A Massive Repository for the National Medical Knowledge Bank

Warren M. Sterling, Ph.D.
Teradata Development Division
NCR Corporation
100 N. Sepulveda Blvd.
El Segundo, CA 90245
warren.sterling@ncr.com
tel +1-310-524-6448
fax +1-310-524-5515

Abstract
This paper describes a massively parallel object relational (O/R) database used in an advanced development program to create a comprehensive medical information system called the National Medical Knowledge Bank (NMKB). This database, named the Teradata Object Relational Database (TOR), was developed for massive data warehouse applications in the multiple terabyte range. We present a brief discussion of the NMKB architecture, its healthcare applications, and the requirements of TOR to support the NMKB. We then concentrate on the architecture of TOR including scalability features, large object store, system and user defined functions, geo-spatial data types, and the internal indices required to support these features.

1 Introduction

A medical knowledge bank is an advanced repository of specialized medical experience and knowledge, featuring easy capture of information, with massive storage and links to a vast selection of supporting resources enabling users to retrieve and learn, any time and place. A coalition of companies and healthcare organizations formed a joint venture to create a medical information system called the National Medical Knowledge Bank (NMKB) [1,2]. This multi-year program was sponsored in part by a grant from the National Institute of Standards and Technology Advanced Technology Program (Project ID 1995-10-0030A Under (95-10) Information Infrastructure for Healthcare). Current participants are:

- Allegheny-Singer Research Institute - the program lead
- MCP Hahnemann University School of Medicine and School of Nursing – primary medical advisors and responsible for implementation of several applications
- NCR– responsible for implementation of several of the applications, the user interfaces, and the content repository
- Millennium Healthcare Solutions – responsible for project management and commercialization plans for the NMKB

The bedrock upon which the NMKB rests is the Teradata Object Relational database (TOR). It serves as the central data repository, called the Multimedia Registry, capable of storing, retrieving and analyzing multimedia data such as medical images (e.g., MRI scans, digital x-rays images), video (e.g., surgical and other health related procedures, conference presentations), other imagery (e.g., photographs, graphics to accompany video presentations) and text documents. This paper will concentrate on the architecture of

TOR, describing its design for scalability, large objects, and system and user defined functions that operate on data, including large objects, stored in the database.

2 The National Medical Knowledge (NMKB) Bank Project

The ambitious goal of this grant was to revolutionize healthcare through the development of advanced tools for computer-assisted diagnosis utilizing case-based reasoning (CBR), initial and continuing medical education (IME, CME), virtual medical conferences, and storage and management of patient records for illustrative cases. The key challenge for database people is the need to develop a medical multimedia data warehouse system that will linearly scale when content volume and concurrent usage increases, improving productivity as well as patient care.

Work on the program has focused on the development of the NMKB components such as the Multimedia Registry, simple user interfaces for healthcare practitioners, interactive web-based medical training, medical abstract search capability for evidence-based medicine, computer-assisted medical diagnosis utilizing CBR, the formatting of patient medical records to support CBR, and virtual conferences.

3 NMKB Architecture

Figure 1 shows the general architecture of the National Medical Knowledge Bank. It is a 3-tiered web-based architecture. Users access the knowledge bank via thin clients at the top tier. The clients only need an Internet Explorer browser with RealPlayer to display streaming video and Macromedia Flash to display animations. The middle tier is the web application layer. This layer consists of the Internet Information Server, and the various applications described in the next section. In addition, this section has the framework that handles user access control, including login security, and record keeping for students, including courses taken and certification logs.

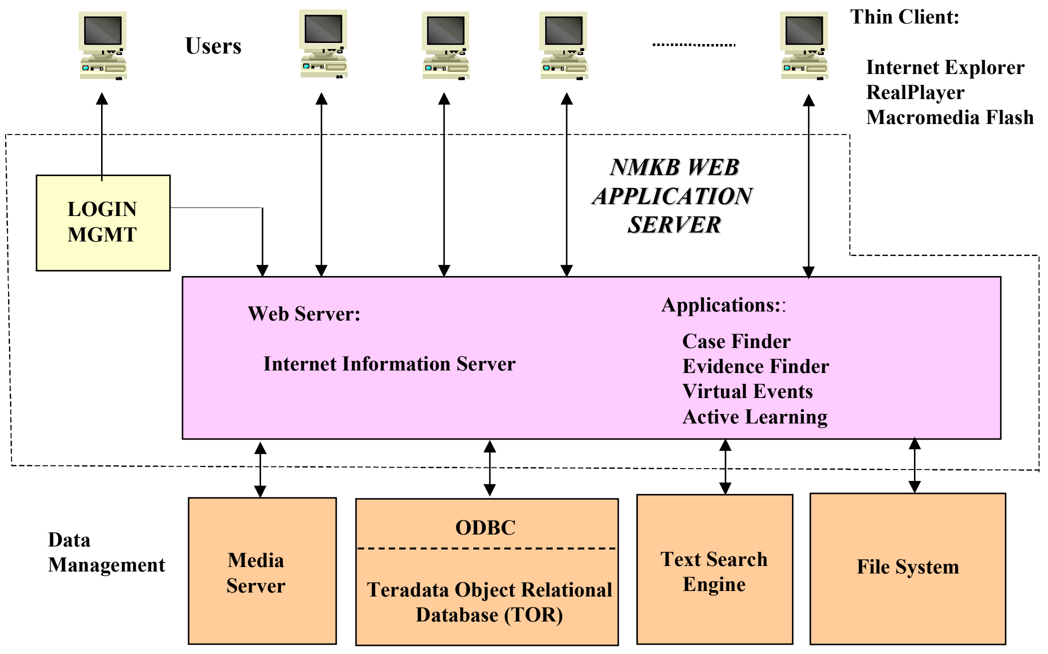


Figure 1. General Architecture of the National Medical Knowledge Bank

The third tier consists of the data management elements of the NMKB. The media server stores video in *.rm format for direct streaming to users on demand. Multiple applications in the NMKB have streaming video content. The video format supports slide presentation synchronized with the video.

