell, MA: Kluwer, 1996.

33. G. Nerjes, P. Muth, and G. Weikum, Stochastic service guarantees for continuous media data on multi-zone disks, *Proc. 16th ACM SIGACT-SIGMOD-SIGART Symp. Principles Database Syst.,* Tucson, AZ, 1997, pp. 154–160.

34. S. Berson et al., Staggered striping in multimedia information systems, *Proc. 1994 ACM SIGMOD Int. Conf. Management Data,* Minneapolis, MN, 1994, pp. 79–90.

35. B. Özden et al., Demand paging for movie-on-demand servers, *Proc. 1995 Int. Conf. Multimedia Comput. Syst.,* Washington, DC, 1995, pp. 264–272.

36. B. Özden, R. Rastogi, and A. Silberschatz, On the design of a low-cost video-on-demand storage system, *ACM Multimedia Systems,* **4** (1): 40–54, 1996.

37. M. N. Garofalakis, Y. E. Ioannidis, and B. Özden, Resource scheduling for composite multimedia objects, *Proc. 24th Int. Conf. Very Large Data Bases,* New York, 1998.

38. R. Fagin, Fuzzy queries in multimedia database systems, *Proc. 17th ACM SIGACT-SIGMOD-SIGART Symp. Principles Database Syst.,* Seattle, WA, 1998.

39. S. Ghandeharizadeh and C. Shahabi, On multimedia repositories, personal computers, and hierarchical storage systems, *Proc. ACM Multimedia '94,* San Francisco, CA, 1994, pp. 407–416.

40. A. Dan, D. Sitaram, and P. Shahabuddin, Scheduling policies for an on-demand video server with batching, *Proc. ACM Multimedia '94,* San Francisco, CA, 1994, pp. 15–23.

41. M. N. Garofalakis, B. Özden, and A. Silberschatz, Resource scheduling in enhanced pay-per-view continuous media databases, *Proc. 23rd Int. Conf. Very Large Data Bases,* Athens, Greece, 1997, pp. 516–525.

42. B. Özden et al., Fault-tolerant architectures for continuous media servers, *Proc. 1996 ACM SIGMOD Int. Conf. Management Data,* Montreal, 1996, pp. 79–90.

Minos N. Garofalakis
Bell Laboratories
Renée J. Miller
University of Toronto

**VERY SMALL APERTURE TERMINALS.** See VSAT Networks.

**VIDEO AMPLIFIERS.** See Wideband amplifiers.

**VIDEO CAMERAS.** See Camera calibration for image processing.

**VIDEO CAPTURE.** See Image processing equipment.

**VIDEO CODEC.** See Data compression for networking.

**VIDEO CODING.** See Image and video coding.

**VIDEO CODING (METHODS).** See Video compression methods.