TOR stores NMKB content, including large objects such as medical imagery. It also stores a key component of the NMKB – the content index. This index, based on the Dublin Core [3], contains a metadata record for every piece of content in the knowledge bank. Users of the system can directly search the index to find content matching various constraints. A system application can also search the index to find not only content to be used by that application, but also content from other applications that may be related to specific topics being addressed in the first application. Finally, content authors can use the index to search for content that can be reused.

The text search engine is used to allow full text indexing of information stored in the content index. This can range from simple medical terms stored in the content index metadata to full medical article abstracts. In addition, the text search engine employs concept mapping to expand the search to include related concepts, synonyms and other linguistically related terms (See **Evidence Finder**, below).

4 NMKB Applications

There are four major applications comprising the NMKB: Evidence Finder, Virtual Events, Intelligent Agent-based Continuing Healthcare Education, and Case Finder. All of these applications operate against a data repository (Multimedia Registry) that contains all content required by each application and also contains a comprehensive index containing metadata describing every piece of content in the NMKB.

4.1 Evidence Finder

Evidence Finder supports evidence-based medicine in a clinical environment. It retrieves the "essence of the essence" of recent published literature. It reduces the need for colleague consultation and provides "think with me" functionality where a healthcare practitioner engaged in clinical work can utilize the application while working through a specific case. Healthcare practitioners can search the repository for medical abstracts and "pearls" (salient points and conclusions of a medical article.) Searches can be narrowed by specific disease domain (e.g., diabetes, lipid disorders, asthma, osteoporosis), sub-topic (e.g., diagnosis, prevention, outcomes). Additionally, every article is rated by its evidence quality level. There are 8 levels recognized at this time. From highest evidence quality level to lowest, they are: Randomized controlled testing, trial, case study, population study, guidelines, review article, meta-analysis, cost-benefit analysis. The search can be further constrained by specifying a minimum evidence quality level for retrieved abstracts. Finally, the search can be further constrained by specifying keywords in the abstract, pearls, or other relevant fields such as author and Unified Medical Language System codes. A key element of keyword search is the ability to do concept mapping. For example, a search of abstracts in the diabetes disease domain may be further constrained by a keyword of "Glucophage", a brand name diabetes drug. The search is automatically expanded to include the generic form of the drug – Metformin.

Likewise, a search utilizing the keyword "stroke" is automatically expanded to include the term "brain attack". This is accomplished through the use of the text search engine utilizing a semantic network based on the Unified Medical Language System (UMLS) developed by the National Institute of Health. A full discussion of the text search engine is beyond the scope of this paper.

Although Evidence Finder is primarily targeted to the search of medical abstracts, searches will also identify other content in the NMKB where the content metadata satisfies the search criteria. This is an example of content sharing among applications. For example, the previous search in the diabetes domain with a keyword of "Glucophage" will return abstracts that satisfy the constraint but will also return a Primary Care Ground Rounds video presentation segment that covers the use of Metformin.

4.2 Virtual Events

Virtual events are medical presentations captured on video either at medical conferences or grand rounds (invited medical presentations at medical schools). Effectively, these videos are streamed to viewers along with a synchronized slide show corresponding to the original visual aids used by the speaker. This application allows for asynchronous, discretionary viewing by users and presents a significantly lower cost option in terms of time and travel. Most of the virtual events have been accredited for Continuing Medical Education (CME) credits. In some cases, there is an alternative text-based version of the presentation. The presentations have been indexed for fast navigation. Furthermore, the presentations have been further subdivided into segments that are indexed. These segments can be shared with other applications or reused by a content author. Figure 2 shows an example of a virtual event from a recent conference on brain attack. The streaming video shows the presenter, with the slides being displayed to the left. Along the bottom, the thumbnail representations of the slides are used for navigation.

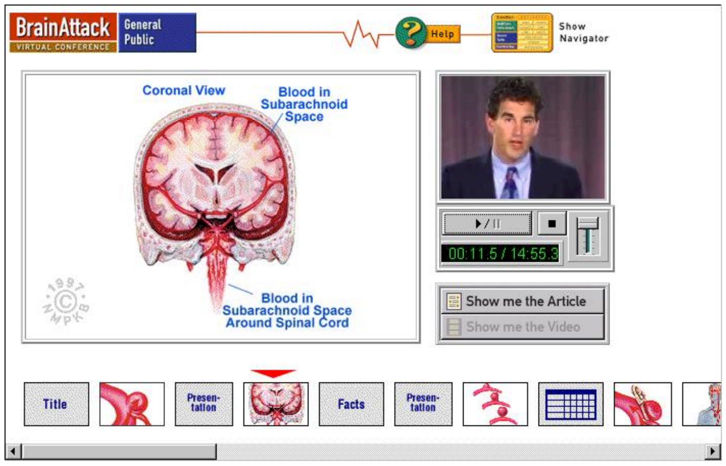


Figure 2. Presentation Format for Brain Attack Virtual Conference

4.3 Intelligent Agent-based Continuing Healthcare Education

This application delivers a personalized, active education experience over the web. In addition, it dynamically measures the student's mastery of the subject matter as the student progresses through learning exercises. There is a strong multimedia component to each learning exercise, including streaming video, medical images and animations. Lessons can be completed at the convenience of the student, in both time and location. Lessons can be completed in multiple sessions; the application keeps track of student progress through previous sessions. It has been designed for problem-based learning where the student takes an active role in proceeding through the learning exercise by choosing activities, the order in which to perform them, analyzing results, making decisions and indicating the reasoning behind those decisions. The students' decisions in all these areas are monitored and used to evaluate student mastery of the subject matter, and to provide feedback to the student upon completion of the exercise. For example, in a series of case studies on fatigue, a student takes the role of a nurse-practitioner working with a patient presenting with fatigue. The student chooses from activities such as conducting a History of Present Illness interview with the patient (including the choice of questions to ask), conducting a Review of Systems interview, reviewing the case scenario, reviewing the patent's medical chart, performing a physical exam, ordering and reviewing laboratory tests, diagnosing the problem and developing a plan-of-care. This application, called the Active Learning Framework (ALF), makes extensive use of intelligent agents. These are task-specific and persistent entities that monitor the environment to provide personalized interaction, and collaborate to perform complex tasks. The key agents used in the ALF are:

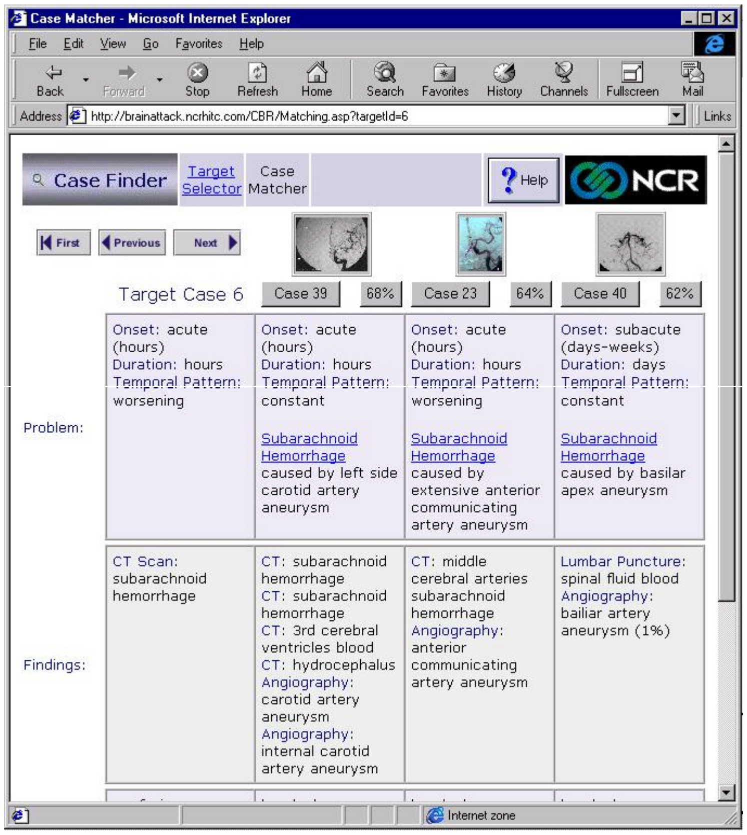
- Student Modeler Agent: this agent understands the student history including subject knowledge level and previous lessons taken and mastered.
- Lesson Planning Agent: this agent understands the lesson material and how to present it within the framework of the application
- Tutor Agent: this agent monitors the student progress through a lesson and performs several functions. It determines when a student can benefit from additional specific material and presents that material. It also inserts activities designed to evaluate the student's thought process during the lesson. For example, the student may have previous selected several possible diagnoses for a case. After some time, the tutor agent will ask the student to take information learned during the course of the lesson and use it to support or refute each of possible diagnoses.
- Performance Agent: this agent monitors the performance of all other agents and clones additional agents, if necessary. For example, if a lesson planning agent is falling behind because of increased workload, the performance agent will instantiate another lesson planning agent to share the load.

The ALF is a reusable framework. New lessons can be implemented simply by adding new content to the system and populating database tables that support the framework. No new coding is required. The ALF is also "horizontal" technology; that is, it is completely applicable to domains outside the medical training domain.

Lessons implemented in the Active Learning Framework have been accredited for Continuing Nursing Education credits.

4.4 Case Finder

This application employs case-based reasoning technology (CBR) to assist a clinician in comparing a target medical case against a database of exemplar medical cases and finding those cases that most closely match the target case. This eases the clinician burden of reading and recalling past cases. Typically, it would be used in cases that are difficult to diagnose or treat because of poor correlation between the symptoms and findings, or because of the presence of multiple medical problems, a common occurrence not often covered well in textbook cases. In general, CBR is the reuse of previous solutions for a current problem. Cognitive science research shows that experts use experience when reasoning about a problem, rather than first principles. As in the case of Evidence Finder, Case Finder functions in a "think with me" mode, aiding the clinician working through a case. Clearly, it does not usurp the clinician's responsibility to diagnose and treat; however, it can help suggest avenues of exploration the clinician might otherwise miss. CBR makes use of similarity metrics to compare various dimensions of cases (e.g., symptoms, physical exam and test findings, family history, patient demographics, patient life style, etc.) between the target case and exemplar cases. Every dimension is evaluated and then weighted to arrive at an overall similarity ranking. A detailed discussion of the metrics is beyond the scope of this paper; however, they range from simple numerical comparisons (e.g., exact matches, ratios, range matches) to indirect matches (comparison not only against actual source values but also against possible source values for a given problem). In evaluating similarity metrics for text fields, the application uses a semantic network based on the UMLS to do comprehensive matching. For example, a match should be generated if a field in the target case specifies *stroke* and the same field in an exemplar case specifies *brain attack*. Figure 3 shows a typical retrieval result. The Target Case 6 is shown in the leftmost column. The three columns to the right show three cases (39, 23, 40) that are


Case Matcher - Microsoft Internet Explorer
File Edit View Go Favorites Help
Back Forward Stop Refresh Home Search Favorites History Channels Fullscreen Mail
Address
Links
Case Finder
Target Selector
Case Matcher
Help
NCR
First
Previous
Next
Internet zone

http://brainattack.ncrhitc.com/CBR/Matching.asp?targetId=6

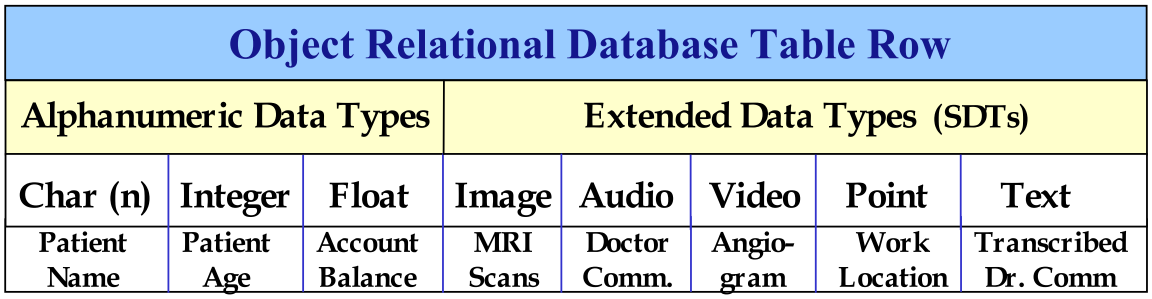
	Target Case 6	Case 39 68%	Case 23 64%	Case 40 62%
Problem:	Onset: acute (hours) Duration: hours Temporal Pattern: worsening	Onset: acute (hours) Duration: hours Temporal Pattern: constant Subarachnoid Hemorrhage caused by left side carotid artery aneurysm	Onset: acute (hours) Duration: hours Temporal Pattern: worsening Subarachnoid Hemorrhage caused by extensive anterior communicating artery aneurysm	Onset: subacute (days-weeks) Duration: days Temporal Pattern: constant Subarachnoid Hemorrhage caused by basilar apex aneurysm
Findings:	CT Scan: subarachnoid hemorrhage	CT: subarachnoid hemorrhage CT: subarachnoid hemorrhage CT: 3rd cerebral ventricles blood CT: hydrocephalus Angiography: carotid artery aneurysm Angiography: internal carotid artery aneurysm	CT: middle cerebral arteries subarachnoid hemorrhage Angiography: anterior communicating artery aneurysm	Lumbar Puncture: spinal fluid blood Angiography: bailiar artery aneurysm (1%)

Figure 3. Case Finder Retrieval of 3 Cases Similar to Target Case

evaluated as most similar to the target case, with Case 39 having the closest match (68%). The clinician can click on the case number and access all details of the case summary. The clinician can click on the percentage match and find the details of the similarity metrics calculations. Note that the active link Subarachnoid Hemorrhage is shown in each case. The application examines the diagnosis of each case. If a search of the content index shows additional content in the knowledge bank for that diagnosis, then an active link is established. If the clinician clicks on the link, then a list of relevant content is displayed and available for viewing.

5 Object Relational Database Fundamentals

An O/R database such as TOR is primarily a relational database design that has extended data types (complex and/or large data types), internal indices that improve performance of the database when accessing certain extended types, and functions that operate on these extended types in order to analyze or manipulate them. Figure 4 illustrates a sample TOR row representing information on a medical patient. Note that the left side of the row has fields that are typical of the traditional alphanumeric data types found in relational databases. The right side of the row has extended data types, called System Defined Types (SDTs), associated with TOR. These data types can store values that are quite large. For example, the *Image* datatype in this example could store a series of MRI scans with a total size measured in Megabytes. Likewise the *video* data type storing a digitized video of an angiogram will be large. TOR has a storage management system that efficiently handles such large data types.



Object Relational Database Table Row							
Alphanumeric Data Types			Extended Data Types (SDTs)				
Char (n)	Integer	Float	Image	Audio	Video	Point	Text
Patient Name	Patient Age	Account Balance	MRI Scans	Doctor Comm.	Angio-gram	Work Location	Transcribed Dr. Comm

Figure 4. Sample Table Row in TOR for Medical Patient Record

The sample row also shows a data type for audio. Typically this would store a digitized audio clip.

Note that the data types can be defined more narrowly than simply *image* or *video*. For example, the *image* data type could be defined as a *JPEG* data type [4], and only JPEG encoded images would be stored in this field. Likewise, the *video* type could be defined as an *MPEG-2* [5] data type. Conversely, data types can be broadly defined as Binary Large Objects (*BLOBs*) and Character Large Objects (*CLOBs*). Such data types imply no structure whatsoever to the content of the data type. It is up to the application accessing the database to understand or determine the structure of the data in order to properly display, analyze or manipulate it.

In addition to the extended data types, TOR has system defined functions (SDFs) that operate on some extended data types. Users can also define application specific functions that can be loaded into the database and operate on data types defined in the database. These functions, called User Defined Functions (UDFs), are usually defined in the C++ programming language. Table 1 lists some examples of SDFs or UDFs that could be associated with specific data types.

System Defined Types (SDTs)	***System/User Defined Functions (SDFs/UDFs)***
Audio	Word Spotting; Voice Recognition for Identification
Image	Tumor Classification in MRI Scans; Color Histogram
Video	Extraction of Video Segment (start and end points)
Geo-spatial	Map/Image Overlay; Distance Between Points; Polygon Overlap
Text	Language translation; Word/Phrase Matching and Counting

Table 1. Examples of Functions for System Defined Types

One particularly useful set of SDTs in TOR is geo-spatial data types. Examples include *points*, *polylines*, *polygons*, and *circles*. In the example of Figure 4, the *Point* data type represents the geo-coded location (earth latitude/longitude) of the patient's home address. Geo-spatial data types are discussed in more detail below and in [6].

Inheritance plays an important role in the definition of SDTs. SDTs will often be derived from other SDTs. As such they inherit the properties of these SDTs, including the SDFs and UDFs defined for the original SDTs. For example, *text*, *video*, *BLOB* and *audio* SDTs can all be derived from a 1-dimensional array SDT.

As in conventional relational database systems, indices play an important role in the performance of an O/R database. Efficient execution of geo-spatial queries in an O/R database requires the use of an R-Tree index [7]. R-Trees support efficient 2-dimensional searches, as opposed to the classic B-Tree, a 1-dimensional index often used in conventional relational database systems. Likewise, an inverted text index supports the efficient execution of queries that require full-text searches of *text* data types.

The query language for O/R databases is SQL3. SQL3 contains constructs for handling the extended types for O/R databases, including the ability to create User Defined Functions.

6 TOR Requirements for NMKB

The NMKB project initially placed a number of requirements on TOR involving the storage and processing of large complex objects and the ability to support a system with heavy concurrent usage and massive amounts of storage. There have been various changes in project direction that have resulted in a relaxation of these requirements, and at the current time, the level of usage and content is below that which requires TOR's scalability. Nonetheless, TOR was designed to meet the stringent initial requirements discussed here.

Patient records and case summaries all contain not only alphanumeric data, but also image and video data from medical tests (e.g., MRI scans, x-rays, video angiograms, EKGs). It must be possible to store these objects within the database table in exactly the same manner as alphanumeric data types are stored – in rows (tuples). Large objects can reach a theoretical size of 2 GBytes.

Initially, the NMKB project planned to use automated brain tumor analysis of MRI scans as part of the similarity evaluation function used in the Case Finder application. This requires the ability to create application specific UDFs that can be loaded into the database to operate on MRI images stored in the database.

In the Virtual Events application, slides accompany the video presentation. For indexing purposes in some presentation series, the slides have to be reduced to thumbnail images for use in establishing an index through each presentation. Image manipulation software can be implemented as a UDF. With the slides loaded into database tables, this UDF can be used manipulate the slides and create the thumbnail images.

One common medical image format is known as DICOM3. Medical diagnostic imaging systems often employ this format when they generate digital images directly. This imaging format contains, among other things, the image itself and a comprehensive metadata header that completely identifies the patient, doctor, hospital, date/time of test/imaging procedure, and test equipment details, including model information and control settings at the time of the test. TOR was required to implement DICOM3 as a data type. This includes a number of SDFs that extract the image and key fields of the header.

The ultimate goal of this project is to implement a system accessible on a national level. As a result, the database must be capable of acceptable performance while supporting large numbers of concurrent users operating against a database potentially in the Terabyte range. In order to avoid having to implement a database system designed from the outset to handle the expected usage, TOR has to be scalable, allowing it to grow gracefully through the addition of processing power and storage capacity while maintaining a linear performance characteristic.

7 TOR Architecture

TOR is a highly parallel object relational database, utilizing a share-nothing architecture. The TOR architecture is based on the Teradata Database architecture. The Teradata Database is a proven, mature massively parallel relational database that is used to implement extremely large data warehouses in mission-critical environments. Teradata data warehouses are as large as 100 terabytes in size and can operated with hundreds of concurrent users running a combination of relatively simple transaction queries and complex decision support queries. The Teradata Database runs on a share-nothing massively parallel processing system (MPP) from NCR [8]. TOR has been designed to run on the same MPP system. The MPP system consists of a set of Symmetric Multiprocessor (SMP) nodes, each with its own processor memory and disk array storage that are not shared with other nodes; hence, the term "share-nothing". The SMP nodes communicate over an active communication network called the BYNET [9], based on a Banyan interconnect topology. This proprietary network is itself scalable, increasing its aggregate bandwidth as nodes are added to the system. The Teradata Database today can run on a system as small as a single node up to a theoretical maximum of 512 nodes in a single MPP system. The largest configuration actually installed is 176 nodes. TOR currently can be sized from 1 to 4 nodes.

Figure 5 shows the system architecture for TOR. TOR consists of the Query Coordinator and a set of nodes called Data Servers. To maximize query performance in a parallel database, every table in the database is distributed across all Data Server nodes. If a database table on a 4-node system contains 1,000,000 rows, then each node of the system will hold 250,000 rows of that table. This becomes extremely important for queries that

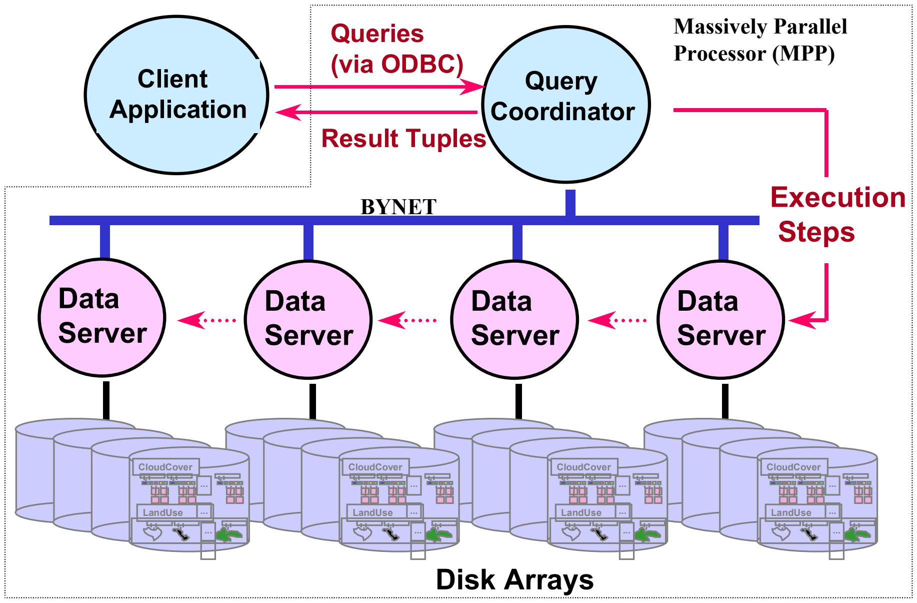


Figure 5. TOR System Architecture

require a "full file scan" (reading every row in the table). Each Data Server can operate against that portion of the table stored on that Data Server, without concern for the portions of the table stored on the other nodes. Such a query can be completed in ¼ the time it would take if the table was stored on only one node, and the Data Server was forced to access every row of the table.

The Query Coordinator can run on a separate node or run on one of the Data Server nodes. Client applications communicate with TOR via the Query Coordinator. Queries are submitted by the Client Application to the Query Coordinator utilizing ODBC, a standard API for relational database queries that has been extended to handle additional data types for TOR. The Query Coordinator parses the query, optimizes it and schedules execution steps that are dispatched to the Data Servers. Upon receiving an execution step, a Data Server runs the step against its portion of the database table or tables, generating a partial result set. This result set is then passed back to the Query Coordinator, which combines the result sets from all Data Servers. The final result set is then passed back to the Client Application. When an ordered result set is required, each Data Server orders its partial result set. When that operation is complete on all Data Servers, then the Query Coordinator orchestrates the return of result sets from the Data Server, assuring that the full result set arrives back at the Query Coordinator in proper order.

Some queries, such as prime key retrievals, only require access to one specific row of a table. In that case, the Query Coordinator sends an execution step only to the Data Server where the row resides and that Data Server responds with its result set. The other Data Servers do not participate in the query.

TOR uses two forms of parallelism for improved performance: pipelined execution and partitioned execution. Referring to the left side of Figure 6, pipelined execution relies on the fact that a query is broken up into one or more operators. Operators take in one or two input relations from input streams and produce one output relation written to an output stream that serves as one input stream to the next operator. As a result query execution flows in a pipeline fashion. In Figure 6, the operators are Select, Join and Aggregation. On the right side of Figure 6, partitioned execution is simply the fact that identical steps are being executed concurrently on different nodes.

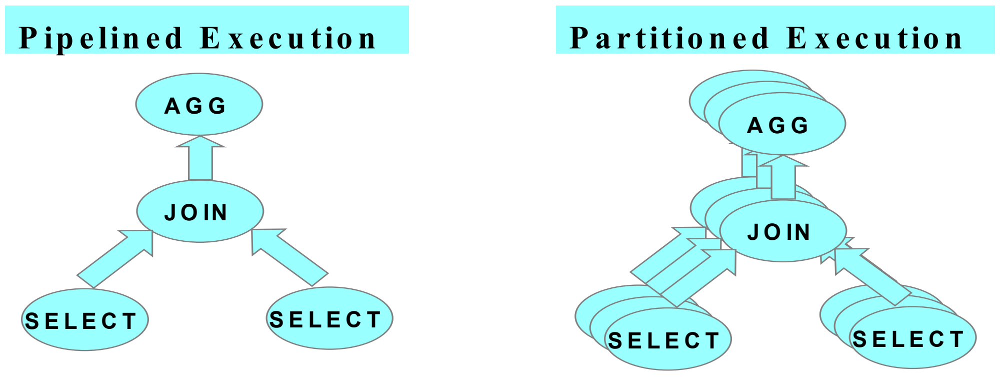


Figure 6. Two Forms of Parallel Execution

The database optimizer is the key to high performance of an object relational database system. Most queries can be accomplished via a number of different plans. The key is to find the least expensive plan in terms of data movement and row access. To accomplish this, optimizers analyze various parameters such as the size of each table referenced in the query and the sensitivity of terms in a constraint clause. Based on these parameters the optimizer will decide the best order for evaluating terms in a constraint clause and select a join plan that minimizes row redistribution. The TOR optimizer is based on the OPT++ query optimizer [10]. It is a third generation query optimizer that knows how to work with the extended data types in TOR. It is also easy to update for new query operators and data types.

7.1 TOR Large Object Store

Normally, all data in a table row is in-line; that is, the data in all columns of the row is stored contiguously. While this works well for the standard alphanumeric data types associated with relational databases, it would result in unacceptable performance problems for object relational databases when the columns contain large objects such as images and video. Rows could not be efficiently blocked. Furthermore, row redistribution operations required for query joins would be extremely inefficient. Large objects should never be moved from one node to another unless the object itself must be accessed explicitly on another node. To this end, large objects are not stored in-line. Instead an object identifier (OID) is stored in the row, and the object itself is stored elsewhere, tiled and compressed. Figure 7 illustrates this with a simple example using a 2-dimensional raster image of cloud cover density stored in the **Cloud Cover** table. The table contains three columns: a column named **date** of type *date*, a column named **density** of type *2Draster*, and a column named **instr** of type *string*. Stored in place of the actual raster is a raster header containing the tile size, the bounding box for the entire image, and the OID pointing to a header for the tiles themselves. The tile header contains a pointer to each tile making up the entire raster image. This repeats for every row of the **CloudCover** table, as shown in Figure 7.

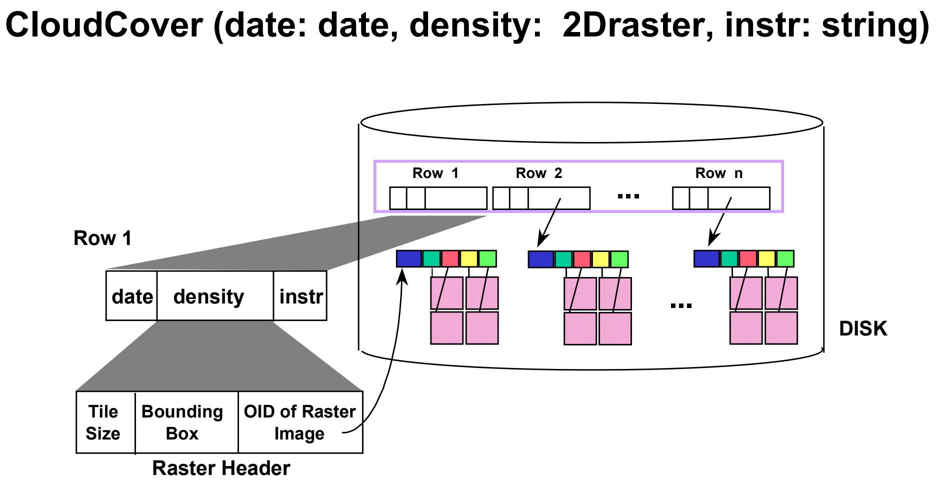


Figure 7. Example Illustrating Tiling of Large Objects in TOR

This method of large object storage allows the table rows in TOR to be manipulated as if they contained only alphanumeric data types. This method also minimizes the number of copies of large objects. Intermediate and temporary result sets share (by reference) the original copy of the large object.

7.2 TOR Geo-spatial Capability

Spatial queries are not used explicitly in any of the NMKB applications; however, the geo-spatial capability of TOR is one of its strongest features. Table 2 lists the geo-spatial data types that exist in TOR and Table 3 gives some examples of geo-spatial operators that are used with the data types. Representative operands are shown for each operator but this is not a complete set of valid operands or operators. Spatial queries are relatively straightforward and intuitive with geo-spatial data types. For example, given all patients' geo-coded home addresses and a *circle* data type with a radius of two miles and centered on the geo-location of a hospital, it now becomes quite simple to construct an SQL spatial query that finds all patients that live within two miles of the hospital.

Geo-spatial Data Type	***Description***
Point	Single geo-located point in 2D space.
Polyline	Line with 1 or more connected segments, each segment endpoint geo-located
Polygon	Closed polyline
Swisscheese Polygon	Polygon with wholly-contained smaller polygons
Circle	Geo-located center point and defined radius
Rectangle	Geo-located rectangle
2Draster	2D geo-located raster image

Table 2. Geo-spatial Data Types in TOR

Geo-spatial Operator with Operands	***Description***
Clip (image, polygon)	Return portion of image within area of polygon
Overlaps (polygon, polygon)	Returns TRUE if polygons overlap
ContainedBy (point, polygon)	Returns TRUE if point contained in boundary of polygon
Contains (circle, point)	Returns TRUE if circle contains point
Distance (point, point)	Returns distance between two points
Closest (polyline, point)	Returns polyline value in a column that is closest to a specified point (aggregate operation)

Table 3. Representative Geo-spatial Operators

Geo-spatial data can be declustered (spread across the nodes of a multi-node TOR system) by mapping 2-dimensional space across the nodes and storing instances of geo-spatial data types on the nodes corresponding to their geo-location. When geo-spatial objects with finite areas such as polygons cross node boundaries then that object is replicated on each affected node. This allows for more efficient execution of geo-spatial operators.

8 Other TOR Research Applications

TOR is being used in several other funded research projects. In a European Commission-funded project called ARTISTE, TOR is being used as the data repository for a distributed system that will give galleries and museums, publishers, distributors, researchers and users of art images, as well as the multimedia information market as a whole, a more efficient system for classifying, storing, linking matching and retrieving art images. This is a joint venture project involving several companies, a university research organization, and four European museums (Uffizi Gallery, Louvre, Victoria & Albert, National Gallery). More information on this project can be found on web site www.artisteweb.org.

TOR is also being used in a geo-spatial data integration project at the University of Southern California Integrated Media Systems Center called Active Navigation of Satellite Images. This project combines open source data available on the Internet about countries throughout the world with satellite imagery and maps from around the world. The system will superimpose on images and maps information ranging from locations of airports, buildings, rivers, and bridges to nuclear power plants and earthquake faults. The system dynamically retrieves the required geo-coded data from online sources and integrates it with the appropriate imagery. The geo-spatial capability of TOR plays a strong role in determining what data must be displayed on the imagery.

9 Conclusions

The NMKB consists of a unique set of multimedia and web-enabled applications that could revolutionize healthcare training and clinician support in the future. The potential for massive amounts of both alphanumeric and multimedia data and high concurrent usage requires that the Multimedia Registry be a scalable database that can also handle large complex objects. The highly parallel share-nothing design of TOR, coupled with its ability to store, analyze and manipulate large objects, indicates that TOR can fulfill the role of the Multimedia Registry in the NMKB.

10 Acknowledgements

A number of individuals and organizations were responsible for implementing the NMKB applications. The Evidence Finder application was implemented by a team at MCP Hahnemann University School of Medicine under the direction of Russell Maulitz, MD, Ph.D., Chief, Division of Medical Informatics. The virtual event grand round video presentations were implemented by a team at MCP Hahnemann University School of Medicine under the direction of Arnold Smolen, Ph.D., Associate Dean for Information Technology. The medical conference video presentations were implemented by a team led by Mark Tarlton, formerly of the NCR Human Interface Technology Center. The Case Finder application was designed and implemented by Barbara Ericson and Michel Robert in NCR Corporate Technology. The Active Learning Framework was designed and implemented by Betty Whitaker, Ph.D., Terry Chism, and Marcia Crosland, Ph.D. in NCR Corporate Technology. All NCR Corporate Technology work on the project was directed by Bob Simpson, Ph.D., Director, Intelligent Systems. The content for the fatigue case studies was prepared by a team at the MCP Hahnemann University School of Nursing under the direction of Debra McGrath, RN, MSN, CRNP, Assistant Professor. TOR was implemented by the TOR Engineering organization under the direction of Diane Wewerka in the NCR Teradata Development Division.

References

[1] W. Sterling. The medical knowledge bank: a multimedia database application, *Proceedings of the Multimedia Technology & Applications Conference (MTAC '98)*, Anaheim, California, pp. 66-70, September 15-17, 1998

[2] W. Sterling. The national medical knowledge bank, *Proceedings of the 24th Annual International Conference on Very Large Data Bases (VLDB'98)*, New York City, New York, pp. 637-640, August 24-27, 1998

[3] Dublin Core metadata element set, version 1.1: reference description, 1999. http://purl.org/dc/documents/rec-dces-19990702.htm, Dublin Core Metadata Initiative, The Dublin Core Directorate, hosted by the Online Computer Library Center (OCLC) Office of Research and Special Projects.

[4] W.B. Pennebaker, J.L. Mitchell. *JPEG Still Image Data Compression Standard*, New York, Van Nostrand Reinhold, 1993

[5] D. Le Gall. MPEG: A video compression standard for multimedia applications, *Communications of the ACM*, 34(4), pp. 46-58, April 1991

[6] M. Stonebraker. *Object relational DBMSs the next great wave*, San Francisco, Morgan Kaufmann, 1996

[7] N. Beckmann, H.-P. Kriegel, R. Schneider, B. Seeger. The r*-tree: an efficient and robust access method for points and rectangles, *ACM SIGMOD*, pp. 322-331, May 1990

[8] *NCR WorldMark 5200 server large scale data warehousing system*, Product Description EB-1093 0899, Dayton, Ohio, NCR Corporation, 1999

[9] R.J. McMillen, The BYNET v2.0 interconnection network, *Conference Record of Hot Interconnects*, IEEE, pp. 117-135, August 6, 1998

[10] N. Kabra, D.J. DeWitt. OPT++ an object oriented implementation for extensible database query optimization, *VLDB Journal* 8(1), pp. 55-78, 1999